Virtual Adversarial Ladder Networks for Semi-Supervised Learning

Saki Shinoda[1], **Daniel E. Worrall**[2] **& Gabriel J. Brostow**[2]
Computer Science Department
University College London
United Kingdom
[1]saki.shinoda.16@ucl.ac.uk
[2]{d.worrall, g.brostow}@cs.ucl.ac.uk

Abstract

Semi-supervised learning (SSL) partially circumvents the high cost of labeling data by augmenting a small labeled dataset with a large and relatively cheap unlabeled dataset drawn from the same distribution. This paper offers a novel interpretation of two deep learning-based SSL approaches, ladder networks and virtual adversarial training (VAT), as applying distributional smoothing to their respective latent spaces. We propose a class of models that fuse these approaches. We achieve near-supervised accuracy with high consistency on the MNIST dataset using just 5 labels per class: our best model, ladder with layer–wise virtual adversarial noise (LVAN-LW), achieves $1.42\% \pm 0.12$ average error rate on the MNIST test set, in comparison with $1.62\% \pm 0.65$ reported for the ladder network. On adversarial examples generated with L_2-normalized fast gradient method, LVAN-LW trained with 5 examples per class achieves average error rate $2.4\% \pm 0.3$ compared to $68.6\% \pm 6.5$ for the ladder network and $9.9\% \pm 7.5$ for VAT.

1 Introduction

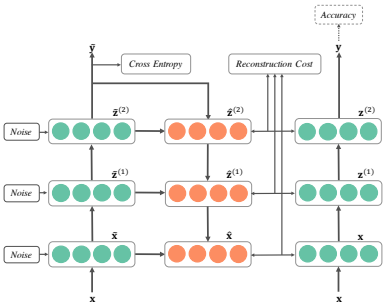
Ladder networks [15, 11] and virtual adversarial training [9] are two seemingly unrelated deep learning methods that have been successfully applied to semi-supervised learning (SSL). Below we present the view that both methods share statistical information between labeled and unlabeled examples by smoothing the probability distributions over their respective latent spaces. With this interpretation, we propose a new class of deep models for SSL that apply spatially-varying, anisotropic smoothing to latent spaces in the direction of greatest curvature of the unsupervised loss function.

In two models we investigate, we apply a virtual adversarial training cost in addition to ladder classification and denoising costs; in the other two models, we inject virtual adversarial noise into the encoder path. We train our models with 5, 10, or 100 labeled examples per class from the MNIST dataset, evaluating performance on both the standard test set and adversarial examples generated using the fast gradient method [3]. We found our models achieve state-of-the-art accuracy with high stability in the 5- or 10- labels per class setting on both normal and adversarial examples for MNIST.

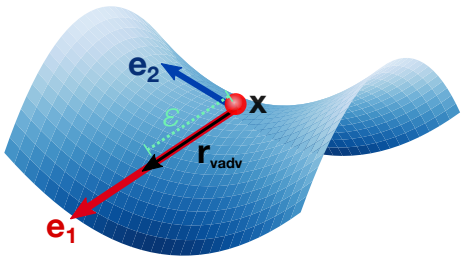
2 Background

In this section, we outline the ladder network [15, 11] and Virtual Adversarial Training (VAT) [9]. We present the ladder network as representing data in a hierarchy of nested latent spaces. SSL is performed by smoothing the labeled and unlabeled data distributions over this hierarchy, thus sharing distributional information between labeled and unlabeled distributions in a coarse-to-fine regime. In VAT, there is no latent space hierarchy, but instead a particularly clever choice of smoothing operator.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



(a) Illustration of a ladder network architecture.
Left: Noisy encoder with activations $\tilde{\mathbf{z}}^{(l)}$, additive Gaussian noise $\mathcal{N}(0, \sigma^2)$ at each layer, and outputs $\tilde{\mathbf{y}}$.
Centre: Decoder; input from layer above and corresponding layer in noisy encoder are combined by a denoising function $g^{(l)}(\cdot, \cdot)$ to form reconstructions $\hat{\mathbf{z}}^{(l)}$.
Right: Clean encoder, weights shared with noisy encoder; activations $\mathbf{z}^{(l)}$ are denoising targets.



(b) Conceptual illustration of virtual adversarial perturbation $\mathbf{r}_{\text{vadv}}$ on a surface representing the divergence $D\left[\Pr(y|\mathbf{x}, \boldsymbol{\theta}), \Pr(y|\mathbf{x} + \mathbf{r}, \boldsymbol{\theta})\right]$. $\mathbf{e}_1$, $\mathbf{e}_2$ indicate eigenvectors of the Hessian $\nabla\nabla_r D|_{\mathbf{r}=\mathbf{0}}$. $\mathbf{r}_{\text{vadv}}$ lies parallel to the dominant eigenvector $\mathbf{e}_1$ and has magnitude equal to the hyperparameter ϵ.

The Ladder Network For both supervised and unsupervised tasks, the ladder network [15, 11] uses a single autoencoder-like architecture with added skip connections from encoder to decoder. For labeled examples, the encoder is used as a feed-forward classifier, and for the unsupervised task, the full architecture is used as a denoising autoencoder, with extra reconstruction costs on intermediate representations (see Figure 1a). For the denoising autoencoder, additive spherical Gaussian noise is applied to encoder activations $\mathbf{z}^{(\ell)}$, which we interpret as applying isotropic smoothing to the hierarchy of latent spaces modelled by the ladder network. We denoted the smoothed activations as $\tilde{\mathbf{z}}^{(\ell)}$. Mathematically, we have

$$\Pr(\{\tilde{\mathbf{z}}^{(1)}, .., \tilde{\mathbf{z}}^{(L)}\}|\mathbf{x}) = \prod_{\ell=1}^{L} \int \mathcal{N}(\tilde{\mathbf{z}}^{(\ell)}; \mathbf{z}^{(\ell)}, \sigma^2_{\text{Ladder}} I) \delta(\mathbf{z}^{(\ell)} - \mathbf{y}^{(\ell-1)}) \, \mathrm{d}\tilde{\mathbf{z}}^{(\ell)} \tag{1}$$

$$\mathbf{y}^{(\ell)} = f_\ell(\mathbf{W}^{(\ell)} \tilde{\mathbf{z}}^{(\ell)} + \mathbf{b}^{(\ell-1)}) \tag{2}$$

where $\mathbf{y}^{(\ell-1)}$ is the output of the previous layer for $\ell > 0$ and $\mathbf{y}^{(0)} = \mathbf{x}$, f_ℓ is the nonlinearity, $\mathbf{W}^{(\ell)}$ is the weights, $\mathbf{b}^{(\ell)}$ is the bias. $\mathcal{N}(\tilde{\mathbf{z}}^{(\ell)}; \mathbf{z}^{(\ell)}, \sigma^2_{\text{Ladder}} I)$ is a normal distribution over injected noise given mean $\mathbf{z}^{(\ell)}$ and variance $\sigma^2_{\text{Ladder}}\mathbf{I}$, $\mathbf{I}$ being the identity matrix.

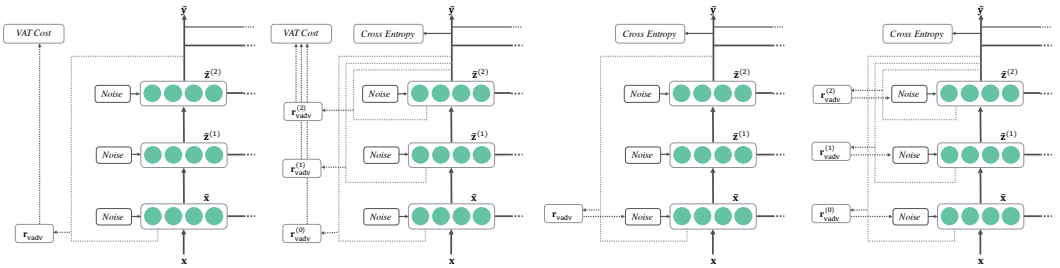
The ladder network architecture is illustrated in Figure 1a for a ladder network with $L = 2$ layers. The ladder network is trained to simultaneously minimize a negative log-likelihood on labeled examples and a denoising reconstruction cost at each layer on unlabeled examples. Specifically, the reconstruction cost is a squared error between the decoder activation $\hat{\mathbf{z}}^{(\ell)}$ and the noiseless encoder activation $\mathbf{z}^{(\ell)}$. Forward passes at training time through the model can be seen as Monte Carlo sampling from equation 1.

Virtual Adversarial Training Virtual adversarial perturbations (VAP) were first presented in [9] extending adversarial perturbations from [3] to the case where there are no labels. Adversarial perturbations are computed as

$$\mathbf{r}_{\text{adv}} := \underset{\mathbf{r};\|\mathbf{r}\| \leq \epsilon}{\arg\max} \, D\left[h(y), \Pr(y|\mathbf{x} + \mathbf{r}, \boldsymbol{\theta})\right], \tag{3}$$

where ϵ is a parameter dictating the size of the perturbation; $h(y)$ is the target distribution, *i.e.,* a one-hot vector of the true labels; $\Pr(y|\mathbf{x} + \mathbf{r}, \boldsymbol{\theta})$ is the output probabilities of the model with parameters $\boldsymbol{\theta}$; and $D[p, q]$ is a statistical divergence between P and Q (*i.e.,* a positive definite functional, only equal to zero when $P = Q$), such as the Kullback–Leibler (KL) divergence [7]. VAPs are generated by approximating $h(y)$ with $\Pr(y|\mathbf{x}, \boldsymbol{\theta})$. The perturbation is defined as

$$\mathbf{r}_{\text{vadv}} := \underset{\mathbf{r};\|\mathbf{r}\|_2 \leq \epsilon}{\arg\max} \, D\left[\Pr(y|\mathbf{x}, \boldsymbol{\theta}), \Pr(y|\mathbf{x} + \mathbf{r}, \boldsymbol{\theta})\right], \tag{4}$$



(a) LVAC (b) LVAC-LW (c) LVAN (d) LVAN-LW

Figure 2: Conceptual illustrations of our proposed models. All show corrupted encoder path only. Decoder and clean encoder (not shown) are identical to that of standard ladder in Figure 1a.

In practice, as further detailed in [8], the direction of $\mathbf{r}_{\text{vadv}}$ is approximated via second-order Taylor expansion as the dominant eigenvector of the Hessian matrix of the divergence D with respect to the perturbation $\mathbf{r}$; the eigenvector is computed with a finite difference power method [2]. At training time, the virtual adversarial perturbed examples $\mathbf{x} + \mathbf{r}_{\text{vadv}}$ are added to the training set and a virtual adversarial training loss is added to the current loss. The virtual adversarial training loss is defined as the average over all input data points of

$$L_{\text{vadv}}(\mathbf{x}, \boldsymbol{\theta}) := D\left[\Pr(y|\mathbf{x}, \boldsymbol{\theta}), \Pr(y|\mathbf{x} + \mathbf{r}_{\text{vadv}}, \boldsymbol{\theta})\right]. \tag{5}$$

Minimization of this term can be seen as a smoothing operation, since it penalizes differences in $\Pr(y|\mathbf{x}, \boldsymbol{\theta})$, in its direction of greatest curvature. The range of smoothing and hence strength of regularization is controlled by the parameter ϵ, the magnitude of the VAP. [8] found that fixing the coefficient of L_{vadv} relative to the supervised cost to 1 and tuning ϵ was sufficient to achieve good results, rather than tuning both.

3 Methods

Ladder with virtual adversarial costs (LVAC and LVAC-LW) One approach for applying the anisotropic smoothing in output space of VAT to the ladder network is to add a virtual adversarial cost term (as in Equation 5) to the supervised cross-entropy cost and unsupervised activation reconstruction cost which is optimized to train the ladder network. In the most general formulation of VAT on a ladder, the loss term can be written

$$C_{\text{vadv}} = \sum_l \alpha^{(l)} D_{KL}\left[\Pr(\tilde{y}|\tilde{\mathbf{z}}^{(l)}, \boldsymbol{\theta}), \Pr(\tilde{y}|\tilde{\mathbf{z}}^{(l)} + \mathbf{r}^{(l)}_{\text{vadv}}, \boldsymbol{\theta})\right], \tag{6}$$

$$\mathbf{r}^{(l)}_{\text{vadv}} = \underset{\mathbf{r};\, \|\mathbf{r}\|_2 \le \epsilon^{(l)}}{\arg\max}\, D_{KL}\left[\Pr(\tilde{y}|\tilde{\mathbf{z}}^{(l)}, \boldsymbol{\theta}), \Pr(\tilde{y}|\tilde{\mathbf{z}}^{(l)} + \mathbf{r}, \boldsymbol{\theta})\right] \tag{7}$$

where $\tilde{\mathbf{z}}^{(l)}$ is the activation in layer l of the corrupted encoder path, with $\tilde{\mathbf{z}}^{(0)} = \tilde{\mathbf{x}}$. This gives the *ladder with virtual adversarial costs, layer-wise (LVAC-LW)*, illustrated conceptually in 2b. Applying VAP to only the input images rather than the intermediate activations in the encoder gives us the *ladder with virtual adversarial cost (LVAC)*, illustrated in Figure 2a.

Ladder with virtual adversarial noise (LVAN and LVAN-LW) Alternatively, VAT smoothing can be applied to the ladder network by injecting virtual adversarial perturbations into the activations at each layer of the corrupted encoder in addition to isotropic Gaussian noise. VAP for each layer can be computed with the same form of Equation 7 for any layer l. As with the models proposed above, the addition of noise can be on the input images only, giving the *ladder with virtual adversarial noise (LVAN)* which is illustrated in Figure 2c. The alternative case of adding noise to each layer of the encoder, *ladder with virtual adversarial noise, layer-wise (LVAN-LW)*, is shown in Figure 2d.

4 Experiments

The structure of the classifier encoder for all of our models was a fully-connected network with layers of 1000, 500, 250, 250, 250, 10 units respectively. In all ladder implementations the decoder was

symmetric to the encoder. All models were trained for 250 epochs using the Adam optimizer [4] with initial learning rate 0.002 and linear learning rate decay from 200 epochs. All models were trained with unlabeled batch size 100; labeled batch size was 50 for training with 50 labeled examples and 100 otherwise. VAP's were generated with L_∞-norm. Hyperparameters were very roughly tuned using Bayesian optimization [13]; values used are given in Appendix A.

Low labeled data Table 1 shows average error rates on MNIST with 50, 100 and 1000 labeled examples for each of our proposed models and benchmark implementations of the ladder network and VAT. Mean and standard deviation for 50 labels is computed across ten training runs with different random seeds (fixed between models) for selecting labeled data and initializing weights; mean and standard deviations on 100 and 1000 labels are computed over five training runs. We expect high variability between training runs for very few labeled examples as performance depends significantly on the particular examples chosen. In this setting LVAN and LVAN-LW are notably highly stable in achieving very good performance.

Table 1: Average Error Rate (%) and standard error on MNIST

Model	**50 labels**		**100 labels**		**1000 labels**	
VAT (ours)	5.38	$\pm$ 2.92	2.14	$\pm$ 0.64	1.11	$\pm$ 0.05
Ladder (ours)	1.86	$\pm$ 0.43	1.45	$\pm$ 0.36	**1.10**	$\pm$ **0.05**
LVAC	2.42	$\pm$ 1.05	1.65	$\pm$ 0.12	1.28	$\pm$ 0.07
LVAC-LW	4.08	$\pm$ 3.55	1.39	$\pm$ 0.06	1.11	$\pm$ 0.12
LVAN	1.52	$\pm$ 0.20	1.30	$\pm$ 0.09	1.48	$\pm$ 0.03
LVAN-LW	**1.42**	$\pm$ **0.12**	**1.25**	$\pm$ **0.06**	1.51	$\pm$ 0.06

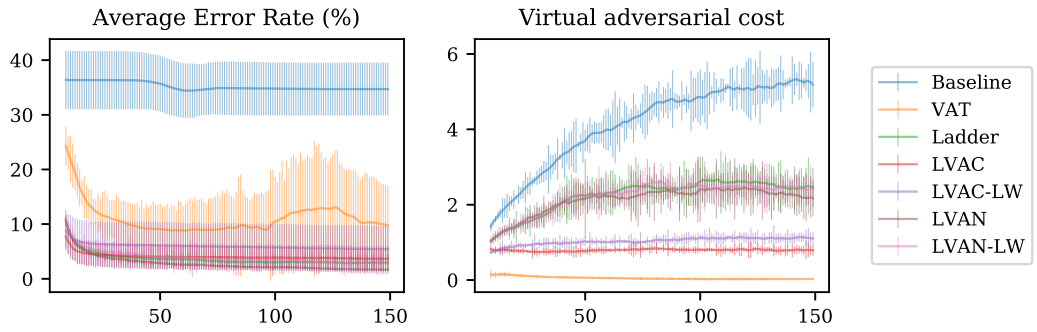
Adversarial examples We tested the performance of our models on adversarial examples generated using the CleverHans implementation of the fast gradient method with L_1, L_2 and L_∞ norms [3, 10]. For adversarial examples generated with L_∞ norm, VAT outperformed the ladder network and all of our proposed models with 50, 100 and 1000 labeled examples. VAT and all of our models showed improvement in AER as the number of labeled examples increased, while the ladder network did not appear to improve substantially. On L_1 and L_2 adversarial examples, LVAN-LW followed closely by LVAN outperformed all models including VAT and ladder for 50, 100 labels, while LVAC and LVAC-LW performed better than LVAN, LVAN-LW, VAT and ladder for 1000 labels. The ladder network performed very poorly on 50 and 100 labels, while VAT AER for 1000 labels appears to have been limited by the relatively high AER on non-adversarial examples. The relative strength of the LVAC models on L_∞ examples compared to L_2 or L_1 where the LVAN models performed better could be due to the relative strengths of regularization by VAT cost, which is based on L_∞ perturbations, and the L_2 denoising cost. Full results are presented in Table 2.

Table 2: Mean average error rate (%) and standard error on adversarial examples from MNIST

Norm	**Labels**	VAT	Ladder	LVAC	LVAC-LW	LVAN	LVAN-LW
$\boldsymbol{L_\infty}$	50	**22 $\pm$ 5.3**	54 $\pm$ 6	49 $\pm$ 2	34 $\pm$ 6	59 $\pm$ 5	56 $\pm$ 3
	100	**16 $\pm$ 1**	58 $\pm$ 2	48 $\pm$ 1	31 $\pm$ 2	60 $\pm$ 3	60 $\pm$ 2
	1000	**10.6 $\pm$ 0.4**	53 $\pm$ 2	36 $\pm$ 1	27 $\pm$ 3	40 $\pm$ 1	43 $\pm$ 2
$\boldsymbol{L_2}$	50	10 $\pm$ 8	26 $\pm$ 5	3 $\pm$ 2	5 $\pm$ 4	1.8 $\pm$ 0.4	**1.6 $\pm$ 0.2**
	100	3 $\pm$ 2	1.6 $\pm$ 0.4	1.7 $\pm$ 0.1	1.4 $\pm$ 0.1	1.4 $\pm$ 0.1	**1.3 $\pm$ 0.1**
	1000	2.4 $\pm$ 0.2	1.5 $\pm$ 0.1	1.4 $\pm$ 0.1	**1.2 $\pm$ 0.2**	1.6 $\pm$ 0.1	1.7 $\pm$ 0.1
$\boldsymbol{L_1}$	50	10 $\pm$ 8	69 $\pm$ 7	3 $\pm$ 2	4 $\pm$ 4	2.5 $\pm$ 0.4	**2.4 $\pm$ 0.3**
	100	4 $\pm$ 2	2.3 $\pm$ 0.4	2.2 $\pm$ 0.1	**1.9 $\pm$ 0.1**	2.0 $\pm$ 0.2	**1.9 $\pm$ 0.2**
	1000	2.6 $\pm$ 0.2	2.3 $\pm$ 0.2	**1.8 $\pm$ 0.2**	1.6 $\pm$ 0.1	2.2 $\pm$ 0.1	2.3 $\pm$ 0.2

Introspection: measuring smoothness The virtual adversarial loss given in Equation 5 is a measure of *lack of* local smoothness of the output distribution with respect to the input images. We expect that the ladder and LVAN models, though they do not explicitly minimize this cost, still perform

Figure 3: Comparison of average error rate and virtual adversarial cost (3 runs of 150 epochs each).



smoothing that should be reflected in this metric. This cost was computed with $\epsilon = 5.0$ for all models over 150 epochs of training (Figure 3). As expected, VAT, which directly minimizes this cost, is most smooth by this metric. The benchmark ladder network is significantly smoother than the fully supervised baseline despite not explicitly minimising the virtual adversarial cost. We measure LVAN and LVAC to be smoother than LVAN-LW and LVAC-LW respectively. This suggests smoothing with respect to the input image, which this metric measures, is traded off in the layer-wise models with smoothing in the intermediate latent spaces.

5 Discussion and Conclusions

In this work, we conducted an analysis of the ladder network from [11] and virtual adversarial training (VAT) from [9, 8] for semi-supervised learning and proposed four variants of a model applying virtual adversarial training to the ladder network: ladder with virtual adversarial cost (LVAC), ladder with layer-wise virtual adversarial cost (LVAC-LW), ladder with virtual adversarial noise (LVAN), and ladder with layer-wise virtual adversarial noise (LVAN-LW).

Based on the manifold and cluster assumptions of semi-supervised learning [1], we hypothesised that virtual adversarial training could improve the classification accuracy of the ladder network trained in a semi-supervised context. We tested this hypothesis on the MNIST dataset [6], by training models on training sets consisting of 50, 100 or 1000 labeled examples augmented by the full 60,000 images in the MNIST training set as unlabeled examples. We measured performance as error rate on the held-out test set of 10,000 examples.

We found that our models, most significantly LVAN-LW, improved on the performance of the ladder for 50 labels and 100 labels, achieving state-of-the-art error rates. For 1000 labels, both VAT and ladder baselines outperformed our models. This leads us to believe that the additional regularization provided by VAP's to the ladder network are useful only when the task is sufficiently challenging, suggesting that we should test our models on more complex datasets such as SVHN or CIFAR-10.

Additionally we found that our models performed better than the ladder network on adversarial examples. VAT outperformed our models for L_∞ adversarial examples, but our models, again especially the LVAN-LW model, achieved best performance for the few-label cases (50 and 100 labels) on L_1 and L_2-normalized adversarial examples.

Our best-performing models overall were based on adding virtual adversarial noise to the corrupted encoder path of the ladder (LVAN and LVAN-LW). These have additional advantages over the LVAC models proposed: they are faster, as they require fewer passes through the network, and produce more stable, consistent results.

A natural extension of our work would be to extend our interpretation to deep SSL methods which have been more recently introduced such as temporal ensembling [5], random data augmentation [12], and the Mean Teacher method [14].

Acknowledgements

This work was carried out by Saki Shinoda while at UCL and completed while at Prediction Machines Pte Ltd. Daniel Worrall is funded by Fight For Sight, UK.

References

[1] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. MIT Press, Cambridge, Massachusetts, 2006.

[2] G. H. Golub and H. A. van der Vorst. Eigenvalue computation in the 20th century. *J. Comput. Appl. Math.*, 123(1-2):35–65, Nov. 2000.

[3] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *arXiv preprint arXiv:1412.6572*, Dec. 2014. Presented at the 3rd International Conference on Learning Representations (San Diego, CA, USA, 7–9 May 2015).

[4] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, Dec. 2014. Presented at the 3rd International Conference on Learning Representations (San Diego, CA, USA, 7–9 May 2015).

[5] S. Laine and T. Aila. Temporal Ensembling for Semi-Supervised Learning. *ArXiv e-prints*, Oct. 2016. Presented at the 5th International Conference on Learning Representations (Toulon, FR, 24–26 April 2017).

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[7] D. J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[8] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual Adversarial Training: a Regularization Method for Supervised and Semi-supervised Learning. *arXiv preprint arXiv:1704.03976*, Apr. 2017.

[9] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional Smoothing with Virtual Adversarial Training. *arXiv preprint arXiv:1507.00677*, July 2015. Presented at the 4th International Conference on Learning Representations (San Juan, PR, USA, 2–4 May 2016).

[10] N. Papernot, I. Goodfellow, R. Sheatsley, R. Feinman, and P. McDaniel. cleverhans v1.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*, 2016.

[11] A. Rasmus, H. Valpola, M. Honkala, M. Berglund, and T. Raiko. Semi-supervised learning with ladder networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 3546–3554, Cambridge, MA, USA, 2015. MIT Press.

[12] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171, 2016.

[13] scikit-optimize contributors. Scikit-Optimize: Sequential model-based optimization with a 'scipy.optimize' interface, 2017. Software used under BSD license.

[14] A. Tarvainen and H. Valpola. Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*, 2017. Accepted as a conference paper at 2017 conference on Neural Information Processing Systems.

[15] H. Valpola. From neural PCA to deep unsupervised learning. *arXiv preprint arXiv:1411.7783*, Nov. 2014.

A Hyperparameters

Model	**Labels**	$\lambda^{(0)}$	$\lambda^{(1)}$	$\lambda^{(\geq 2)}$	$\epsilon^{(0)}$	$\epsilon^{(1)}$	$\epsilon^{(\geq 2)}$
LVAC	50	1504	16.15	0.0381	0.0733	-	-
	100	1966	14.20	0.1563	0.0731	-	-
	1000	3883	12.35	0.0539	2.5206	-	-
LVAC-LW	50	1000	10.00	0.1000	1.0000	0.1000	1.00×10^{-3}
	100	1966	14.20	0.1563	0.0731	0.4822	1.402×10^{-3}
	1000	3883	12.35	0.0539	2.5206	0.0143	6.002×10^{-4}
LVAN	50	1504	16.15	0.0381	0.0733	-	-
	100	1966	14.20	0.1563	0.0731	-	-
	1000	3883	12.35	0.0539	2.5206	-	-
LVAN-LW	50	1504	16.15	0.0381	0.0733	0.3897	8.372×10^{-2}
	100	1966	14.20	0.1563	0.0731	0.4822	1.402×10^{-3}
	1000	3883	12.35	0.0539	2.5206	0.0143	6.002×10^{-4}
Ladder	50	1504	16.15	0.0381	-	-	-
	100	1966	14.20	0.1563	-	-	-
	1000	3883	12.35	0.0539	-	-	-
VAT	50	-	-	-	5.0	-	-
	100	-	-	-	5.0	-	-
	1000	-	-	-	2.5	-	-

Learning to Model the Tail

Yu-Xiong Wang **Deva Ramanan** **Martial Hebert**
Robotics Institute, Carnegie Mellon University
{yuxiongw, dramanan, hebert}@cs.cmu.edu

Abstract

We describe an approach to learning from long-tailed, imbalanced datasets that are prevalent in real-world settings. Here, the challenge is to learn accurate "few-shot" models for classes in the tail of the class distribution, for which little data is available. We cast this problem as transfer learning, where knowledge from the data-rich classes in the head of the distribution is transferred to the data-poor classes in the tail. Our key insights are as follows. First, we propose to transfer *meta*-knowledge about *learning-to-learn* from the head classes. This knowledge is encoded with a meta-network that operates on the space of model parameters, that is trained to predict many-shot model parameters from few-shot model parameters. Second, we transfer this meta-knowledge in a *progressive* manner, from classes in the head to the "body", and from the "body" to the tail. That is, we transfer knowledge in a gradual fashion, regularizing meta-networks for few-shot regression with those trained with more training data. This allows our final network to capture a notion of *model dynamics*, that predicts how model parameters are likely to change as more training data is gradually added. We demonstrate results on image classification datasets (SUN, Places, and ImageNet) tuned for the long-tailed setting, that significantly outperform common heuristics, such as data resampling or reweighting.

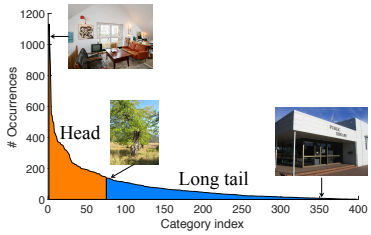
1 Motivation

Deep convolutional neural networks (CNNs) have revolutionized the landscape of visual recognition, through the ability to learn "big models" with hundreds of millions of parameters [1, 2, 3, 4]. Such models are typically learned with *artificially balanced* datasets [5, 6, 7], in which objects of different classes have approximately evenly distributed, very large number of human-annotated images. In real-world applications, however, visual phenomena follow a long-tailed distribution as shown in Fig. 1, in which the number of training examples per class varies significantly from hundreds or thousands for head classes to as few as one for tail classes [8, 9, 10].

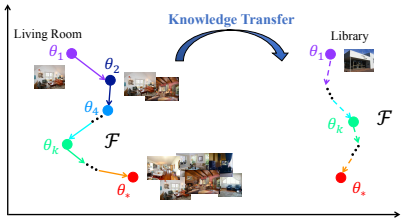
Long-tail: Minimizing the skewed distribution by collecting more tail examples is a notoriously difficult task when constructing datasets [11, 6, 12, 10]. Even those datasets that are balanced along one dimension still tend to be imbalanced in others [13]; *e.g.*, balanced scene datasets still contain long-tail sets of objects [14] or scene subclasses [8]. This *intrinsic* long-tail property poses a multitude of open challenges for recognition in the wild [15], since the models will be largely dominated by those few head classes while degraded for many other tail classes. Rebalancing training data [16, 17] is the most widespread state-of-the-art solution, but this is *heuristic and suboptimal* — it merely generates redundant data through over-sampling or loses critical information through under-sampling.

Head-to-tail knowledge transfer: An attractive alternative is to *transfer* knowledge from data-rich head classes to data-poor tail classes. While transfer learning [18, 19, 20] from a source to target task is a well studied problem [18, 21], by far the most common approach is fine-tuning a model pre-trained on the source task [22]. In the long-tailed setting, this fails to provide any noticeable improvement since pre-training on the head is quite similar to training on the unbalanced long-tailed dataset (which is dominated by the head) [10].

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.



(a) Long-tail distribution on the SUN-397 dataset.



(b) Knowledge transfer from head to tail classes.

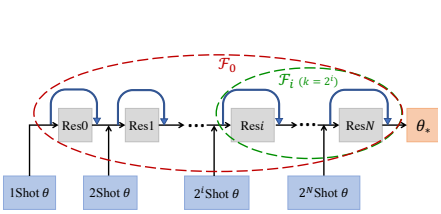
Figure 1: Head-to-tail knowledge transfer in model space for long-tail recognition. Fig. 1a shows the number of examples by scene class on SUN-397 [14], a representative dataset that follows an intrinsic long-tailed distribution. In Fig. 1b, from the data-rich head classes (*e.g.*, living rooms), we introduce a meta-learner $\mathcal{F}$ to learn the model dynamics — a series of transformations (denoted as solid lines) that represents how few k-shot models θ_k start from θ_1 and gradually evolve to the underlying many-shot models θ_* trained from large sets of samples. The model parameters θ are visualized as points in the "dual" model (parameter) space. We leverage the model dynamics as prior knowledge to facilitate recognizing tail classes (*e.g.*, libraries) by hallucinating their model evolution trajectories (denoted as dashed lines).

Transferring meta-knowledge: Inspired by the recent work on meta-learning [23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37], we instead transfer meta-level knowledge about *learning to learn* from the head classes. Specifically, we make use of the approach of [23], which describes a method for learning from small datasets (the "few-shot" learning problem [20, 38, 39, 40, 41, 42, 43, 44, 23, 45, 24, 46, 47, 48, 41, 49, 50, 51, 52, 53, 54]) through estimating a generic model transformation. To do so, [23] learns a meta-level network that operates on the space of model parameters, which is specifically trained to *regress* many-shot model parameters (trained on large datasets) from few-shot model parameters (trained on small datasets). Our meta-level regressor, which we call *MetaModelNet*, is trained on classes from the head and then applied to those from the tail. As an illustrative example in Fig. 1, consider learning scene classifiers on a long-tailed dataset with many living-rooms but few outside libraries. We learn both many-shot and few-shot living-room models (by subsampling the training data as needed), and train a regressor that maps between the two. We can then apply the regressor on few-shot models of libraries learned from the tail.

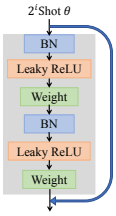
Progressive transfer: The above description suggests that we need to split up a long-tailed training set into a distinct set of source classes (the head) and target classes (the tail). This is most naturally done by thresholding the number of training examples per class. But what is the correct threshold? A high threshold might result in a meta-network that simply acts as an identity function, returning the input set of model parameters. This certainly would not be useful to apply on few-shot models. Similarly, a low threshold may not be useful when regressing from many-shot models. Instead, we propose a "continuous" strategy that builds multiple regressors across a (logarithmic) *range* of thresholds (*e.g.*, 1-shot, 2-shot, 4-shot regressors, *etc.*), corresponding to different head-tail splits. Importantly, these regressors can be efficiently implemented with a *single, chained* MetaModelNet that is naturally regularized with residual connections, such that the 2-shot regressor need only predict model parameters that are fed into the 4-shot regressor, and so on (until the many-shot regressor that defaults to the identity). By doing so, MetaModelNet encodes a trajectory over the space of model parameters that captures their evolution with increasing sample sizes, as shown in Fig. 1b. Interestingly, such a network is naturally trained in a *progressive* manner from the head towards the tail, effectively capturing the gradual dynamics of transferring meta-knowledge from data-rich to data-poor regimes.

Model dynamics: It is natural to ask what kind of dynamics are learned by MetaModelNet. How can one consistently predict how model parameters will change with more training data? We posit that the network learns to capture implicit *data augmentation*. For example, given a 1-shot model trained with a single image, the network may learn to implicitly add rotations of that single image. But rather than explicitly creating data, MetaModelNet predicts their impact on the learned model parameters.

Our contributions are three-fold. (1) We analyze the *dynamics* of how model parameters evolve when given access to more training examples. (2) We show that a single meta-network, based on deep residual learning, can learn to accurately predict such dynamics. (3) We train such a meta-network on long-tailed datasets through a recursive approach that gradually transfers meta-knowledge learned from the head to the tail, significantly improving long-tail recognition on a broad range of tasks.



(a) Learning a sample size dependent transformation.



(b) Structure of residual blocks.

Figure 2: MetaModelNet architecture for learning model dynamics. We instantiate MetaModelNet as a deep residual network with residual blocks $i = 0,\ 1, \ldots, N$ in Fig. 2a, which accepts few-shot model parameters θ (trained on small datasets across a logarithmic range of sample sizes k, $k = 2^i$) as (multiple) inputs and regresses them to many-shot model parameters θ_* (trained on large datasets) as output. The skip connections ensure the identity regularization. $\mathcal{F}_i$ denotes the meta-learner that transforms (regresses) k-shot θ to θ_*. Fig. 2b shows the structure of the residual blocks. Note that the meta-learners $\mathcal{F}_i$ for different k are derived from this single, chained meta-network, with nested circles (subnetworks) corresponding to $\mathcal{F}_i$.

2 Head-to-tail meta-knowledge transfer

Given a long-tail recognition task of interest and a base recognition model such as a deep CNN, our goal is to transfer knowledge from the data-rich head to the data-poor tail classes. As shown in Fig. 1, knowledge is represented as trajectories in model space that capture the evolution of parameters with more and more training examples. We train a meta-learner (MetaModelNet) to learn such model dynamics from the head classes, and then "hallucinate" the evolution of parameters for the tail classes. To simplify exposition, we first describe the approach for a fixed split of our training dataset into a head and tail. We then generalize the approach to multiple splits.

Fixed-size model transformations: Let us write H_t for the "head" training set of (x, y) data-label pairs constructed by assembling those classes for which there exists more than t training examples. We will use H_t to learn a meta-network thats maps few-shot model parameters to many-shot parameters, and then apply this network on few-shot models from the tail classes. To do so, we closely follow the model regression framework from [23], but introduce notation that will be useful later. Let us write a base learner as $g(x; \theta)$ as a feedforward function $g(\cdot)$ that processes an input sample x given parameters θ. We first learn a set of "optimal" model parameters θ_* by tuning g on H_t with a standard loss function. We also learn few-shot models by randomly sampling a smaller fixed number of examples per class from H_t. We then train a meta-network $\mathcal{F}(\cdot)$ to regress few-shot parameters to θ_*, where $\mathcal{F}(\cdot)$ is itself parameterized with weights w. We focus on parameters from the last fully-connected layer for a single class — *e.g.*, $\theta \in \mathbb{R}^{4096}$ for an AlexNet architecture. This allows us to learn regressors that are shared across classes (as in [23]), and so can be applied to any individual test class. The objective function for each class is:

$$\sum_{\theta \in k\text{Shot}(H_t)} \Big\{ ||\mathcal{F}(\theta; w) - \theta_*||^2 + \lambda \sum_{(x,y) \in H_t} \text{loss}\Big(g\big(x; \mathcal{F}(\theta; w)\big), y\Big) \Big\}. \tag{1}$$

The final loss is averaged over all the head classes and minimized with respect to w. Here, $k\text{shot}(H_t)$ is the set of few-shot models learned by subsampling H_t, and loss refers to the performance loss used to train the base network (*e.g.*, cross-entropy). [23] found that the performance loss was useful to learn regressors that maintained high accuracy on the base task.

Training: What should be the value of k, for the k-shot models being trained? One might be tempted to set $k = t$, but this implies that there will be some head classes near the cutoff that have only t training examples, implying θ and θ_* will be identical. To ensure that a meaningful mapping is learned, we set $k = t/2$.

Recursive residual transformations: We wish to apply the above module on all possible head-tail splits of a long-tailed training set. To do so, we extend the above approach in three crucial ways:

- (Sample-size dependency) Generate a sequence of different meta-learners $\mathcal{F}_i$ each tuned for a specific k, where $k = k(i)$ is an increasing function of i (that will be specified shortly). Through a straightforward extension, prior work on model regression [23] learns a single fixed meta-learner for all the k-shot regression tasks.

- (Identity regularization) Ensure that the meta-learner defaults to the identity function for large i: $\mathcal{F}_i \to \mathcal{I}$ as $i \to \infty$.
- (Compositionality) Compose meta-learners out of each other: $\forall i < j$, $\mathcal{F}_i(\theta) = \mathcal{F}_j\Big(\mathcal{F}_{ij}(\theta)\Big)$ where $\mathcal{F}_{ij}$ is the regressor that maps between $k(i)$-shot and $k(j)$-shot models.

Here we dropped the explicit dependence of $\mathcal{F}(\cdot)$ on w for notational simplicity. These observations emphasize the importance of (1) the identity regularization and (2) sample-size-dependent regressors for long-tailed model transfer. We operationalize these extensions with a recursive residual network:

$$\mathcal{F}_i(\theta) = \mathcal{F}_{i+1}\Big(\theta + f(\theta; w_i)\Big), \tag{2}$$

where f denotes a residual block parameterized by w_i and visualized in Fig. 2b. Inspired by [55, 23], f consists of batch normalization (BN) and leaky ReLU as pre-activation, followed by fully-connected weights. By construction, each residual block transforms an input $k(i)$-shot model to a $k(i+1)$-shot model. The final MetaModelNet can be efficiently implemented through a chained network of $N+1$ residual blocks, as shown in Fig. 2a. By feeding in a few-shot model at a particular block, we can derive any meta-learner $\mathcal{F}_i$ from the central underlying chain.

Training: Given the network structure defined above, we now describe an efficient method for training based on two insights. (1) The recursive definition of MetaModelNet suggests a recursive strategy for training. We begin with the *last* block and train it with the *largest* threshold (*e.g.*, those few classes in the head with many examples). The associated k-shot regressor should be easy to learn because it is similar to an identity mapping. Given the learned parameters for the last block, we then train the next-to-last block, and so on. (2) Inspired by the general observation that recognition performance improves *on a logarithmic scale* as the number of training samples increases [8, 9, 56], we discretize blocks accordingly, to be tuned for 1-shot, 2-shot, 4-shot, ... recognition. In terms of notation, we write the recursive training procedure as follows. We iterate over blocks i from N to 0, and for each i:

- Using Eqn. (1), train parameters of the residual block w_i on the head split H_t with k-shot model regression, where $k = 2^i$ and $t = 2k = 2^{i+1}$.

The above "back-to-front" training procedure works because whenever block i is trained, all subsequent blocks $(i+1, \ldots, N)$ have already been trained. In practice, rather than holding all subsequent blocks fixed, it is natural to fine-tune them while training block i. One approach might be fine-tuning them on the current $k = 2^i$-shot regression task being considered at iteration i. But because MetaModelNet will be applied across a wide range of k, we fine-tune blocks in a *multi-task* manner across the current viable range of $k = (2^i, 2^{i+1}, \ldots, 2^N)$ at each iteration i.

3 Experimental Evaluation

Evaluation and analysis on SUN-397: We first focus on fine-tuning the classifier module while freezing the representation module of a pre-trained ResNet152 CNN model [4] on long-tailed SUN-397 [14, 57] for its state-of-the-art performance. In addition to the "plain" baseline that fine-tunes on the target data following the standard practice, we compare against three state-of-the-art baselines that are widely used to address the imbalanced distributions. (1) Over-sampling [16, 17], which uses the balanced sampling via label shuffling as in [16, 17]. (2) Under-sampling [58], which reduces the number of samples per class to 47 at most (the median value). (3) Cost-sensitive [59], which introduces additional weights in the loss function for each class with inverse class frequency.

Table 1 summarizes the performance comparison averaged over all classes and Fig. 3 details the per class comparison. Table 1 shows that our MetaModelNet provides a promising way of encoding the shared structure across classes *in model space*. It outperforms existing approaches by a large margin. Fig. 3 shows that our approach significantly improves accuracy in the tail.

Ablation analysis: Table 2 shows that training for a fixed sample size and identity regularization provide a noticeable performance boost (2%). Adding multiple head-tail splits through recursion further improves accuracy by a small but noticeable amount (0.5% as shown in Table 2). Table 3 shows that progressively learning classifier dynamics while fine-tuning features performs the best when using ResNet50 [4].

Understanding model dynamics: Fig. 4 shows some empirical analysis of model dynamics.

Method	Plain [4]	Over-Sampling [16, 17]	Under-Sampling [58]	Cost-Sensitive [59]	MetaModelNet (Ours)
Acc (%)	48.03	52.61	51.72	52.37	**57.34**

Table 1: Performance comparison between our MetaModelNet and state-of-the-art approaches for long-tailed scene classification when fine-tuning the pre-trained ILSVRC ResNet152 on the SUN-397 dataset. We focus on learning the model dynamics of the classifier module while freezing the CNN representation module. By benefiting from the learned generic model dynamics from the head classes, ours significantly outperforms all the baselines for the long-tail recognition.

Method	Model Regression [23]	MetaModelNet+Fix Split (Ours)	MetaModelNet+ Recur Split (Ours)
Acc (%)	54.68	56.86	**57.34**

Table 2: Ablation analysis of variations of our MetaModelNet.

Scenario	Pre-Trained Features		Fine-Tuned Features (FT)		
Method	Plain [4]	MetaModelNet (Ours)	Plain [4]	Fix FT + MetaModelNet (Ours)	Recur FT + MetaModelNet (Ours)
Acc (%)	46.90	54.99	49.40	58.53	**58.74**

Table 3: Ablation analysis of joint feature fine-tuning and model dynamics learning.

Dataset	Places-205 [7]		ILSVRC-2012 [5]	
Method	Plain [1]	MetaModelNet (Ours)	Plain [1]	MetaModelNet (Ours)
Acc (%)	23.53	**30.71**	68.85	**73.46**

Table 4: Performance comparisons on large-scale scene-centric Places [7] and object-centric ImageNet [5] datasets, which are tuned for the long-tailed setting.

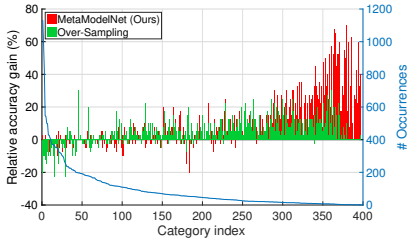


Figure 3: Detailed per class performance comparison between our MetaModelNet and the state-of-the-art over-sampling approach for long-tailed scene classification on the SUN-397 dataset. X-axis: class index. Y-axis (Left): per class classification accuracy improvement relative to the plain baseline. Y-axis (Right): number of training examples. Ours significantly improves for the few-shot tail classes.

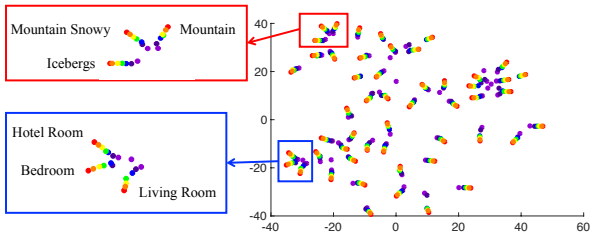


Figure 4: Visualizing model dynamics. We visualize models as points in the "dual" model (parameter) space and examine the evolution of parameters predicted by MetaModelNet with t-SNE [60]. 1-shot models (purple) to many-shot models (red) are plotted in a rainbow order. These visualizations show that MetaModelNet learns an approximately-smooth, nonlinear warping of this space that transforms (few-shot) input points to (many-shot) output points. Similar semantic classes tend to be close and transform in similar ways.

Generalization to other tasks and datasets: Table 4 shows the generality of our approach and shows that the MetaModelNets facilitate the recognition of other long-tailed datasets with significantly different visual concepts and distributions.

4 Conclusions

In this work we proposed a conceptually simple but powerful approach to address the problem of long-tail recognition through knowledge transfer from the head to the tail of the class distribution. Our key insight is to represent the model dynamics through meta-learning, *i.e.,* how a recognition model transforms and evolves during the learning process when gradually encountering more training examples. To do so, we introduce a meta-network that learns to progressively transfer meta-knowledge from the head to the tail classes. We present several state-of-the-art results on benchmark datasets (SUN, Places, and ImageNet) tuned for the long-tailed setting, that significantly outperform common heuristics, such as data resampling or reweighting.

Acknowledgments. We thank Liangyan Gui and Olga Russakovsky for valuable and insightful discussions. This work was supported in part by ONR MURI N000141612007 and U.S. Army Research Laboratory (ARL) under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016. DR was supported in part by the National Science Foundation (NSF) under grant number IIS-1618903, Google, and Facebook. We also thank NVIDIA for donating GPUs and AWS Cloud Credits for Research program.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.
[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.
[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
[5] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.
[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014.
[7] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017.
[8] X. Zhu, D. Anguelov, and D. Ramanan. Capturing long-tail distributions of object subcategories. In *CVPR*, 2014.
[9] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan. Do we need more training data? *IJCV*, 119(1):76–92, 2016.
[10] G. Van Horn and P. Perona. The devil is in the tails: Fine-grained classification in the wild. *arXiv preprint arXiv:1709.01450*, 2017.
[11] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.
[12] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalanditis, L.-J. Li, D. A. Shamma, M. Bernstein, and L. Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 123(1):32–73, 2017.
[13] W. Ouyang, X. Wang, C. Zhang, and X. Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *CVPR*, 2016.
[14] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva. SUN database: Exploring a large collection of scene categories. *IJCV*, 119(1):3–22, 2016.
[15] S. Bengio. Sharing representations for long tail computer vision problems. In *ICMI*, 2015.
[16] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016.
[17] Q. Zhong, C. Li, Y. Zhang, H. Sun, S. Yang, D. Xie, and S. Pu. Towards good practices for recognition & detection. In *CVPR workshops*, 2016.
[18] S. J. Pan and Q. Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2010.
[19] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.
[20] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap. One-shot learning with memory-augmented neural networks. In *ICML*, 2016.
[21] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.
[22] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
[23] Y.-X. Wang and M. Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, 2016.
[24] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi. Learning feed-forward one-shot learners. In *NIPS*, 2016.
[25] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.
[26] K. Li and J. Malik. Learning to optimize. In *ICLR*, 2017.
[27] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.
[28] S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012.
[29] J. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28(1):105–130, 1997.
[30] A. Sinha, M. Sarkar, A. Mukherjee, and B. Krishnamurthy. Introspection: Accelerating neural network training by learning weight evolution. In *ICLR*, 2017.
[31] J. Schmidhuber. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta-... hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

[32] J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.
[33] J. Schmidhuber. A neural network that embeds its own meta-levels. In *IEEE International Conference on Neural Networks*, 1993.
[34] D. Ha, A. Dai, and Q. V. Le. Hypernetworks. In *ICLR*, 2017.
[35] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
[36] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *NIPS*, 2017.
[37] T. Munkhdalai and H. Yu. Meta networks. In *ICML*, 2017.
[38] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *TPAMI*, 28(4):594–611, 2006.
[39] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013.
[40] Y.-X. Wang and M. Hebert. Model recommendation: Generating object detectors from few samples. In *CVPR*, 2015.
[41] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Workshops*, 2015.
[42] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.
[43] J. Ba, K. Swersky, S. Fidler, and R. Salakhutdinov. Predicting deep zero-shot convolutional neural networks using textual descriptions. In *ICCV*, 2015.
[44] Y.-X. Wang and M. Hebert. Learning by transferring from unsupervised universal sources. In *AAAI*, 2016.
[45] Z. Li and D. Hoiem. Learning without forgetting. In *ECCV*, 2016.
[46] B. Hariharan and R. Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, 2017.
[47] Y.-X. Wang and M. Hebert. Learning from small sample sets by combining unsupervised meta-training with CNNs. In *NIPS*, 2016.
[48] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):669–688, 1993.
[49] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.
[50] H. Noh, P. H. Seo, and B. Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *CVPR*, 2016.
[51] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.
[52] Y. Fu, T. Xiang, Y.-G. Jiang, X. Xue, L. Sigal, and S. Gong. Recent advances in zero-shot recognition. *IEEE Signal Processing Magazine*, 2017.
[53] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, and D. S. Phoenix. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 2017.
[54] E. Triantafillou, R. Zemel, and R. Urtasun. Few-shot learning through an information retrieval lens. In *NIPS*, 2017.
[55] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.
[56] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.
[57] Y.-X. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *CVPR*, 2017.
[58] H. He and E. A. Garcia. Learning from imbalanced data. *TKDE*, 21(9):1263–1284, 2009.
[59] C. Huang, Y. Li, C. C. Loy, and X. Tang. Learning deep representation for imbalanced classification. In *CVPR*, 2016.
[60] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *JMLR*, 9(Nov):2579–2605, 2008.

Neural Simpletrons – Learning in the Limit of Few Labels with Directed Generative Networks

Dennis Forster
Machine Learning Group
University of Oldenburg
26129 Oldenburg, Germany
FIAS, Goethe-University
60438 Frankfurt, Germany
dennis.forster@uol.de

Abdul-Saboor Sheikh
Zalando Research
11501 Berlin, Germany
Cluster of Excellence H4a
University of Oldenburg
26129 Oldenburg, Germany
sheikh.abdulsaboor@gmail.com

Jörg Lücke
Machine Learning Group
Cluster of Excellence H4a &
RC Neurosensory Sciences
University of Oldenburg
26129 Oldenburg, Germany
joerg.luecke@uol.de

Abstract

We investigate the task of classifier training for data sets with limited labels using a neural network derived from a hierarchical normalized Poisson mixture model. With the single objective of likelihood optimization, both labeled and unlabeled data are naturally incorporated into learning. Using standard benchmarks for non-negative data, such as text document representations (20 Newsgroups), MNIST and NIST SD19, we study the classification performance when only very few labels are used for training and parameter tuning. In different settings, the network's performance is compared to standard and recently suggested semi-supervised classifiers. While other recent approaches are more competitive for many labels or fully labeled data sets, we find that the here studied network can be applied to limits of few labels where no other system has been reported to operate so far.

1 Introduction

Large data sets, e.g., in the form of digital texts, images or sounds, become increasingly ubiquitous. However, acquisition of fully labeled data becomes increasingly costly with larger amounts of data points, as correct labeling requires ground-truth or a human who can hand-label the data. Consequently, classifiers leveraging information from both labeled and unlabeled data have in recent years shifted into the focus of many research groups (Liu et al., 2010; Weston et al., 2012; Pitelis et al., 2014; Kingma et al., 2014; Forster et al., 2015; Rasmus et al., 2015; Miyato et al., 2015). Typically, a supervised deep neural network (DNN) is used in combination with additional mechanisms that allow usability when labels are limited. However, such systems generally come with large numbers of free parameters, which in the semi-supervised setting increases the risk of overfitting to very small validation sets. Alternatively, standard probabilistic networks, e.g., in the form of deep directed graphical models (DDMs) can in principle be trained using unlabeled and labeled data. However, while being potentially very powerful information processors, typical directed models are limited in size (compare, e.g. Larochelle and Murray, 2011; Bornschein and Bengio, 2015; Gan et al., 2015).

In contrast to DNNs and SVMs, DDMs are primarily used for unsupervised learning. For the targeted limit of few labels, DDMs thus appear as a more natural starting point if we are able to address scalability for classification applications. In order to do so, we base our study on a directed graphical model which is sufficiently richly structured to give rise to a good classifier, while it allows for efficient training on large data sets and with large network sizes. Scalability will be realized by the derivation of a neural network equivalent for maximum likelihood learning of the considered graphical model. The emerging compact and local inference and learning equations of the network can then be parallelized and scaled using the same tools as were originally developed for conventional deep neural networks. By additionally considering a minimalistic network architecture, the number of free parameters will, at the same time, be kept low and easily tunable on few labels.

LLD Workshop, 31st Conf. on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

2 A Hierarchical Neural Network for Learning with Limited Labels

A classification problem can be modeled as an inference task based on a probabilistic hierarchical mixture model (e.g., Duda et al., 2001). For our goal of semi-supervised learning with limited labels, we restrain the model complexity to the minimum of no more than three layers:

$$p(k) = {}^1\!/_K, \qquad p(l|k) = \delta_{lk} \tag{1}$$

$$p(c|k,\mathcal{R}) = \mathcal{R}_{kc}, \qquad \textstyle\sum_c \mathcal{R}_{kc} = 1 \tag{2}$$

$$p(\vec{y}\,|c,\mathcal{W}) = \textstyle\prod_d \mathrm{Poisson}(y_d; \mathcal{W}_{cd}), \qquad \textstyle\sum_d \mathcal{W}_{cd} = A \tag{3}$$

The parameters of the model, $\mathcal{W} \in \mathbb{R}_{>0}^{C\times D}$ and $\mathcal{R} \in \mathbb{R}_{\geq 0}^{K\times C}$, will be referred to as generative weights, which are normalized to constants A and 1, respectively. The top node represents K abstract concepts or super classes k (for example, ten classes of digits). The middle node represents any of the occurring C subclasses c (like different writing styles of the digits). And the bottom nodes represent an observed data sample $\vec{y}$ with an according data label l (e.g., ranging from '0' to '9'). Our model assumes non-negative observed data, and we use the Poisson distribution as an elementary distribution for non-negative observations.

For the purposes of this study, we specify a neural network formulation that corresponds to learning and inference in this hierarchical generative model. Consider the neural network in Fig. 1 with neural activities $\vec{y}$, $\vec{s}$ and $\vec{t}$ and learning rules as in Tab. 1. We assume the values of $\vec{y}$ to be obtained from a set of unnormalized data points $\tilde{\vec{y}}$ by Eq. (T1.3), and the label information to be presented as top-down input vector $\vec{u}$ as given in Eq. (T1.2). For the neural weights (W, R) of the network, we consider Hebbian learning with a subtractive synaptic scaling term (see for example Abbott and Nelson, 2000) as in Eqs. (T1.7) and (T1.8), where $\epsilon_W > 0$ and $\epsilon_R > 0$ are learning rates. These learning rules are local, can integrate both supervised and unsupervised learning, are highly parallelizable and they result in normalized neural weights (W, R). Those we can relate to the generative weights $(\mathcal{W}, \mathcal{R})$ by identifying s_c with the posterior probability $p(c|\vec{y}^{\,(n)}, l^{(n)}, \Theta)$ by Eq. (T1.4) and t_k with $p(k|\vec{y}^{\,(n)}, l^{(n)}, \Theta)$ by Eq. (T1.6), using the neural weights as model parameters Θ. Such neural learning converges to the same fixed points as EM for the hierarchical Poisson mixture model (compare Lücke and Sahani, 2008; Keck et al., 2012; Forster et al., 2015; Forster and Lücke, 2017). In other words, executing the online neural network of Tab. 1 optimizes the likelihood of the generative model Eqs. (1) to (3). The network's neural activities therein provide the posterior probabilities, which we use for classification with the MAP estimate of t_k (Eq. 1.6) giving the inferred classes. The computation of posteriors is in general a difficult and computationally intensive endeavor, and their interpretation as neural activation rules is usually difficult. In our case, because of a specific interplay between introduced constraints, categorical distribution and Poisson noise, the posteriors and their neural interpretability however greatly simplify. The equations defining the neural network are elementary, very compact, and contain a total number of only four free parameters: the number of hidden units C, an input normalization constant A, and learning rates ϵ_W and ϵ_R. Because of its compactness, we call the network *Neural Simpletron* (NeSi).

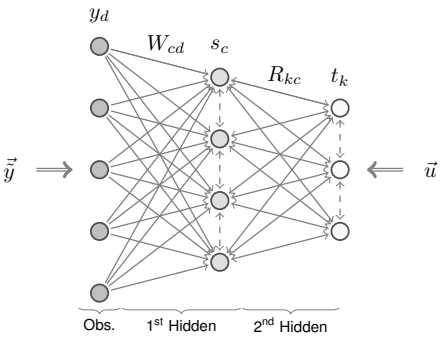


Figure 1: Graphical illustration of the hierarchical recurrent neural network.

Table 1: Neural network formulation of probabilistic inference and maximum likelihood learning.

Neural Simpletron

Input

Bottom-Up: $\tilde{y}_d$ unnormalized data (T1.1)

Top-Down: $u_k = \begin{cases} \delta_{kl} & \text{for labeled data} \\ \frac{1}{K} & \text{for unlabeled data} \end{cases}$ (T1.2)

Activation Across Layers

Obs. Layer: $y_d = (A - D)\frac{\tilde{y}_d}{\sum_{d'} \tilde{y}_{d'}} + 1$ (T1.3)

1st Hidden: $s_c = \frac{\exp(I_c)}{\sum_{c'} \exp(I_{c'})}$, with (T1.4)

$I_c = \sum_d \log(W_{cd}) y_d + \log(\sum_k u_k R_{kc})$ (T1.5)

2nd Hidden: $t_k = \begin{cases} u_k & \text{labeled data} \\ \sum_c \frac{R_{kc}}{\sum_{k'} R_{k'c}} s_c & \text{unlabeled data} \end{cases}$ (T1.6)

Learning of Neural Weights

1st Hidden: $\Delta W_{cd} = \epsilon_W (s_c y_d - s_c W_{cd})$ (T1.7)

2nd Hidden: $\Delta R_{kc} = \epsilon_R (t_k s_c - t_k R_{kc})$ labeled data (T1.8)

2.1 Simpletron Variants

We investigate different NeSi versions for learning with limited labels. All variants accord with the likelihood objective and were chosen such that the number of tunable parameters remains small: (I) The complete formulas for the first hidden layer, given in Eqs. (T1.4) and (T1.5), define a recurrent network ('r-NeSi'), i.e., combine both bottom-up and top-down information. By removing the second term in Eq. (T1.5), we gain a feedforward network ('ff-NeSi') that is equivalent to treating the $p(c|k,R)$ in the first hidden layer as uniform $1/C$. (II) Using the posterior t_k, the network can itself provide missing training labels (often termed 'self-labeling'; see, e.g., Lee, 2013; Triguero et al., 2015) and the 'Best versus Second Best' (BvSB) measure on t_k gives an index for classification certainty. We then train the top layer also on those inferred labels with high certainty, i.e., where the BvSB lies above a given threshold ϑ. We mark NeSi networks using self-labeling by a superscript '$+$' ('r$^+$-NeSi' and 'ff$^+$-NeSi'). (III) We apply TV-EM (Lücke, 2016) to ff$^+$-NeSi by keeping only the C' highest values in s_c (Eq. T1.4) and setting lower values to hard zero with subsequent renormalization. Such truncation can be shown to maximize the variational free energy of the mixture model with a significantly lower computational cost (Forster and Lücke, 2017). We refer to simpletrons with truncated middle layer activations as 't-NeSi'. This represents a further development of the network in Forster et al. (2015) with enhanced training and significantly improved results.

3 Parameter Tuning with Limited Labels

It is customary to regard limited numbers of labels as restriction only on training itself and not on the preceding optimization of free model parameters by using validations sets with (much) more labels than available during training. For model comparison, this however introduces a bias towards more complex models, that would be otherwise more prone to overfitting to small validation sets. We therefore train our models given a strictly limited total number of labels for the complete tuning and training procedure. For parameter tuning, we use 10 labeled training data points per class (the setting with the lowest number of labels on which models are generally compared on MNIST) with a half/half split into training and validation set. Once optimized, we keep the free parameters fixed for all following experiments. In doing so, we ensure that all our results of 10 labels per class and more are achievable by using no more labels in total than provided within each training setting. Furthermore, using only training data for parameter tuning guarantees a fully blind test set, such that the test error gives a reliable index for generalization.

4 Numerical Experiments

We apply the NeSi networks to three standard benchmarks for classification on non-negative data: the 20 Newsgroups text data set (Lang, 1995), the MNIST data set of handwritten digits (LeCun et al., 1998) and the NIST Special Database 19 of handwritten characters (Grother, 1995). We perform experiments for different proportions of randomly chosen, class-balanced labeled data and measure the mean classification error on the blind test set.

4.1 Document Classification (20 Newsgroups)

We preprocess the newsgroups data using only tf-idf weighting (Sparck Jones, 1972). No stemming, removals of stop words or frequency cutoffs were applied. We investigate semi-supervised settings of 20, 40, 200, 800 and 2000 labels in total – that is 1, 2, 10, 40 and 100 labels per class – as well as the fully labeled setting. For each setting, we present the mean test error averaged over 100 independent runs and the standard error of the mean (SEM). On each new run, a new set of class balanced labels is chosen randomly from the training set. We train our model on the full 20-class problem without any feature selection.

As reported results on the full 20-class task in the semi-supervised setting are rare, we here only compare to a hybrid of generative and discriminative RBMs (HDRBM) trained by Larochelle and Bengio (2008) using stochastic gradient descent to perform semi-supervised learning. The test errors of the methods are compared in Tab. 2. For results marked with '$(*)$', free parameters were optimized

Table 2: Test error on 20 Newsgroups.

#labels	ff-NeSi	r-NeSi	HDRBM
20	70.64 ± 0.68 $^{(*)}$	**68.68** ± 0.77 $^{(*)}$	
40	55.67 ± 0.54 $^{(*)}$	**54.24** ± 0.66 $^{(*)}$	
200	30.59 ± 0.22	**29.28** ± 0.21	
800	28.26 ± 0.10	**27.20** ± 0.07	31.8 $^{(*)}$
2000	27.87 ± 0.07	**27.15** ± 0.07	
11269	28.08 ± 0.08	**17.85** ± 0.01	23.8

using additional labels: NeSi used the same parameter setting in all semi-supervised experiments on 20 Newsgroups, which was tuned with 200 labels in total; HDRBM used 1000 labels in total for tuning in the semi-supervised setting (200 additional labels for the validation set).

4.2 Handwritten Digit Recognition (MNIST)

We perform experiments on MNIST in the semi-supervised setting using 10, 100, 600, 1000 and 3000 labels in total – that is 1, 10, 60, 100 and 300 labels per class –, which are randomly and class balanced chosen from the 10 classes. Results are given as the mean and standard error (SEM) over 100 independent repetitions, with randomly drawn, class-balanced labels.

Tab. 3 shows the results of the NeSi algorithms. Again, for results marked with '$(*)$', the free parameters were optimized using more labels than available in the given setting. We used the same parameter setting for all experiments shown here, which was tuned using 100 labels in total. As the NeSi model has no prior knowledge about spatial relations in the data, the given results are invariant to pixel permutation.

Table 3: Test error of NeSi algorithms on permutation invariant MNIST for different semi-supervised settings.

#labels	ff-NeSi	r-NeSi	ff$^+$-NeSi	r$^+$-NeSi	t-NeSi
10	55.5 ± 0.6 $^{(*)}$	29.6 ± 0.6 $^{(*)}$	10.9 ± 0.9 $^{(*)}$	17.9 ± 0.9 $^{(*)}$	$\mathbf{7.2} \pm 0.5$ $^{(*)}$
100	19.1 ± 0.3	12.4 ± 0.2	4.96 ± 0.08	4.93 ± 0.05	$\mathbf{4.23} \pm 0.07$
600	7.27 ± 0.05	6.94 ± 0.05	4.08 ± 0.02	4.34 ± 0.01	$\mathbf{3.65} \pm 0.01$
1000	5.88 ± 0.03	6.07 ± 0.03	4.00 ± 0.01	4.26 ± 0.01	$\mathbf{3.63} \pm 0.01$
3000	4.39 ± 0.02	4.68 ± 0.02	3.85 ± 0.01	4.05 ± 0.01	$\mathbf{3.52} \pm 0.01$
60000	3.27 ± 0.01	$\mathbf{2.94} \pm 0.01$	3.27 ± 0.01	$\mathbf{2.94} \pm 0.01$	$\mathbf{2.94} \pm 0.01$

Fig. 2 shows a comparison to standard and recent state-of-the-art approaches for 100 labels and more. The performance of the models is given with respect to the number of labels used during training (left-hand side) and with respect to the total number of labels used for the complete tuning and training procedure (right-hand side). For the NeSi algorithms, these plots are identical, as we only use maximally as many labels in the tuning phase as in the training phase for the shown results of 100 labels and more. All other algorithms (for lack of more comparable findings) either use a validation set with a substantial amount of additional labels than available during training or (explicitly) use the test set for parameter tuning. Also, some of the shown results (namely the TSVM, AGR, AtlasRBF and the Em-networks) were achieved in the transductive setting, where the (unlabeled) test data is included into the training process. The NeSi approaches are to our knowledge so far the closest to our goal of a competitive algorithm in the limit of as few labels as possible. Regarding classification performance, the NeSi networks achieve competitive results, surpassing even deep belief networks ('DBN-rNCA') and other recent approaches (like the 'Embed'-networks,

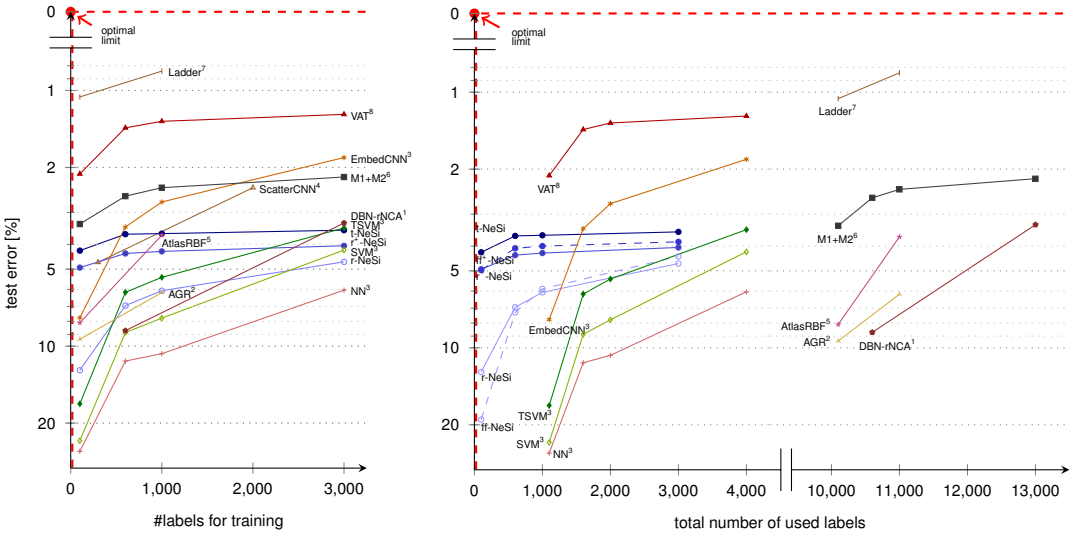


Figure 2: Classification performance of different algorithms compared against varying proportion of labeled training data. The algorithms are described in detail in the corresponding papers: [1]Salakhutdinov and Hinton (2007), [2]Liu et al. (2010), [3]Weston et al. (2012), [4]Bruna and Mallat (2013). [5]Pitelis et al. (2014), [6]Kingma et al. (2014), [7]Rasmus et al. (2015), [8]Miyato et al. (2015). All algorithms except ours use 1000 or 10 000 additional data labels (from the training or test set) for parameter tuning. For ScatterCNN (Bruna and Mallat, 2013) the validation set size is not reported.

'AGR' and 'AtlasRBF'). In the light of reduced model complexity and effectively used labels, we can furthermore compare to the few very recent algorithms with a lower error rate ('M1+M2', 'VAT' and the 'Ladder'-networks).

One competing model that so far comes closest to our limit setting of as few labels as possible is an approach which combines 10 generative adversarial networks (GANs) (Salimans et al., 2016) with 5 layers each. With down to 20 labels for training (and an however unknown additional number of labels for the validation set) the classification error of the full ensemble of 10 GANs the error was reported to be $(11.34 \pm 4.45)\%$. This compares to an error of $(7.22 \pm 0.53)\%$ for t-NeSi which was trained with only 10 labels (one per class) and used 100 labels during parameter tuning.

4.3 Large Scale Handwriting Recognition (NIST SD19)

To show large scale applicability, we show experiments on the NIST Special Database 19, containing over $800\,000$ binary 128×128 images. We preprocess the data similar to MNIST, which allows us to use the same setting for our free model parameters without retuning, and perform experiments on digit recognition (10 classes) and case-sensitive letter recognition (52 classes). The experiments are done using 1, 10, 60, 100, 300, or all labels per class. In Tab. 4, we report the mean and standard error over 10 experiments and compare to the state-of-the-art 35c-MCDNN (Cireşan et al., 2012). For the NeSi networks, the results are given for the permutation invariant task. To the best of our knowledge, this is the first system to report results for NIST SD19 in the semi-supervised setting.

Table 4: Test error on NIST SD19 data set on the task of digit and letter recognition for different total amounts of labeled data.

#lbls/class	1	10	60	100	300	fully labeled
digits (10 classes)						
#lbls total	10	100	600	1000	3000	344 307
ff$^+$-NeSi	7.6 ± 1.8	6.2 ± 0.2	6.0 ± 0.1	6.0 ± 0.1	5.70 ± 0.03	5.11 ± 0.01
r$^+$-NeSi	9.8 ± 2.4	6.1 ± 0.2	5.8 ± 0.1	5.9 ± 0.1	5.7 ± 0.1	4.52 ± 0.01
t-NeSi	**5.7** ± 0.4	**5.3** ± 0.2	**4.84** ± 0.02	**4.86** ± 0.03	**4.84** ± 0.02	4.50 ± 0.01
35c-MCDNN						**0.77**
letters (52 classes)						
#lbls total	52	520	3120	5200	15600	387361
ff$^+$-NeSi	55.7 ± 0.6	46.2 ± 0.4	44.2 ± 0.2	43.7 ± 0.2	43.0 ± 0.3	34.66 ± 0.05
r$^+$-NeSi	65.0 ± 0.9	54.1 ± 0.4	43.7 ± 0.2	**41.6** ± 0.1	**38.0** ± 0.1	31.93 ± 0.06
t-NeSi	**52.1** ± 1.1	**45.6** ± 0.4	**41.9** ± 0.3	41.8 ± 0.4	41.1 ± 0.3	33.34 ± 0.04
35c-MCDNN						**21.01**

5 Discussion

In this study, we explored classifier training on data sets with limited labels. We put a special emphasize on adhering to this restriction not only for the training phase, but the complete tuning, training and testing procedure. Our tool was a novel neural network with learning rules based on a maximum likelihood objective. Starting from hierarchical Poisson mixtures, the derived three layer directed data model can be observed to take on a form similar to learning in standard DNNs. The parameters of the network can be optimized with a very limited amount of labels and training in the same setting showed to achieve competitive results, giving the first network shown to operate using no more than 10 labels per class in total and down to a single training label per class on the investigated data sets.

Our main empirical results for the NeSi systems were obtained using the 20 Newsgroups, the MNIST and the NIST SD19 datasets. Tabs. 2 to 4 and Fig. 2 summarize the results and provide comparison to other approaches. The r-NeSi system is the best performing system for the 20 Newsgroups data set (Tab. 2), but comparative results were only available for HDRBM in the semi-supervised setting. Both on MNIST and NIST the performance of our 3-layer network in the fully labeled setting is not competitive to state-of-the-art fully supervised algorithms. Our results however do apply for the permutation invariant setting and do not take prior knowledge about two-dimensional image data into account (like convolutional networks do). More importantly, we only see a relatively mild decrease in test error when we strongly decrease the total number of used labels, with the t-NeSi consistently performing best. It has so far not been shown that other classifiers can be trained with similarly low total numbers of labels, as all comparable approaches use at least 1000 additional labels to optimize the free parameters of their respective systems (Fig. 2, right-hand-side). Furthermore, as all better performing approaches on MNIST combine different objectives to perform well with limited labels, the NeSi networks can be considered as the best performing non-hybrid approaches even if we only consider exclusively the labels for training (Fig. 2, left-hand-side).

References

Abbott, L. F. and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–1183.

Bornschein, J. and Bengio, Y. (2015). Reweighted wake-sleep. In *International Conference on Learning Representations (ICLR)*.

Bruna, J. and Mallat, S. (2013). Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 35(8):1872–1886.

Cireşan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649. IEEE.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley-Interscience (2nd Edition).

Forster, D. and Lücke, J. (2017). Truncated variational EM for semi-supervised Neural Simpletrons. In *IJCNN 2017, in press.*

Forster, D., Sheikh, A.-S., and Lücke, J. (2015). Neural Simpletrons – Minimalistic directed generative networks for learning with few labels. *arXiv preprint arXiv:1506.08448*.

Gan, Z., Henao, R., Carlson, D., and Carin, L. (2015). Learning deep sigmoid belief networks with data augmentation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 268–276.

Grother, P. J. (1995). NIST special database 19 handprinted forms and characters database. *National Institute of Standards and Technology*.

Keck, C., Savin, C., and Lücke, J. (2012). Feedforward inhibition and synaptic scaling – two sides of the same coin? *PLoS Computational Biology*, 8:e1002432.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3581–3589.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning (ICML)*, pages 331–339.

Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 536–543.

Larochelle, H. and Murray, I. (2011). The neural autoregressive distribution estimator. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 29–37.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2.

Liu, W., He, J., and Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *International Conference on Machine Learning (ICML)*, pages 679–686.

Lücke, J. (2016). Truncated variational expectation maximization. *arXiv preprint, arXiv:1610.03113*.

Lücke, J. and Sahani, M. (2008). Maximal causes for non-linear component extraction. *Journal of Machine Learning Research (JMLR)*, 9:1227–1267.

Miyato, T., Maeda, S.-i., Koyama, M., Nakae, K., and Ishii, S. (2015). Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677v6*.

Pitelis, N., Russell, C., and Agapito, L. (2014). Semi-supervised learning using an unsupervised atlas. In *Machine Learning and Knowledge Discovery in Databases*, pages 565–580. Springer.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3532–3540.

Salakhutdinov, R. and Hinton, G. E. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 412–419.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2226–2234.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Triguero, I., García, S., and Herrera, F. (2015). Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study. *Knowledge and Information Systems*, 42(2):245–284.

Weston, J., Ratle, F., Mobahi, H., and Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer.

Subsampling for Ridge Regression via Regularized Volume Sampling

Michał Dereziński
Department of Computer Science
University of California Santa Cruz
mderezin@ucsc.edu

Manfred K. Warmuth
Department of Computer Science
University of California Santa Cruz
manfred@ucsc.edu

Abstract

Given n vectors $\mathbf{x}_i \in \mathbb{R}^d$, we want to fit a linear regression model for noisy labels $y_i \in \mathbb{R}$. The ridge estimator is a classical solution to this problem. However, when labels are expensive, we are forced to select only a small subset of vectors $\mathbf{x}_i$ for which we obtain the labels y_i. We propose a new procedure for selecting the subset of vectors, such that the ridge estimator obtained from that subset offers strong statistical guarantees in terms of the mean squared prediction error over the entire dataset of n labeled vectors. The number of labels needed is proportional to the statistical dimension of the problem which is often much smaller than d. Our method is an extension of a joint subsampling procedure called volume sampling. A second major contribution is that we speed up volume sampling so that it is essentially as efficient as leverage scores, which is the main i.i.d. subsampling procedure for this task. Finally, we show theoretically and experimentally that volume sampling has a clear advantage over any i.i.d. sampling in the sparse label case.

1 Introduction

Given a matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, we consider the task of fitting a linear model[1] to a vector of labels $\mathbf{y} = \mathbf{X}^\top \mathbf{w}^* + \boldsymbol{\xi}$, where $\mathbf{w}^* \in \mathbb{R}^d$ and the noise $\boldsymbol{\xi} \in \mathbb{R}^n$ is a mean zero random vector with covariance matrix $\mathrm{Var}[\boldsymbol{\xi}] \preceq \sigma^2\mathbf{I}$ for some $\sigma > 0$. A classical solution to this task is the ridge estimator:

$$\widehat{\mathbf{w}}^*_\lambda = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} \ \|\mathbf{X}^\top\mathbf{w} - \mathbf{y}\|^2 + \lambda\|\mathbf{w}\|^2 = (\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{y}.$$

In many settings, obtaining labels y_i is expensive and we are forced to select a subset $S \subseteq \{1..n\}$ of label indices. Let $\mathbf{y}_S \in \mathbb{R}^{|S|\times 1}$ be the sub vector of labels indexed by S and $\mathbf{X}_S \in \mathbb{R}^{d\times|S|}$ be the columns of $\mathbf{X}$ indexed by S. We will show that if S is sampled with a new variant of *volume sampling* [3] on the columns of $\mathbf{X}$, then the ridge estimator for the subproblem $(\mathbf{X}_S, \mathbf{y}_S)$

$$\widehat{\mathbf{w}}^*_\lambda(S) = (\mathbf{X}_S\mathbf{X}_S^\top + \lambda\mathbf{I})^{-1}\mathbf{X}_S\mathbf{y}_S$$

has strong generalization properties with respect to the full problem $(\mathbf{X}, \mathbf{y})$.

Volume sampling is a sampling technique which has received a lot of attention recently [3, 5, 6, 7, 15]. For a fixed size $s \geq d$, the original variant samples $S \subseteq \{1..n\}$ of size s proportional to the squared volume of the parallelepiped spanned by the rows of $\mathbf{X}_S$ [3]:

$$P(S) \propto \det(\mathbf{X}_S\mathbf{X}_S^\top). \tag{1}$$

[1] This setting can easily be extended to "non-linear models" via kernelization.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

A simple approach for implementing volume sampling (just introduced in [5]) is to start with the full set of column indices $S = \{1..n\}$ and then (in reverse order) select an index i in each iteration to be eliminated from set S with probability proportional to the change in matrix volume caused by removing the ith column:

$$\text{Sample } i \sim P(i \,|\, S) = \frac{\det(\mathbf{X}_{S_{-i}}\mathbf{X}_{S_{-i}}^\top)}{(|S|-d)\det(\mathbf{X}_S\mathbf{X}_S^\top)}, \tag{2}$$

$$\text{Update } S \leftarrow S - \{i\}.$$

(Reverse Iterative Volume Sampling)

Note that when $|S| < d$, then all matrices $\mathbf{X}_S\mathbf{X}_S^\top$ are singular, and so the distribution becomes undefined. Motivated by this limitation, we propose a regularized variant, called λ-regularized volume sampling:

$$\text{Sample } i \sim P(i \,|\, S) \propto \frac{\det(\mathbf{X}_{S_{-i}}\mathbf{X}_{S_{-i}}^\top + \lambda\mathbf{I})}{\det(\mathbf{X}_S\mathbf{X}_S^\top + \lambda\mathbf{I})}, \tag{3}$$

$$\text{Update } S \leftarrow S - \{i\}.$$

(λ-Regularized Volume Sampling)

Note that in the special case of no regularization (i.e. $\lambda = 0$), then (3) sums to $\frac{1}{|S|-d}$ (see equality (2)). However when $\lambda > 0$, then the sum of (3) depends on all columns of $\mathbf{X}_S$ and not just the size of S. This makes regularized volume sampling more complicated and certain equalities proven in [5] for $\lambda = 0$ become inequalities.

Nevertheless, we were able to show that the proposed λ-regularized distribution exhibits a fundamental connection to ridge regression, and introduce an efficient algorithm to sample from it in time $O((n+d)d^2)$. In particular, we prove that when S is sampled according to λ-regularized volume sampling with $\lambda \leq \frac{\sigma^2}{\|\mathbf{w}^*\|^2}$, then the mean squared prediction error (MSPE) of estimator $\widehat{\mathbf{w}}_\lambda^*(S)$ over the entire dataset $\mathbf{X}$ is bounded:

$$\mathbb{E}_S\mathbb{E}_{\boldsymbol{\xi}}\,\frac{1}{n}\|\mathbf{X}^\top(\widehat{\mathbf{w}}_\lambda^*(S) - \mathbf{w}^*)\|^2 = O\left(\frac{\sigma^2 d_\lambda}{s}\right),$$

$$\text{where} \quad d_\lambda = \operatorname{tr}(\mathbf{X}^\top(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{X}) \tag{4}$$

FastRegVol: λ-Regularized Volume Sampling

```
Z ← (XX^⊤ + λI)^{-1}
S ← {1..n}
while |S| > s
  repeat
    Sample i uniformly out of S
    h_i ← 1 − x_i^⊤ Z x_i
    Sample A ∼ Bernoulli(h_i)
  until A = 1
  S ← S − {i}
  Z ← Z + h_i^{-1} Z x_i x_i^⊤ Z
end
return S
```

is the statistical dimension. If λ_i are the eigenvalues of $\mathbf{X}\mathbf{X}^\top$, then $d_\lambda = \sum_{i=1}^d \frac{\lambda_i}{\lambda_i+\lambda}$. Note that d_λ is decreasing with λ and $d_0 = d$. If the spectrum of the matrix $\mathbf{X}\mathbf{X}^\top$ decreases quickly then d_λ does so as well with increasing λ. When λ is properly tuned then d_λ is the effective degrees of freedom of $\mathbf{X}$. Our new lower bounds will show that the above upper bound for regularized volume sampling is essentially optimal with respect to the choice of a subsampling procedure.

Volume sampling can be viewed as a non-i.i.d. extension of leverage score sampling [8], a widely used method where columns are sampled independently according to their leverage scores. Volume sampling has been shown to return better column subsets than its i.i.d. counterpart in many applications like experimental design, linear regression and graph theory [3, 5]. In this paper we additionally show that any i.i.d. subsampling with respect to any fixed distribution such as leverage score sampling can require $\Omega(d_\lambda \ln(d_\lambda))$ labels to achieve good generalization for ridge regression, compared to $O(d_\lambda)$ for regularized volume sampling. We reinforce this claim experimentally in Section 3.

The main obstacle against using volume sampling in practice has been high computational cost. Only recently, the first polynomial time algorithms have been proposed for exact [5] and approximate [15] volume sampling (see Table 1 for comparison). In particular, the fastest algorithm for exact volume sampling[2] is $O(n^2 d)$ whereas exact leverage score sampling[3] is $O(nd^2)$ (in both cases, the dependence on sample size s is not asymptotically significant). In typical settings for experimental design [9] and active learning [18], quality of the sample is more important than the runtime.

[2]The exact time complexity is $O((n-s+d)nd)$ which is $O(n^2 d)$ for $s < n/2$.

[3]Approximate leverage score sampling methods achieve even better runtime of $\widetilde{O}(nd + d^3)$.

However for many modern datasets, the number of examples n is much larger than d, which makes existing algorithms for volume sampling infeasible. In this paper, we give an easy-to-implement volume sampling algorithm, called FastRegVol, that runs in time $O(nd^2)$. Thus we give the first volume sampling procedure which is essentially linear in n and matches the time complexity of exact leverage score sampling. For example, dataset MSD from the UCI data repository [16] has $n = 464,000$ examples with dimension $d = 90$. Our algorithm performed volume sampling on this dataset in 39 seconds, whereas the previously best $O(n^2d)$ algorithm [5] did not finish within 24 hours. Sampling with leverage scores took 12 seconds on this data set. Finally our procedure also achieves regularized volume sampling for any $\lambda > 0$ with the running time of $O((n + d)d^2)$.

1.1 Related work

	Exact	Approximate
[15]	$O(n^4 s)$	$\widetilde{O}(nd^2s^3)$
[5]	$O(n^2 d)$	-
here	$O(nd^2)$	-

Table 1: Comparison of runtime for exact and approximate volume sampling algorithms, where $d \leq s \leq n$.

Many variants of probability distributions based on the matrix determinant have been studied in the literature, including Determinantal Point Processes (DPP) [14], k-DPP's [13] and volume sampling [3, 5, 6, 15], with applications to matrix approximation [7], clustering [12], recommender systems [10], etc. More recently, further theoretical results suggesting applications of volume sampling in linear regression were given by [5], where an expected loss bound for the unregularized least squares estimator was given under volume sampling of size $s = d$. Moreover, Reverse Iterative Volume Sampling – a technique enhanced in this paper with a regularization – was first proposed in [5].

Subset selection techniques for regression have long been studied in the field of experimental design [9]. More recently, computationally tractable techniques have been explored [2]. Statistical guarantees under i.i.d. subsampling in kernel ridge regression have been analyzed for uniform sampling [4] and leverage score sampling [1]. In this paper, we propose the first tractable non-i.i.d. subsampling procedure with strong statistical guarantees for the ridge estimator and show its benefits over using i.i.d. sampling approaches.

For the special case of volume sampling size $s = d$, a polynomial time algorithm was developed by [6], and slightly improved by [11], with runtime $O(nd^3)$. An exact sampling algorithm for arbitrary $s \geq d$ was given by [5], with time complexity $O((n - s + d)nd)$ which is $O(n^2d)$ when $s < n/2$. Also, [15] proposed a Markov-chain procedure which generates ϵ-approximate volume samples in time $\widetilde{O}(nd^2s^3)$. The algorithm proposed in this paper, running in time $O(nd^2)$, enjoys a direct asymptotic speed-up over all of the above methods. Moreover, the procedure suffers only a small constant factor overhead over computing exact leverage scores of matrix $\mathbf{X}$.

2 Main results

The main contributions[4] of this paper are two-fold:

1. **Statistical**: We define a regularized variant of volume sampling and show that it offers strong generalization guarantees for ridge regression in terms of mean squared error (MSE) and mean squared prediction error (MSPE).
2. **Algorithmic**: We propose a simple implementation of volume sampling, which not only extends the procedure to its regularized variant, but also offers a significant runtime improvement over the existing methods when $n \gg d$.

The key technical result of this paper, needed to obtain statistical guarantees for ridge regression, is the following property of regularized volume sampling (where d_λ is defined as in (4)):

Theorem 1 *For any $\mathbf{X} \in \mathbb{R}^{d\times n}$, $\lambda \geq 0$ and $s \geq d_\lambda$, let S be sampled according to λ-regularized size s volume sampling from $\mathbf{X}$. Then,*

$$\mathbb{E}_S \left(\mathbf{X}_S\mathbf{X}_S^\top + \lambda\mathbf{I}\right)^{-1} \preceq \frac{n - d_\lambda + 1}{s - d_\lambda + 1}(\mathbf{X}\mathbf{X}^\top + \lambda\mathbf{I})^{-1},$$

where $\preceq$ denotes a positive semi-definite inequality between matrices.

[4]Due to space limitations, we omit the proofs of our results in this extended abstract.

As a consequence of Theorem 1, we show that ridge estimators computed from volume sampled subproblems offer statistical guarantees with respect to the full regression problem $(\mathbf{X}, \mathbf{y})$, despite observing only a small portion of the labels.

Theorem 2 *Let $\mathbf{X} \in \mathbb{R}^{d\times n}$ and $\mathbf{w}^* \in \mathbb{R}^d$, and suppose that $\mathbf{y} = \mathbf{X}^\top \mathbf{w}^* + \boldsymbol{\xi}$, where $\boldsymbol{\xi}$ is a mean zero vector with $\mathrm{Var}[\boldsymbol{\xi}] \preceq \sigma^2 \mathbf{I}$. Let S be sampled according to λ-regularized size $s \geq d_\lambda$ volume sampling from $\mathbf{X}$ and $\widehat{\mathbf{w}}^*_\lambda(S)$ be the λ-ridge estimator of $\mathbf{w}^*$ computed from subproblem $(\mathbf{X}_S, \mathbf{y}_S)$. Then, if $\lambda \leq \frac{\sigma^2}{\|\mathbf{w}^*\|^2}$, we have*

$$\textbf{(MSPE)} \qquad \mathbb{E}_S \mathbb{E}_{\boldsymbol{\xi}} \frac{1}{n} \|\mathbf{X}^\top (\widehat{\mathbf{w}}^*_\lambda(S) - \mathbf{w}^*)\|^2 \leq \frac{\sigma^2 d_\lambda}{s - d_\lambda + 1},$$

$$\textbf{(MSE)} \qquad \mathbb{E}_S \mathbb{E}_{\boldsymbol{\xi}} \|\widehat{\mathbf{w}}^*_\lambda(S) - \mathbf{w}^*\|^2 \leq \frac{\sigma^2 n \,\mathrm{tr}((\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})^{-1})}{s - d_\lambda + 1}.$$

Next, we present two lower-bounds for MSPE of a subsampled ridge estimator which show that the statistical guaranties achieved by regularized volume sampling are nearly optimal for $s \gg d_\lambda$ and better than standard approaches for $s = O(d_\lambda)$. In particular, we show that non-i.i.d. nature of volume sampling is essential if we want to achieve good generalization when the number of labels is close to d_λ. Namely, for certain data matrices, any subsampling procedure which selects examples in an i.i.d. fashion (e.g., leverage score sampling), requires more than $d_\lambda \ln(d_\lambda)$ labels to achieve MSPE below σ^2, whereas volume sampling obtains that bound for any matrix with $2d_\lambda$ labels.

Theorem 3 *For any $p \geq 1$ and $\sigma \geq 0$, there is $d \geq p$ such that for any sufficiently large n divisible by d there exists a matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ such that*

$$d_\lambda(\mathbf{X}) \geq p \quad \text{for any} \quad 0 \leq \lambda \leq \sigma^2,$$

and for each of the following two statements there is a vector $\mathbf{w}^* \in \mathbb{R}^d$ for which the corresponding regression problem $\mathbf{y} = \mathbf{X}^\top \mathbf{w}^* + \boldsymbol{\xi}$ with $\mathrm{Var}[\boldsymbol{\xi}] = \sigma^2 \mathbf{I}$ satisfies that statement:

1. *For any subset $S \subseteq \{1..n\}$ of size s,*

$$\mathbb{E}_{\boldsymbol{\xi}} \frac{1}{n} \|\mathbf{X}^\top (\widehat{\mathbf{w}}^*_\lambda(S) - \mathbf{w}^*)\|^2 \geq \frac{\sigma^2 d_\lambda}{s + d_\lambda};$$

2. *For multiset $S \subseteq \{1..n\}$ of size $s \leq d_\lambda(\ln(d_\lambda) - 1)$, sampled i.i.d. from any distribution,*

$$\mathbb{E}_S \mathbb{E}_{\boldsymbol{\xi}} \frac{1}{n} \|\mathbf{X}^\top (\widehat{\mathbf{w}}^*_\lambda(S) - \mathbf{w}^*)\|^2 \geq \sigma^2.$$

Finally, we propose an algorithm for regularized volume sampling (see FastRegVol in Section 1) which runs in time $O((n + d)d^2)$. For the previously studied case of $\lambda = 0$, this algorithm offers a significant asymptotic speed-up over existing volume sampling algorithms (both exact and approximate).

Theorem 4 *For any $\lambda, \delta, s \geq 0$, there is an algorithm sampling according to λ-regularized size s volume sampling, that with probability at least $1 - \delta$ runs in time*[5]

$$O\left(\left(n + d + \log\left(\frac{n}{d}\right)\log\left(\frac{1}{\delta}\right)\right) d^2\right).$$

When $n > d$ the time complexity of our proposed algoirthm is not deterministic, but its dependence on the failure probability δ is very small – even for $\delta = 2^{-n}$ the time complexity is still $\widetilde{O}(nd^2)$.

3 Experiments

In this section we experimentally evaluate the proposed algorithm for regularized volume sampling, in terms of runtime and the quality of subsampled ridge estimators. The list of implemented algorithms is:

[5] We are primarily interested in the case where $n \geq d$. However, if $n < d$ and $\lambda > 0$, then our techniques can be adapted to obtain an $O(n^2 d)$ algorithm.

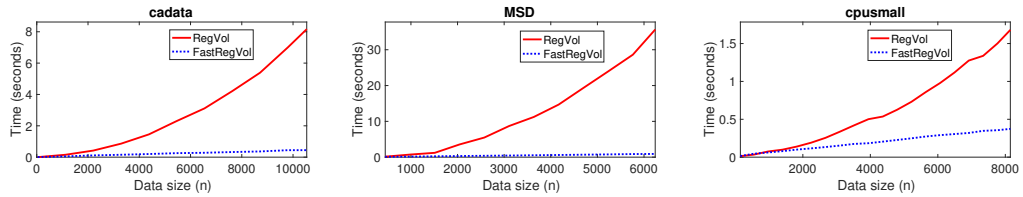


Figure 1: Comparison of runtime between FastRegVol and RegVol (adapted from [5]).

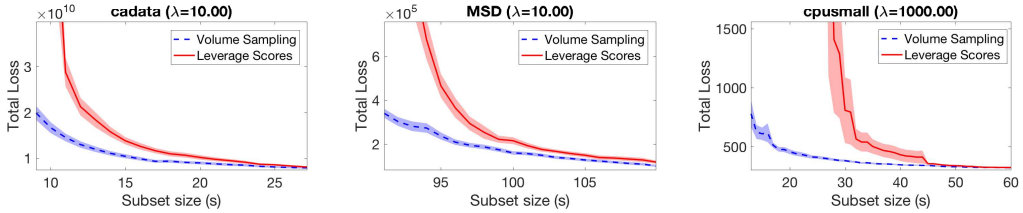


Figure 2: Comparison of loss of the subsampled ridge estimator when using regularized volume sampling vs using leverage score sampling (confidence regions based on standard error of the mean).

1. Regularized Volume Sampling: **FastRegVol** (our algorithm); **RegVol**[6] – adapted from [5];
2. Leverage Score Sampling[7] (LSS) – a popular i.i.d. sampling technique [17], where examples are selected w.p. $P(i) = (\mathbf{x}_i^\top(\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{x}_i)/d$.

The experiments were performed on several benchmark linear regression datasets [16]. Table 2 lists those datasets along with running times for sampling dimension many columns with each method[8].

In Figure 1 we plot the runtime against varying values of n (using portions of the datasets), to compare how FastRegVol and RegVol scale with respect to the datasize. We observe that unlike RegVol, our new algorithm exhibits linear dependence on n, thus it is much better suited for running on large datasets.

Dataset	$d \times n$	RegVol	FastRegVol	LSS
cadata	8×21k	33.5s	0.9s	0.1s
MSD	90×464k	>24hr	39s	12s
cpusmall	12×8k	1.7s	0.4s	0.07s

Table 2: A list of regression datasets, with runtime comparison between RegVol [5] and FastRegVol. We also provide runtime for obtaining exact leverage score samples (LSS).

3.1 Subset selection for ridge regression

We applied volume sampling to the task of subset selection for linear regression, by evaluating the subsampled ridge estimator $\widehat{\mathbf{w}}_\lambda^*(S)$ using the total loss over the full dataset:

$$L(\widehat{\mathbf{w}}_\lambda^*(S)) \overset{\text{def}}{=} \frac{1}{n}\|\mathbf{X}^\top \widehat{\mathbf{w}}_\lambda^*(S) - \mathbf{y}\|^2.$$

We computed $L(\widehat{\mathbf{w}}_\lambda^*(S))$ for a range of subset sizes and values of λ, when the subsets are sampled according to λ-regularized volume sampling and leverage score sampling. The results were averaged over 20 runs of each experiment. For clarity, Figure 2 shows the results only with one value of λ for each dataset, chosen so that the subsampled ridge estimator performed best (on average over all samples of preselected size s). The results on all datasets show that when only a small number of labels s is obtainable, then regularized volume sampling offers better estimators than leverage score sampling (as predicted by Theorems 2 and 3).

4 Conclusions

We proposed a sampling procedure called regularized volume sampling, which offers near-optimal statistical guarantees for subsampled ridge estimators. We also gave a new algorithm for volume sampling which is essentially as efficient as i.i.d. leverage score sampling.

[6] The volume sampling algorithm of [5] can be trivially adapted to the regularized case (i.e. where $\lambda > 0$).

[7] Regularized variants of leverage scores have also been considered in context of kernel ridge regression [1]. However, in our experiments regularizing leverage scores did not provide any improvements.

[8] Dataset MSD was too big for RegVol to finish in reasonable time.

References

[1] Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 775–783, Cambridge, MA, USA, 2015. MIT Press.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, Aarti Singh, and Yining Wang. Near-optimal design of experiments via regret minimization. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 126–135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[3] Haim Avron and Christos Boutsidis. Faster subset selection for matrices and applications. *SIAM Journal on Matrix Analysis and Applications*, 34(4):1464–1499, 2013.

[4] Francis Bach. Sharp analysis of low-rank kernel matrix approximations. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 185–209, Princeton, NJ, USA, 12–14 Jun 2013. PMLR.

[5] Michał Dereziński and Manfred K. Warmuth. Unbiased estimates for linear regression via volume sampling. *CoRR*, abs/1705.06908, 2017.

[6] Amit Deshpande and Luis Rademacher. Efficient volume sampling for row/column subset selection. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 329–338, Washington, DC, USA, 2010. IEEE Computer Society.

[7] Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 1117–1126, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics.

[8] Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *J. Mach. Learn. Res.*, 13(1):3475–3506, December 2012.

[9] Valeri Vadimovich Fedorov, W.J. Studden, and E.M. Klimko, editors. *Theory of optimal experiments*. Probability and mathematical statistics. Academic Press, New York, 1972.

[10] Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. Bayesian low-rank determinantal point processes. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 349–356, New York, NY, USA, 2016. ACM.

[11] Venkatesan Guruswami and Ali Kemal Sinop. Optimal column-based low-rank matrix reconstruction. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 1207–1214, Philadelphia, PA, USA, 2012. Society for Industrial and Applied Mathematics.

[12] Byungkon Kang. Fast determinantal point process sampling with application to clustering. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 2319–2327, USA, 2013. Curran Associates Inc.

[13] Alex Kulesza and Ben Taskar. k-DPPs: Fixed-Size Determinantal Point Processes. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1193–1200. Omnipress, 2011.

[14] Alex Kulesza and Ben Taskar. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA, 2012.

[15] C. Li, S. Jegelka, and S. Sra. Column Subset Selection via Polynomial Time Dual Volume Sampling. *ArXiv e-prints*, March 2017.

[16] M. Lichman. UCI machine learning repository, 2013.

[17] Michael W. Mahoney. Randomized algorithms for matrices and data. *Found. Trends Mach. Learn.*, 3(2):123–224, February 2011.

[18] Masashi Sugiyama and Shinichi Nakajima. Pool-based active learning in approximate linear regression. *Mach. Learn.*, 75(3):249–274, June 2009.

Learning to Search via Self-Imitation with Application to Risk-Aware Planning

Jialin Song*
Caltech
jssong@caltech.edu

Ravi Lanka*
JPL
ravi.kiran@jpl.nasa.gov

Albert Zhao
Caltech
azzhao@caltech.edu

Yisong Yue
Caltech
yyue@caltech.edu

Masahiro Ono
JPL
masahiro.ono@jpl.nasa.gov

Abstract

We study the problem of learning a good local search policy for solving combinatorial optimization problems such as mixed integer linear programs. To do so, we propose the self-imitation learning setting, which builds upon imitation learning in two ways. First, self-imitation uses feedback provided by retroactive analysis of demonstrated search traces. Second, the policy can learn from its own decisions and mistakes without requiring repeated feedback from an external expert. Combined, these two properties allow our approach to iteratively scale up to larger problem sizes than the initial problem size for which expert demonstrations were provided. We showcase the effectiveness of our approach on the challenging problem of risk-aware planning.

1 Introduction

Many complex prediction and decision-making tasks require solving challenging optimization problems such as mixed integer linear programs (MILPs) [1]. Examples include MAP inference in graphical models [2, 3, 4], path planning [5, 6, 7], transportation scheduling [8] and many others.

Since solving MILPs is NP-hard, one typically resorts to branch-and-bound [9] combined with local search heuristics for making branching and bounding decisions. One key design question in branch-and-bound (and virtually all other local search heuristics) is how to prioritize the search space, e.g., prioritize which integer variables to set first. The conventional approach is to manually design such heuristics to exploit specific structural assumptions (cf. [10, 11]). However, this conventional approach is labor intensive and relies on human experts developing a strong understanding of structural properties of some class of MILPs.

In this paper, we take a learning approach to finding an effective search heuristic over a (focused) distribution of MILP instances. We build upon the imitation learning paradigm [12, 13] and propose the self-imitation learning approach, where the policy can iteratively learn from its own decisions and mistakes without requiring continuous feedback from an external expert. Our approach improves upon previous imitation approaches for solving MILPs [14] in two major aspects. First, our learning approach allows the trained policy to iteratively refine towards new feasible solutions that may be easier for the policy to find than those in the original training set. Second, by learning from its own decisions, our approach can then train on larger problem instances than contained in the original supervised training set. We also provide a mistake bound

*Equal contribution.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

of our policy in a restricted setting of the general problem. Finally, we showcase the practicality of our approach on an application of risk-aware path planning that is built on top of MILP-based path planning [5] and chance-constrained optimization [15], where we demonstrate improvements upon prior imitation learning work [14] as well as commercial solvers such as Gurobi.

2 Related Work

Imitation learning is an increasingly popular paradigm, whereby a policy is trained to mimic the decision-making of a expert or oracle [13, 12]. Existing imitation learning approaches for solving mixed integer linear programs [14] do not iteratively learn from their own mistakes, and are restricted to solving fixed-size problem instances. In contrast, our self-imitation approach is able to learn from its own mistakes as well as train on larger problem instances than contained in the original supervised training set.

Other work on using machine learning to learn a branch-and-bound policy focused on designing a good feature representation for the variable selection procedure [16]. In contrast, we adopt the imitation learning reduction paradigm [12, 14], so that we can leverage powerful function classes such as deep learning in order to avoid requiring a heavily engineered feature representation.

Our self-imitation approach bears some affinity to other imitation learning approaches that aim to exceed the performance of the oracle teacher [17]. One key difference is that we are effectively using self-imitation as a form of transfer learning by learning to solve problem instances of increasing size in the absence of expert demonstrations.

3 Problem Setting & Preliminaries

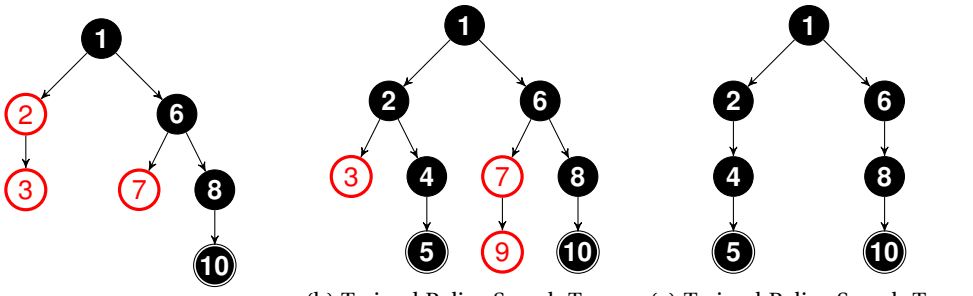
Learning a Search Policy. We consider the following search problem: given a problem instance x (which includes the initial state s_0) drawn from some distribution $\mathscr{D}$, an agent follows a policy $\pi \in \Pi$ which chooses an action $a \in A$ at each non-terminal state s, i.e., $\pi(x,s) \to a$. The agent then transitions to a new state s' after action a. The search ends once the agent arrives at a terminal state. The objective is to train a policy that quickly reaches a terminal state.

Mixed Integer Linear Programs. We focus on policies that solve mixed integer linear programs (MILPs). MILPs are optimization problems of the following form:

$$\begin{array}{ll} \text{minimize} & c^t z \\ \text{subject to} & Az \preceq b \\ & z_i \in \mathbb{Z}, i \in I \end{array}$$

A popular procedure used to solve MILPs is branch-and-bound, which can be visualized as a tree search scheme, such as depicted in Figure 1a. Each node in the search tree corresponds to a state s defined above. Each internal node of the search tree is a partial setting of the integer variables, and the remaining variables are allowed to be real-valued, yielding an LP-relaxation of the MILP. If the solution of the LP-relaxation happens to satisfy all the integer constraints, we have discovered a new feasible solution. Otherwise, we choose a variable z_i in the solution with value z_i^0, which should be an integer but is not, and generate two branches corresponding to constraints $z_i \le \lfloor z_i^0 \rfloor$ and $z_i \ge \lceil z_i^0 \rceil$, respectively, and decide one branch to continue searching. At each new node, the LP-relaxation and branching action are performed again. The search continues until some termination criterion is met, and the best feasible solution found is returned.

Imitation Learning. We build upon the imitation learning paradigm to learn a good policy. In imitation learning, there is typically an expert policy π_{expert} from which a trained policy acquires feedback on its actions. The expert feedback are actions that π_{expert} would have taken in those states encountered by the trained policy. We will build upon the DAgger algorithm [12], where the trained policy iteratively learns to make action decisions more similar to those of an expert. Central to DAgger and other imitation learning algorithm is the availability of π_{expert}, which can be a human expert or another (expensive) solver. However, this assumption does not always hold. As an alternative to expert feedback, we show that our trained policy can derive feedback for itself by examining its past decisions, a process we call *self-imitation*.



(a) Expert Search Trace

(b) Trained Policy Search Trace before Self-Imitation

(c) Trained Policy Search Trace after Self-Imitation

Figure 1: An example where the expert trace is a subset of the policy trace (the numbers on the expert trace are inherited from the policy trace; nodes with double circles are terminal states). The data collected via imitation learning and self-imitation learning differ in two aspects. First, there is no clear expert feedback on nodes missing from the expert trace, e.g. node 4. Second, for nodes common in two traces, sometimes the feedback will be different. In this example, our policy has discovered an additional terminal node 5. Self-imitation guides the training of the policy to more efficiently search for node 5 in the cases where it chooses to branch to node 2 over node 6.

4 Self-Imitation Approach

We now describe the self-imitation learning approach for training a MILP search policy. Our approach builds upon the data aggregation imitation learning framework (DAgger) [12, 14]. We present our overall approach in two steps. First, Algorithm 1 describes our core self-imitation approach for learning a search policy to solve a distribution of fixed-size MILP instances. Secondly, Algorithm 2 explains how self-imitation learning can scale up beyond the original problem size.

Core Algorithm. We assume access to an initial dataset of expert demonstrations to help bootstrap the policy. We enter an iterative self-imitation learning phase. At each iteration, we first run the current policy (potentially blended with an exploration policy), until reaching a terminal state. Then, we identify the retrospective optimal path (from the initial state to the terminal state) and collect data on mistakes made with respect to this path. The reasoning about hindsight path $\pi^*(P)$ is what distinguishes our self-imitation approach from conventional imitation learning. In particular, we do not require an expert to repeatedly provide feedback on the roll-out traces. Instead, self-imitation only relies on a (light-weight) algorithm to compute the optimal path retrospectively. For example, in tree search problems, an optimal path is obtained by tracing parent nodes from a solution node, which takes linear time to compute. In Figure 1b, if the policy is a ranking-based one, the collected feedback will contain pairwise preference such as $4 \prec 3$ and $8 \prec 7$ from this specific search trace. We can thus avoid repeatedly querying the expert by training on hindsight optimal paths of traces generated by our policy – hence self-imitation.

Generalized Self-Imitation & Scaling Up. Another major benefit of self-imitation is that it is not constrained by the problem instance size. Thus, one can apply self-imitation to problem of sizes beyond those in the initial dataset consisting of expert demonstrations. Algorithm 2 describes our generalized self-imitation approach that iteratively learns to solve increasingly larger MILP instances using Algorithm 1 as a subroutine. We start by training on demonstrated feasible solutions from some expert (such as Gurobi) on MILP instances of size S_1. If no such expert is available, we can set S_1 small enough such that exhaustive search is tractable.

5 Theoretical Results

Our theoretical analysis aims to compare self-imitation learning with conventional imitation learning. To do so, we analyze the connection between the feedback derived from self-imitation and expert demonstration. Consider the search trace P_1 generated by a trained policy and P_2 generated by an expert policy. If $P_2 \subseteq P_1$, the self-imitation will collect a dataset that aligns better with the objective than one collected using imitation learning, as explained by an example in Figure 1. In other words, consider a policy π' outside of the policy class Π. π' looks at a successful

Algorithm 1: Self-Imitation Policy Learning

Inputs:
N: number of iterations
π_1: initial policy trained by imitating expert traces
α: mixing parameter
D_0: expert traces dataset
$D = D_0$
for $i \leftarrow 1$ **to** N **do**
  train π_i on D
  $\hat{\pi}_i \leftarrow \alpha\pi_i + (1-\alpha)\pi_{explore}$ (optionally explore)
  run $\hat{\pi}_i$ to generate trace P
  compute the retrospective optimal path $\pi^*(P)$
  collect new dataset D_i for each node based on $\pi^*(P)$
  $D \leftarrow D \cup D_i$
end
return best π_i on validation

Algorithm 2: Generalized Self-Imitation

Inputs:
S_1: initial problem size
S_2: target problem size
π_{S_1}: policy trained on expert data of problem size S_1
for $s \leftarrow S_1 + 1$ **to** S_2 **do**
  generate problem instances P_s of size s
  train π_s via Algorithm 1 by running π_{s-1} on P_s
end

search trace and computes the retrospective optimal path. π' always takes the minimal number of time steps (in the MILP case, the number of integer variables). Since the goal of self-imitation is to minimize the number of time steps, the feedback from the retrospective reasoning is exactly the feedback from π'. Hence we have the following proposition.

Proposition 1. *Assume π_{S_1} is a policy trained using imitation learning on problem size S_1. If, during the scaling-up training process to problems of size $S_2 > S_1$, the trained policy search trace, starting from π_{S_1}, always contains the expert search trace, then the final error rate ϵ_{S_2} on problems of size S_2 is at most that obtained by running imitation learning directly on problems of size S_2.*

Proof. By our assumption on the trace inclusion property, the dataset obtained by self-imitation corresponds to the right loss objective while the dataset collected by imitation learning does not. So the error rate trained on self-imitation learning data will be at most that trained on imitation learning. □

6 Experimental Results

Our empirical result is a case study on the task of risk-aware planning that is built on top of MILP-based path planning [5] and chance-constrained optimization [15].

Risk-Aware Path Planning. We briefly describe the problem setup of risk-aware planning. Appendix A in the supplementary material contains a detailed description. Given a start point, a goal point, a set of polygonal obstacles, and an upper bound of the probability of failure (risk bound), we need to find a path, represented by a sequence of way points, that minimizes an objective function while limiting the probability of failure to the risk bound. This task can be formulated as MILPs.

Experimental Setup. The experiment is conducted on a set of 150 different instances of randomly generated obstacle maps with 10 obstacles each. We split them 50/50/50 for train/validation/test. We used a commercially available MILP solver Gurobi to generate expert solutions. Details on dataset generation can be found in the Appendix. For this study, we set the risk bound $\delta = 0.02$ and vary the number of way points from 10 to 14, in increment of 1. The number of integer variables range from 400 to 560.

Policy Class. Our policy class consists of a node ranking model and a pruning model. We use RankNet [18] as the ranking model, instantiated using a 2-layer neural network. For the pruning model, we train a 1-layer neural network classifier.

Method	Explored nodes (Avg)	Optimality gap (Node Limit)
Imitation Learning [14]	49	48%
Self-Imitation Learning	43	31.2%

Table 1: Comparison of Imitation Learning and Self-Imitation Learning for 10 way points test data

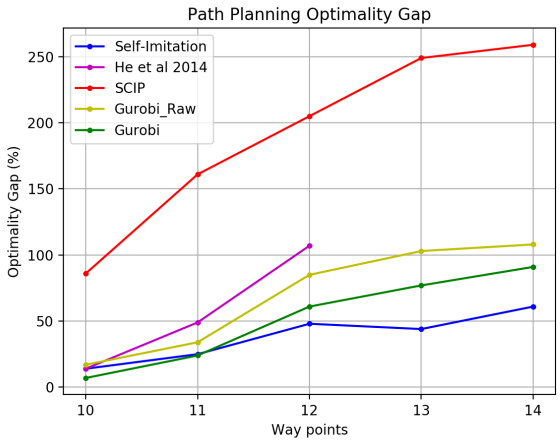


Figure 2: Mean Percentage Optimality Gap on the Risk-Aware Path Planning held-out test data.

Methods Compared. We compared our method with a commercial solver Gurobi, SCIP [19], as well as the previous imitation learning approach by [14]. Due to the difference in the implementation, we use the number of explored nodes as a proxy for runtime, i.e. we compare the optimality gap of the algorithms at the same number of explored nodes. We also compare against a version of Gurobi that de-prioritized primal heuristics, which we call Gurobi_Raw, in order to compare the local search branch-and-bound behavior. Since the method in [14] relies on supervised demonstrations by an expert, it is only trained on the reference problem size.

Main Results. We conduct two sets of experiments to demonstrate: 1) improvement over imitation learning [14] brought by self-imitation learning within the reference problem size; and 2) scaling ability of self-imitation learning to larger problem sizes.

In the first experiment, we compare against [14] within the reference problem size (10 way points). The results are summarized in the Table 1. Our approach provides a 15% relative improvement in search time and 50% relative improvement in solution quality. These results suggest that initial expert demonstrations were sub-optimal for the inductive biases of our policy class, and self-imitation was able to discover new feasible solutions easier for our policy class to learn.

In the second experiment, we validate the ability of self-imitation learning to scale up in a self-supervised fashion. A policy is initially trained on the reference scale (10 way points) using the expert solutions. It is then used to train on the next problem scale (11 way points) via Algorithm 2. We also utilized some exploration schemes into our policy (see Appendix C). Upon convergence, this process is repeated on increasing problem scales up to 14 way points. Figure 2 depicts the optimality gap as we scale up to larger problem instances. We see that self-imitation learning is able to effectively scale up. Note that method in [14] and SCIP failed in finding feasible solutions for $\sim 60\%$ and $\sim 20\%$, respectively, on the test instances, when scaling up beyond 12 way points. We thus did not test [14] beyond 12 way points. Also note that the self-imitation approach achieved a comparable performance with the expert solver (Gurobi) even though it is trained with expert feedback only at the initial problem scale (10 way points).

7 Conclusion

We have presented the self-imitation approach for training search policies to solve MILPs. Our approach extends conventional imitation learning by being able to learn good policies without requiring repeated queries to an expert. Our theoretical analysis shows that, under certain assumptions, the self-imitation learning scheme is provably more powerful and general than conventional imitation learning. We validated our approach on the challenging problem of risk-aware planning, where we demonstrated both performance gains over conventional imitation learning and the ability to scale up to large problem instances not tractably solvable by commercial solvers.

References

[1] Robert S Garfinkel and George L Nemhauser. *Integer programming*, volume 4. Wiley New York, 1972.

[2] Min Sun, Murali Telaprolu, Honglak Lee, and Silvio Savarese. Efficient and exact map-mrf inference using branch and bound. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

[3] Alexander Schwing and Raquel Urtasun. Efficient exact inference for 3d indoor scene understanding. *European Conference on Computer Vision (ECCV)*, 2012.

[4] Xian Qian and Yang Liu. Branch and bound algorithm for dependency parsing with non-local features. *Transactions of the Association for Computational Linguistics (TACL)*, 1:37–48, 2013.

[5] Tom Schouwenaars, Bart DeMoor, Eric Feron, and Jonathan How. Mixed integer programming for multi-vehicle path planning. In *In European Control Conference 2001*, pages 2603–2608, 2001.

[6] Masahiro Ono, Brian C Williams, and Lars Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research (JAIR)*, 46:511–577, 2013.

[7] Tichakorn Wongpiromsarn, Ufuk Topcu, Necmiye Ozay, Huan Xu, and Richard M Murray. Tulip: a software toolbox for receding horizon temporal logic planning. In *International Conference on Hybrid Systems: Computation and Control*, 2011.

[8] Christopher A Hane, Cynthia Barnhart, Ellis L Johnson, Roy E Marsten, George L Nemhauser, and Gabriele Sigismondi. The fleet assignment problem: Solving a large-scale integer program. *Mathematical Programming*, 70(1):211–232, 1995.

[9] Ailsa H Land and Alison G Doig. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, pages 497–520, 1960.

[10] Rica Gonen and Daniel Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *ACM Conference on Economics and Computation (EC)*, 2000.

[11] Kaj Holmberg and Di Yuan. A lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–481, 2000.

[12] Stéphane Ross, Geoffrey Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[13] Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[14] He He, Hal Daume III, and Jason M Eisner. Learning to search in branch and bound algorithms. In *NIPS*, 2014.

[15] András Prékopa. The use of discrete moment bounds in probabilistic constrained stochastic programming models. *Annals of Operations Research*, 85:21–38, 1999.

[16] Ro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. A supervised machine learning approach to variable branching in branch-and-bound. In *European Conference on Machine Learning (ECML)*, 2014.

[17] Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2058–2066, 2015.

[18] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning (ICML)*, 2005.

[19] Tobias Achterberg. Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

Supplementary Material

A MILP formulation of risk-aware path planning

This section describes the MILP formulation of risk-aware path planning solved in Section 6. Our formulation is based on the MILP-based path planning originally presented by [5], combined with risk-bounded constrained tightening [15]. It is a similar formulation as that of the state-of-the-art risk-aware path planner pSulu [6] but without risk allocation.

We consider a path planning problem in $\mathbb{R}$, where a path is represented as a sequence of N waypoints $x_1, \cdots x_N \in X$. The vehicle is governed by a linear dynamics given by:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k \\ u_k &\in U, \end{aligned}$$

where $U \subset \mathbb{R}^m$ is a control space, $u_k \in U$ is a control input, $w_k \in \mathbb{R}^n$ is a zero-mean Gaussian-distributed disturbance, and A and B are n-by-n and n-by-m matices, respectively. Note that the dynamic of the mean and covariance of x_i, denoted by $\bar{x}_i$ and Σ_i, respectively, have a deterministic dynamics:

$$\begin{aligned} \bar{x}_{k+1} &= A\bar{x}_k + Bu_k + w_k \\ \Sigma_{k+1} &= A\Sigma A^T + W, \end{aligned} \tag{1}$$

where W is the covariance of w_k. We assume there are M polygonal obstacles in the state space, hence the following linear constraints must be satisfied in order to be safe (as in Figure 3):

$$\bigwedge_{k=1}^{N} \bigwedge_{i=1}^{M} \bigvee_{j=1}^{L_i} h_{ij} x_k \leq g_{ij},$$

where $\bigwedge$ is conjunction (i.e., AND), $\bigvee$ is disjunction (i.e., OR), L_i is the number of edges of the i-th obstacle, and h_{ij} and g_{ij} are constant vector and scaler, respectively. In order for each of the linear constraints to be satisfied with the probability of $1-\delta_{kij}$, the following has to be satisfied:

$$\bigwedge_{k=1}^{N} \bigwedge_{i=1}^{M} \bigvee_{j=1}^{L_i} h_{ij} \bar{x}_k \leq g_{ij} - \Phi(\delta_{kij}) \tag{2}$$

$$\Phi(\delta_{kij}) = -\sqrt{2h_{ijk}\Sigma_{x,k}h_{ijk}^T}\,\mathrm{erf}^{-1}(2\delta_{ijk}-1),$$

where erf^{-1} is the inverse error function.

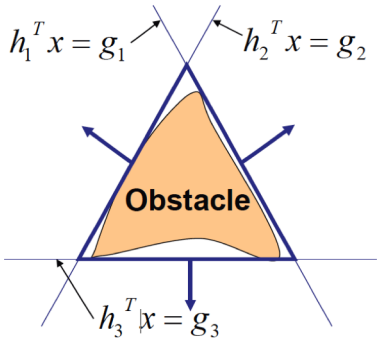


Figure 3: Representation of polygonal obstacle by disjuctive linear constraints

The problem that we solve is, given the initial state $(\bar{x}_0, \Sigma_0)$, to find $u_1 \cdots u_N \in U$ that minimizes a linear objective function and satisfies (1) and (2). An arbitrary nonlinear objective function can be approximated by a piecewise linear function by introducing integer variables. The disjunction in (2) is also replaced by integer variables using the standard Big M method. Therefore, this problem is equivalent to MILP. In the branch-and-bound algorithm, the choice of which linear constraint to be satisfied among the disjunctive constraints in (2) (i.e., which side of the obstacle x_k is) corresponds to which branch to choose at each node.

B Risk-aware Planning Dataset Generation

We generate 150 obstacle maps. Each map contains 10 rectangle obstacles, with the center of each obstacle chosen from a uniform random distribution over the space $0 \leq y \leq 1$, $0 \leq x \leq 1$. The side length of each obstacle was chosen from a uniform distribution in range $[0.01, 0.02]$ and the orientation was chosen from a uniform distribution between 0° and 360°. In order to avoid trivial infeasible maps, any obstacles centered close to the destination are removed.

C Exploration Strategy

For self-imitation learning to succeed in scaling up to larger problem instances, it is important to enable exploration strategies in the search process. In our experiments, we have found the following two strategies to be most useful.

- ϵ-greedy strategy allows a certain degree of random exploration. This helps learned policies to discover new terminal states and enables self-imitation learning to learn from a more diverse goal set. Discovering new terminal states is especially important when scaling up because the learned policies are trained for a smaller problem size; to counter the domain shift when scaling up, we add exploration to enable the learned policies to find better solutions for the new larger problem size.
- Searching for multiple terminal states and choosing the best one as the learning target. This is an extension to the previous point since by comparing multiple terminal states, we can pick out the one that is best for the policy to target, thus improving the efficiency of learning.
- When scaling up, for the first training pass on each problem scale, we collect multiple traces on each data point by injecting 0.05 variance Gaussian noise into the regression model within the policy class, before choosing the best feasible solution.

An Adversarial Regularisation for Semi-Supervised Training of Structured Output Neural Networks

Mateusz Koziński[2] **Loïc Simon**[1] **Frédéric Jurie**[1]

[1]Groupe de recherche en Informatique, Image, Automatique et Instrumentation de Caen Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
[2]CVLab, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

mateusz.kozinski@epfl.ch loic.simon@ensicaen.fr frederic.jurie@unicaen.fr

Abstract

We propose a method for semi-supervised training of structured-output neural networks. Inspired by the framework of Generative Adversarial Networks (GAN), we train a discriminator to capture the notion of a 'quality' of network output. To this end, we leverage the qualitative difference between outputs obtained on labelled training data and unannotated data. The discriminator serves as a source of error signal for unlabelled data. Initial experiments in image segmentation demonstrate that including unlabelled data with the proposed loss function into the training procedure enables attaining the same network performance as in a fully supervised scenario, while using two times less annotations.

1 Introduction

We propose an approach to semi-supervised training of structured output neural networks that enables saving significant labelling effort. We show that the performance of a network trained in a fully supervised regime on a certain amount of labelled data can be matched by training on a significantly smaller amount of labelled data, combined with a volume of unlabelled data in the proposed semi-supervised setting.

Our technical contribution consists in generating a useful error signal for unlabelled data by means of adversarial training. During training, both the labelled and the unlabelled data is forwarded through the network. The network produces qualitatively better output on the labelled images than on the unlabelled ones. Much like in GAN training, we train a discriminator network to capture this difference. The negative gradient of the discriminator with respect to its input is used as the error signal for the unlabelled data. Contrary to pre-training, our method can be applied to any structured output problem, and enables using unlabelled data to train the complete network, end-to-end, independently of its architecture.

2 Related work

The most common methods of handling limited availability of training data include training a neural network on an auxiliary task, for which large volume of data is available, and fine tuning it on a small amount of application-specific data. In self-supervised methods the auxiliary task consists in applying a perturbation to an image, like masking image regions [14], or shuffling image tiles [3, 13], and training the network to recover the original image. In case of an autoencoder [6, 11, 20, 16, 22, 21], the auxiliary task is to encode an image into a strongly regularized latent representation, and reconstruct the original image from the latter. From the perspective of structured prediction, the common

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

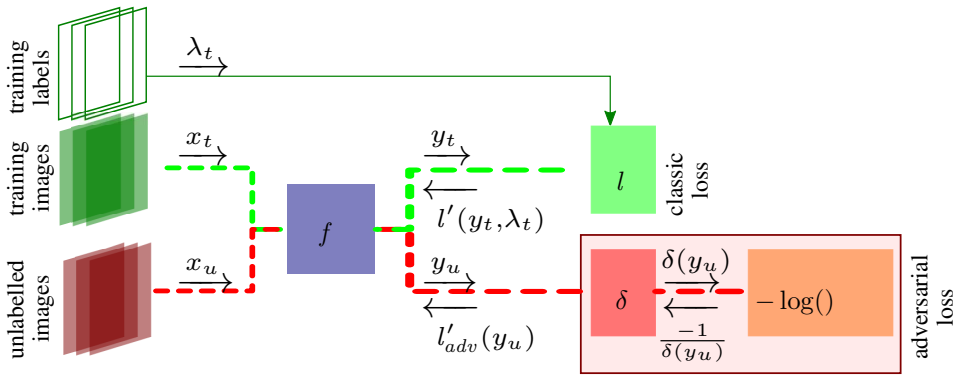


Figure 1: The flow of data and error signals when training a structured output network f with the proposed method, presented in algorithm 1. The discriminator update is not shown in the drawing. The green line denotes the flow of labelled training data and the corresponding gradients. The red line denotes the flow of unlabelled data and the corresponding gradients.

drawback of the above methods is the constraint on the network architecture. For example, only the first half of a typical segmentation network [12, 2] can be matched to an encoder of an autoencoder. In consequence, the remaining layers do not benefit from the unlabelled data. Moreover, structured output problems are characterized by correlations between output variables, and these cannot be learned by a method that is never exposed to data from the output domain. Our method is free from these limitations, enabling end-to-end training of structured-prediction networks.

Our work is inspired by the Generative Adversarial Networks (GANs) [5]. In GAN, a generator network is trained to transform a random vector drawn from a simple sampling distribution to a sample from a complicated target distribution. The flagship application is to train the generator to yield realistically looking images. The key property of GANs is that all that is required for training the generator is a collection of samples from the target distribution. The error signal is backpropagated to the generator from a discriminator network, trained to differentiate samples from the target distribution from ones output by the generator. Previous applications of GANs to semi-supervised learning focus on re-using the discriminator for feature extraction [15], generating additional training images [18], or regularizing the predictions on the generated data to have low confidence [23]. We propose to use the *gradient* of the discriminator as opposed to its outputs or weights.

GANs can also be used for mapping between two domains of interest [7]. In this case discrimination is performed between pairs of input and output items. This type of loss has been demonstrated to boost segmentation results when combined with a standard cost function [9]. In contrast to these supervised methods, we use a discriminator to generate a training signal for *unlabelled* data.

For domain adaptation [4, 8], the discriminator is trained to differentiate between features obtained for samples from two different domains, like synthetic and real images. It is then a source of an error signal, that makes the network invariant to the inter-domain shift. In contrast, in our method, the discriminator regularizes the network with use of unlabelled data from the same domain.

3 Method description

We address the problem of training a structured output network f_w, parametrised with a weight vector w, to produce predictions y on input data x. In our scenario, in addition to the training examples x_t, $t \in \mathcal{T}$, with annotations λ_t, a volume of unlabelled examples x_u, for $u \in \mathcal{U}$, is available. To handle the unlabelled data, we combine a classic supervised loss $l(y_t, \lambda_t)$, measuring the consistency of $y_t = f(x_t)$ and λ_t, with a novel and unsupervised loss $l_{adv}(y_u)$

$$C_{tot}(w) = C(w) + \alpha C_{adv}(w) = \mathbb{E}[l(f_w(x_t), \lambda_t)] + \alpha \mathbb{E}[l_{adv}(f_w(x_u))]\,, \tag{1}$$

where α is a constant. Training consists in determining the optimal network parameter by solving

$$w^* = \arg\min_w C_{tot}(w). \tag{2}$$

We describe the unsupervised cost C_{adv} in section 3.1 and the training algorithm in section 3.2.

3.1 Adversarial loss

Training a network on the labelled data (x_t, λ_t), $t \in \mathcal{T}$ results in a qualitative difference between outputs $y_t = f_w(x_t)$ and outputs produced for the unseen data $y_u = f_w(x_u)$, $u \in \mathcal{U}$. Ideally, x_t and x_u are identically distributed, so one might think the same holds for $f_w(x_t)$ and $f_w(x_u)$. In practice, the dimensionality of x is typically large and the training set is not representative of the variability of the unseen data. This biases f_w to perform better on the training examples. We leverage this difference to define the unsupervised cost C_{adv} as a regularisation term that tends to close this gap.

Inspired by GANs, we propose to train a discriminator network δ_v, parametrised by v, to capture the qualitative difference between y_t for $t \in \mathcal{T}$, and y_u, for $u \in \mathcal{U}$. We interpret the discriminator output $\delta_v(y)$ as the likelihood that y has been obtained from an element of the labelled training set, and we interpret $1 - \delta_v(y)$ as the likelihood of y originating from the unlabelled set. The optimal parameter of the discriminator $v^* = \arg\min_v CE_{disc}(v)$ is defined in terms of the cross-entropy

$$CE_{disc}(v) = -\mathbb{E}_{t\in\mathcal{T}}[\log(\delta_v(y_t))] - \mathbb{E}_{u\in\mathcal{U}}[\log(1 - \delta_v(y_u))]. \tag{3}$$

The negative logarithm of the output of the optimal discriminator can be used as a 'quality measure' for image segmentations. Indeed, $\delta_{v^*}(y)$ is the likelihood that y originates from the training set, and the outputs on the training set are qualitatively better. We therefore define the unsupervised cost as

$$C_{adv}(w) = \mathbb{E}_{u\in\mathcal{U}}[-\log(\delta_{v^*}(f_w(x_u)))]. \tag{4}$$

Minimising (4) with respect to w drives f_w towards reducing the gap between performance on labelled and unlabelled data.

3.2 Algorithm

The minimization can be performed with a gradient-based optimization routine, for example SGD. The gradient of the objective consists of two components and its estimate on a training batch T and unlabelled batch U can be denoted as

$$\nabla_w C^{TU}_{tot}(w) = \nabla_w C^T(w) + \alpha \nabla_w C^U_{adv}(w). \tag{5}$$

Likewise, we denote a batch estimate of the cross entropy gradient (3) by $\nabla_v CE^{TU}_{disc}(v)$. The component gradients can be computed by backpropagation. The flow of data and gradients forward and back through the networks is depicted in Figure 1. In practice, we train the network using algorithm 1. The $\mathrm{update}(w, g)$ procedure accepts the network weights w and a gradient of the cost function g and performs an update on w. While we used SGD with momentum, any update rule used for training neural networks is applicable. Instead of training the discriminator to optimality at each iteration, we perform k updates of the discriminator for a single update of the network f_w itself. There is no guarantee of convergence of the algorithm. However, our experiments demonstrate its practical utility.

Algorithm 1 Training a structured output network with adversarial cost for unlabelled data

$v, w \leftarrow \mathrm{randInit}()$
while not converged **do**
  for $IterNum = 1$ to k **do**
    $T \leftarrow \mathrm{pickBatch}(\mathcal{T}, batchSize)$
    $U \leftarrow \mathrm{pickBatch}(\mathcal{U}, batchSize)$
    $g \leftarrow \nabla_v CE^{TU}_{disc}(v)$ ▷ backpropagation of batches T and U through the discriminator
    $v \leftarrow \mathrm{update}(v, g)$
  end for
  $T \leftarrow \mathrm{pickBatch}(\mathcal{T}, batchSize)$
  $g_T \leftarrow \nabla_w C^T(w)$ ▷ standard backpropagation
  $U \leftarrow \mathrm{pickBatch}(\mathcal{U}, batchSize)$
  $g_U \leftarrow \alpha \nabla_w C^U_{adv}(w)$ ▷ backpropagation through the discriminator and the network
  $w \leftarrow \mathrm{update}(w, g_T + \alpha g_U)$
end while

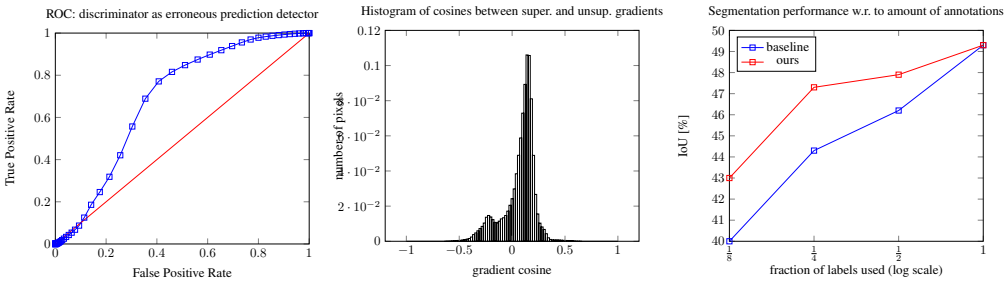


Figure 2: Left: The ROC of discriminator as a detector of erroneous predictions. Middle: The histogram of cosines between error signals originating from a discriminator and ones originating from a cross-entropy loss function. Right: The IoU attained by segnet-basic on the CamVid dataset with respect to the number of annotations used for training. See section 4 for details.

4 Experimental evaluation

Due to space limitations, we only report the key aspects of the experiments. We refer the reader to our online report for technical details.

To evaluate the adversarial regularization we reproduce the road scene segmentation setup of Badrinarayanan, Kendall and Cipolla [2]. It consists of the CamVid dataset and the segnet-basic neural network. It has 367 training images captured by a forward-looking vehicle-mounted camera, annotated in terms of 11 semantic classes. The segnet-basic architecture has a symmetric encoder-decoder architecture, with max pooling-unpooling coupling between the corresponding layers of the encoder and decoder. In total, the network has eight convolutional layers, each with 64 filters.

Our discriminator consists of three blocks of convolution with 64 filters and stride 2, batch normalization and leaky ReLU. It outputs a single variable *per patch* of the input image.

Correlation of the discriminator to prediction error. First, we verify that the discriminator is capable of capturing the quality of the structured output. We train segnet-basic on $\frac{1}{8}$-th of its original training set. Then, we train the discriminator to differentiate the outputs on the training images from ones obtained on the remaining $\frac{7}{8}$ of the original data set. We then plot the ROC curve of the discriminator as a predictor of correct pixel classification on the held out data in figure 2. We conclude that the discriminator output is informative of whether the prediction is correct.

Comparison of discriminator gradient to cross-entropy. To verify the utility of the discriminator gradient, we compare it to the gradient of a standard, fully supervised cross entropy loss function. In the same setup as for the previous experiment, we compute the cosine between these gradients for each pixel of every image held out for training. As can be seen in figure 2, most of the histogram mass accumulates on the positive side of the plot. This suggest that discriminator gradients point to the direction of more accurate predictions more often than not.

Segmentation performance. To check the performance of the adversarial regularization in a realistic application scenario, we compare semi-supervised training with the adversarial loss to fully supervised training, while decreasing the number of annotated training data in the original setup by a factor of 2, 4 and 8. The images excluded from the annotated training set are used as unannotated data for the adversarial regularization.

We train the network by SGD with momentum for 50000 iterations with a decaying learning rate. We jitter the training images. We perform the accuracy tests using the weight vectors after the last update in this procedure, instead of cherry-picking the best models.

We present numerical results in figure 2. When trained on the whole dataset, the baseline attains an accuracy of 49.3% Intersection-over-Union (IoU), exceeding the performance of 47.7% reported in the original paper [2]. Our method consistently outperforms the baseline when trained on a fraction of the dataset. Besides, when labelled data constitutes $\frac{1}{8}$ and $\frac{1}{4}$ of the training set, the regularized network is nearly as good as the baseline trained with twice as many labels.

Table 1: Left: Impact of weight decay and adversarial regularisation on the accuracy of segnet-basic trained on $\frac{1}{8}$-th of the CamVid training set. Right: The accuracy of segnet-basic trained on $\frac{1}{8}$-th of the data set with adversarial regularisation for different discriminator losses.

	weight decay alone						ours
decay factor	0	5e-4	1e-3	5e-3	1e-2	5e-2	1e-3
IoU	38.5	38.5	40.0	40.0	39.5	29.8	43.0

	IoU
baseline - $\frac{1}{8}$	42.3
ours - Cross Entropy	45.1
ours - Wasserstein	47.0

Comparison to weight decay To compare the performance gain due to the adversarial regularisation and weight decay, we train the network according to the protocol used in the previous experiment, on $\frac{1}{8}$ of the original training set, for several values of the weight decay coefficient. We present the results in table 1. Although weight decay leads to improved test performance, the improvement is limited. The adversarial regularisation enables breaking this limit.

Comparison of different discriminator variants We compared different variants of discriminator loss: the standard binary cross entropy (3) and the Wasserstein-GAN-like criterion [1]. We modified the Wasserstein criterion: instead of clipping the discriminator weights, we use weight decay and a hard hyperbolic tangent layer on top of the discriminator. These modifications consistently lead to improved results of our experiments. We changed the experimental setup by switching from SGD to the ADAM optimizer, leading to increased baseline performance. We present the results in table 1. The proposed regularization still improves performance, with the Wasserstein-like discriminator loss yielding significantly better results than the classical cross-entropy-based formulation.

Retinal vein segmentation We test the performance of adversarial regularisation in retinal vein segmentation. We use the DRIVE dataset [19] with 20 training eye fundus images, with binary annotations of retinal blood vessels. We adopt the U-Net symmetric encoder-decoder architecture [17], with skip connections copying feature maps of the encoder and concatenating them to the corresponding decoder feature maps. We modify it by adding batch normalization after each convolution, inserting residual skip connections over each pair of convolutions, and adding dropout after each such block. The network is trained for 20 thousands iterations using ADAM, on randomly cropped and rotated data. We vary the number of annotated training images and compare the results of supervised training to semi-supervised training, where the adversarial loss function is used for unlabelled images. We present the results in table 2. With the complete set of 20 labelled training images our baseline attains the F1 score of 0.819, close to the current state of the art of 0.821 attained by a network pre-trained on ImageNet [10]. Interestingly, with the adversarial regularisation we obtain competitive performance for just 3 training images.

Table 2: The performance of U-Net in retinal vessel segmentation vs. number of labelled training images (F1-score).

# images	baseline	ours
20	0.819	
3	0.785	0.813
1	0.725	0.769

5 Discussion

We proposed a loss function for semi-supervised learning, capable of generating useful error signals based exclusively on predictions. Contrary to the pre-training and co-training approaches, it enables end to end training on unlabelled data, irrespective of the task or network architecture. We have demonstrated that it allows to capitalize on unannotated data, narrowing the performance gap between predictors trained on the fully- and partly-labelled training sets, and enabling training useful predictors on just one annotated image. These advantages come at a cost of computation time and memory needed to train the discriminator network. Moreover, the regularised networks typically took more training iterations to attain their maximum performance. Finally, the presented experiments are performed on small data sets. It remains to be seen if the proposed method yields considerable improvements when hundreds, as opposed to tens, annotated images are available.

Acknowledgement This work was partially funded by the French National Research Agency, grant number ANR-13-CORD-003 (project SEMAPOLIS).

References

[1] ARJOVSKY, M., CHINTALA, S., AND BOTTOU, L. Wasserstein GAN. *CoRR abs/1701.07875* (2017).

[2] BADRINARAYANAN, V., KENDALL, A., AND CIPOLLA, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561* (2015).

[3] DOERSCH, C., GUPTA, A., AND EFROS, A. A. Unsupervised visual representation learning by context prediction. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015* (2015), pp. 1422–1430.

[4] GANIN, Y., USTINOVA, E., AJAKAN, H., GERMAIN, P., LAROCHELLE, H., LAVIOLETTE, F., MARCHAND, M., AND LEMPITSKY, V. S. Domain-adversarial training of neural networks. *CoRR abs/1505.07818* (2015).

[5] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[6] HINTON, G. E., AND SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *Science 313*, 5786 (July 2006), 504–507.

[7] ISOLA, P., ZHU, J.-Y., ZHOU, T., AND EFROS, A. A. Image-to-image translation with conditional adversarial networks. *arxiv* (2016).

[8] JUDY HOFFMAN, DEQUAN WANG, F. Y., AND DARRELL, T. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR abs/1612.02649* (2016).

[9] LUC, P., COUPRIE, C., CHINTALA, S., AND VERBEEK, J. Semantic segmentation using adversarial networks. *CoRR abs/1611.08408* (2016).

[10] MANINIS, K., PONT-TUSET, J., ARBELÁEZ, P. A., AND GOOL, L. J. V. Deep retinal image understanding. *CoRR abs/1609.01103* (2016).

[11] MASCI, J., MEIER, U., CIREŞAN, D., AND SCHMIDHUBER, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks* (2011), Springer, pp. 52–59.

[12] NOH, H., HONG, S., AND HAN, B. Learning deconvolution network for semantic segmentation. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015* (2015), pp. 1520–1528.

[13] NOROOZI, M., AND FAVARO, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI* (2016), pp. 69–84.

[14] PATHAK, D., KRÄHENBÜHL, P., DONAHUE, J., DARRELL, T., AND EFROS, A. A. Context encoders: Feature learning by inpainting. *CoRR abs/1604.07379* (2016).

[15] RADFORD, A., METZ, L., AND CHINTALA, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR abs/1511.06434* (2015).

[16] RANZATO, M., HUANG, F., BOUREAU, Y., AND LECUN, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)* (2007), IEEE Press.

[17] RONNEBERGER, O., FISCHER, P., AND BROX, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR abs/1505.04597* (2015).

[18] SALIMANS, T., GOODFELLOW, I. J., ZAREMBA, W., CHEUNG, V., RADFORD, A., CHEN, X., AND CHEN, X. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain* (2016), pp. 2226–2234.

[19] STAAL, J., ABRAMOFF, M., NIEMEIJER, M., VIERGEVER, M., AND VAN GINNEKEN, B. Ridge based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging 23*, 4 (2004), 501–509.

[20] ZEILER, M. D., TAYLOR, G. W., AND FERGUS, R. Adaptive deconvolutional networks for mid and high level feature learning. In *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011* (2011), pp. 2018–2025.

[21] ZHANG, Y., LEE, K., AND LEE, H. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016* (2016), pp. 612–621.

[22] ZHAO, J., MATHIEU, M., GOROSHIN, R., AND LECUN, Y. Stacked what-where auto-encoders. *CoRR abs/1506.02351* (2015).

[23] ZHENG, Z., ZHENG, L., AND YANG, Y. Unlabeled samples generated by GAN improve the person re-identification baseline in vitro. *CoRR abs/1701.07717* (2017).

Regularizing Prediction Entropy Enhances Deep Learning with Limited Data

Abhimanyu Dubey
MIT
dubeya@mit.edu

Otkrist Gupta
MIT
otkrist@mit.edu

Ramesh Raskar
MIT
raskar@mit.edu

Iyad Rahwan
MIT
irahwan@mit.edu

Nikhil Naik
Harvard University
naik@fas.harvard.edu

Abstract

Many supervised learning problems require learning with small amounts of training data, since constructing large training datasets could be impractical due to cost, labor, or unavailability of data. For such tasks, constructing deep learning approaches that generalize to new data is difficult. In this paper, we demonstrate the effectiveness of using entropy as a regularizer on image classification tasks involving very small amounts of data. Optimizing with entropy regularization enables neural networks to learn more generalizable feature representations in the penultimate layers. We conduct experiments on training from scratch on limited subsets of CIFAR10 and CIFAR100 as well as on fine-tuning from existing models on three datasets for fine-grained visual recognition (FGVC) and observe significant improvements in classification performance on both tasks.

1 Introduction

A plethora of machine learning problems across scientific domains involve very small amounts of training data. Applications of deep learning in medical imaging [15], fine-grained recognition [16, 14] and domain adaptation [3] have training data that is orders of magnitude lower than traditional image classification datasets such as ImageNet [5] or Places365 [27] which have thousands of training samples for every output class.

The difficulty in obtaining annotated training samples for such tasks cannot necessarily be mitigated easily; in applications such as medical image classification and fine-grained recognition, obtaining annotations is expensive and requires domain experts [14]. In addition, due to concerns around privacy (in medical imaging) or the inability to photograph certain fine-grained categories, obtaining unlabeled raw samples in large amounts is in itself impractical.

The effectiveness of large-scale deep convolutional neural networks across tasks in computer vision [13, 23] have made deep learning the *de facto* choice for image classification. Apart from their high computational complexity, another issue of concern is the requirement of high amounts of training data for generalizable performance. Despite the incredible success of large CNNs trained on ImageNet as generic image feature representations [6, 22], these networks do not naively generalize well to tasks involving limited amounts of training data.

A common approach in such classification problems is to initialize a model on weights obtained by training on a large corpus such as ImageNet, and *fine-tuning* the model on the target task. This approach is preferred for such problems, with extensive analysis on techniques to improve fine-tuning performance [4, 25]. Domain-specific neural network architectures have been designed for some

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

of these problems, such as Bilinear Pooling for fine-grained visual recognition [14], and D-CAN for histology image segmentation [2]. Pereyra *et al.* [19] experimented with the idea of penalizing deep neural network classifiers for confident predictions. While they demonstrate improvements in performance across several tasks, these improvements—especially on image classification—are negligible (0.1% average improvements, which is within the standard deviation of the accuracy across trials).

Penalizing overconfidence, is an interesting proposition, but applicable more strongly when limited training data is present. In the absence of densely sampled data points, the training data is more prone to have sampling biases and may not be representative of the underlying distribution. Hence, it is reasonable to prevent the neural network classifier from making overconfident predictions in this case.

In this paper, we establish the effectiveness of regularizing the entropy of the output predictions (a measure for classifier overconfidence) on image classification problems with limited amounts of training data. We observe that as training data increases, the benefits obtained from this penalty diminish, in accordance with our results. Contrary to Pereyra *et al.* [19], our demonstrated improvements are larger, substantiating the applicability of entropy-based regularization in data-constrained classification tasks.

2 Method

2.1 Motivation

The motivation behind the formulation for entropy regularization stems from penalizing the peakiness of the output probability distribution, that is, we require the conditional probability distribution $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ produced as output by the network for an input sample $\boldsymbol{x}$ (under model with parameters θ) to have mass distributed across the alphabet of Y, that is, across several classes. Peaky distributions have lower entropy [8], and it is evident that we obtain maximum entropy when all events are equally likely, i.e. there is no "surprise" in the observation.

Since fine-tuning a model trained on a large dataset on the target dataset is common practice in image classification, the model used for prediction usually has a much higher capacity than the small target dataset it is being fine-tuned on. This practice hence makes it very easy to overfit to the training data. One method of performing regularization for this problem is early-stopping [20], which involves ceasing training once validation performance begins decreasing. While effective, early-stopping is often plagued by the issue deciding when the model has begun to overfit, and ameliorating that requires keeping subsequent copies of the model, which can be memory intensive.

Additionally, if the available labels are corrupted by labeling noise, we cannot allow training until convergence. For applications where the target classes have high similarity (e.g., fine-grained recognition and classification), training with naive cross-entropy would imply that for each sample, there is no noise present in its label, and only the specified label is present with certainty, consequently leading to overfitting when training.

2.2 Formulation

As we specified earlier, a metric to regularize the confidence and increase confusion in output distributions would increase the entropy of the output conditional probability distribution. Entropy $H(p_\theta(\boldsymbol{y}|\boldsymbol{x}))$ for the conditional probability distribution $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ can be given by:

$$H(p_\theta(\boldsymbol{y}|\boldsymbol{x})) = -\sum_{i=1}^{N} p_\theta(\boldsymbol{y}_i|\boldsymbol{x}) \cdot \log(p_\theta(\boldsymbol{y}_i|\boldsymbol{x})) \tag{1}$$

We can maximize entropy by specifying our learning objective function $\mathcal{L}$ as:

$$\mathcal{L} = \mathcal{L}_{ce}(p_\theta(\boldsymbol{y}|\boldsymbol{x})) - \alpha \cdot H(p_\theta(\boldsymbol{y}|\boldsymbol{x})) \tag{2}$$

Where $\mathcal{L}_{ce}$ denotes the prevalent cross-entropy loss. The new functional $\mathcal{L}$ promotes learning a classifier with a maximum entropy output distribution (under suitable regularization parameter α), while maintaining classification accuracy.

Another interpretation of the same objective function can be drawn from measuring the divergence of the conditional probability distribution from the uniform distribution. One frequently used measure for estimating this metric, that can also be employed in the training of neural networks with cross-entropy, is the Kullback-Leibler(KL) divergence [10]. The asymmetry of this metric gives rise to two forms of regularization. If we write the KL-divergence of $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ from the uniform distribution with mean $\frac{1}{N}$, denoted as $U(\frac{1}{N})$, we get:

$$\mathbb{D}_{\mathsf{KL}}(p_\theta(\boldsymbol{y}|\boldsymbol{x}) \parallel U(\frac{1}{N})) = \sum_{i=1}^{N} p_\theta(\boldsymbol{y}_i|\boldsymbol{x}) \cdot \log\Big(\frac{p_\theta(\boldsymbol{y}_i|\boldsymbol{x})}{N^{-1}}\Big) \tag{3}$$

$$= \log N + \sum_{i=1}^{N} p_\theta(\boldsymbol{y}_i|\boldsymbol{x}) \cdot \log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}) \tag{4}$$

$$= \log N - H(p_\theta(\boldsymbol{y}|\boldsymbol{x})) \tag{5}$$

Hence, we can see that maximizing the entropy H is identical to minimizing one direction of KL-divergence of the output conditional probability distribution from the uniform distribution. Reversing the direction of the divergence yields the label-smoothing regularization (LSR) [24], a technique which involves altering the one-hot label vector to a smoother version, by replacing the mass at the incorrect classes with $\frac{1}{N}$ instead of zero.

Label Smoothing Regularization provides small increases in performance [24], which are limited since it confuses the classifier with all classes equally. Regularizing entropy, however, confuses the classifier with its own predictions, which is informative in situations where subsets of classes are confusing, since we would want to penalize the classifier for predictions only within a subset. We argue that the mean-seeking nature of LSR omits modes in the distribution that are captured by the entropy formulation [17], i.e. we can expect an entropy regularized network to bootstrap from its own predictions in the presence of label noise, as successfully utilized by Reed *et al*. [21]. Our final objective for a batch of b training samples can be given by:

$$\mathcal{L} = \sum_{i=1}^{b} \Big(\mathcal{L}_{ce}(p_\theta(\boldsymbol{y}|\boldsymbol{x}^{(i)}), \boldsymbol{y}^{(i)}) - \frac{\alpha}{b} \cdot H(p_\theta(\boldsymbol{y}|\boldsymbol{x}^{(i)}))\Big) \tag{6}$$

3 Experiments

We continue with the formulation described in Equation 6. We experiment with implementations in popular libraries of Caffe [9] and PyTorch [18], over a cluster of NVIDIA TITAN X and GTX 1080 GPUs. We design experiments specifically to support our claim on deep learning with limited training data. We evaluate on a variety of deep learning model architectures, including AlexNet [13], VGGNet-16 [23], GoogLeNet [24] and ResNets [7]. We select the hyperparameter α via cross-validation, and present a short analysis of the effect of variation and selection of the hyperparameter on prediction performance as well.

3.1 Limited Data CIFAR-10 and CIFAR-100

Our first set of experiments include the classic image classification dataset of CIFAR-10 [12] and CIFAR-100 [12]. CIFAR-10 has 10 target classes, with 5000 samples per class, and CIFAR-100 consists of 100 classes and 500 samples per class. It is critical to note that the number of training samples per class for CIFAR-10 is much higher than that even of ImageNet [5], and hence we experiment with very small fractions of the dataset, in order to match dataset sizes present in domains with limited training data. We progressively experiment training models with randomly selected subsets of the training data and observing the gain in test performance.

As hypothesized, we observe an increase in validation accuracy with limited amounts of training data in both cases, that reduces as the amount of training data available increases. In case of CIFAR-10, we observe no significant performance increase (consistent with [19]) when trained using the entire training dataset, however, as depicted in Table 1, for small amounts of training data (1% to 10%) we observe a performance increase as large as 6%. These improvements are present across several model architectures (see Figure 1a and Table 1). We also observe that the gains reduce when using complete training data—with no significant improvement in CIFAR-10 and a gain of 1% to 3% on

Method	Accuracy(%) on CIFAR-10					Accuracy(%) on CIFAR-100				
	1%	2%	5%	10%	100%	1%	2%	5%	10%	100%
ResNet20 [7]	29.98	35.19	43.54	51.25	92.34	19.45	25.78	33.91	36.16	75.81
ResNet20 (**E**)	32.75	39.95	45.90	54.89	92.57	21.06	27.35	36.08	42.12	77.40
VGGNet16 [23]	30.18	36.02	41.78	50.21	92.06	18.19	22.35	30.16	35.46	73.81
VGGNet16 (**E**)	33.51	40.68	46.11	55.62	92.17	20.05	23.39	34.08	40.20	75.08
GoogLeNet [24]	26.15	33.57	38.10	46.18	84.16	17.90	20.38	25.34	30.19	70.24
GoogLeNet (**E**)	30.14	36.15	42.25	50.19	84.19	19.17	23.65	28.86	34.01	73.15

Table 1: The impact of adding entropy regularization on classification for random subsets of CIFAR-10 and CIFAR-100 datasets. The models trained with entropy regularization are displayed with the tag (**E**).

CIFAR-100. This can be attributed to the fact that as more training data becomes available, the training data approximates the validation data more precisely, alleviating the need to account for confusion (or penalize peakiness).

3.2 Fine-Grained Visual Classification

Fine-Grained Visual Classification (FGVC) is an important problem in computer vision, which involves distinguishing between object classes with substantially higher visual similarity compared to those in large-scale image classification tasks. Some examples of FGVC include differentiating between species of birds, flowers and animals; or the brands and models of vehicles. These tasks depart from conventional image classification in that they require expert knowledge, rather than crowdsourcing, for gathering annotations. Additionally for fine-grained wildlife data collection, several species are generally harder to photograph, resulting in long tails in the data distribution. Owing to the difficulty in capturing and annotating samples, most FGVC datasets are orders of magnitude smaller than traditional image classification datasets, making it harder to learn deep learning classifiers on such data.

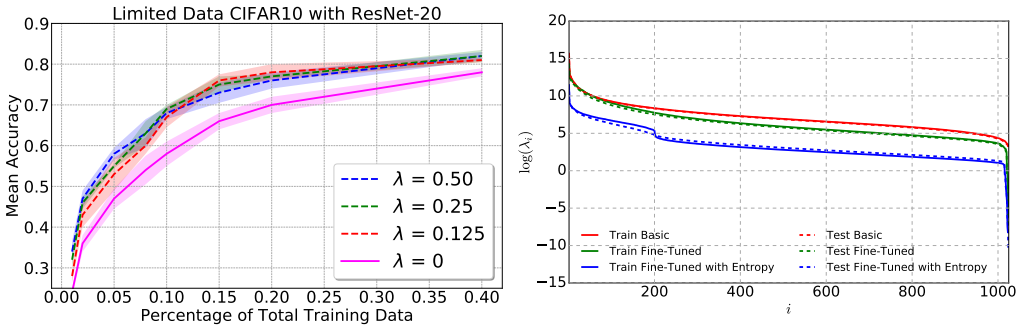
Method	Accuracy(%) on FGVC Dataset		
	CUB-2011 [26]	Stanford Dogs [11]	NABirds [1]
GoogLeNet [24]	68.19 ($\pm$ 0.14)	55.76 ($\pm$ 0.08)	70.66 ($\pm$ 0.28)
GoogLeNet (**E**)	74.37 ($\pm$ 0.19)	61.98 ($\pm$ 0.16)	71.15 ($\pm$ 0.19)
VGGNet16 [23]	73.30 ($\pm$ 0.22)	61.87 ($\pm$ 0.25)	68.29 ($\pm$ 0.31)
VGGNet16 (**E**)	77.91 ($\pm$ 0.17)	65.56 ($\pm$ 0.23)	72.66 ($\pm$ 0.23)
ResNet50 [7]	76.15 ($\pm$ 0.18)	69.92 ($\pm$ 0.16)	63.55 ($\pm$ 0.15)
ResNet50 (**E**)	81.24 ($\pm$ 0.28)	74.31 ($\pm$ 0.31)	70.84 ($\pm$ 0.18)
BilinearCNN [14]	84.10 ($\pm$ 0.12)	82.13 ($\pm$ 0.22)	80.90 ($\pm$ 0.16)
BilinearCNN (**E**)	84.93 ($\pm$ 0.15)	83.04 ($\pm$ 0.14)	81.14 ($\pm$ 0.12)

Table 2: The impact of adding entropy regularization on classification for fine-grained visual classification datasets. All models have been fine-tuned from their publicly available ImageNet-trained weights. The models trained with entropy regularization are displayed with the tag (**E**).

Our results for FGVC tasks are summarized in Table 2. We observe gains larger than the previous set of experiments, especially on generic models such as GoogLeNet [24], VGGNet16 [23] and ResNet50 [7]. We additionally observe an improvement on specialized FGVC deep models, such as Bilinear-CNN [14] across 3 datasets (CUB-2011 [26], Stanford Dogs [11], NABirds [1]).

4 Analysis and Conclusion

In this paper, we demonstrated the impact of regularizing entropy for tasks involving learning with limited amounts of training data. Since we are adding a regularizer, it would be critical to understand the variation of performance with variation in the hyperparameter. We find that for smaller amounts of data present, larger values of α provide benefits in classification, but overall we find that the algorithm is largely insensitive to different values of α in the range of 0 to 1. This variation in performance is summarized for CIFAR-10 in Figure 1a.



(a) (b)

Figure 1: (a) Analysis of variation of classification performance as training data is increased, plotted for various values of α, on CIFAR10 with model ResNet20. (b) Eigenvalue decomposition of covariance (unnormalized PCA) of penultimate layer GoogLeNet features for both training and test sets of CUB2011. We plot the value of $\log(\alpha_i)$ for the ith eigenvalue α_i obtained after decomposition of test set (dashed) and training set (solid) on three models.

Subsequently, it is also interesting to visualize what the effect of regularizing entropy is on the underlying feature maps. Adding entropy to the classifier will encourage the classifier to reduce the specificity of the features, since we discourage peakiness in the output distribution. To evaluate this hypothesis, we perform the eigendecomposition of the covariance matrix (unnormalized PCA) on the penultimate layer features of GoogLeNet trained on CUB-2011, and analyze the trend of sorted eigenvalues (Figure 2b). We examine the features obtained from a network with (i) no fine-tuning ("Basic"), (ii) fine-tuning without regularization, and (iii) fine-tuning with entropy regularization.

For a feature matrix with large covariance between the features of different classes, we would expect the first few eigenvalues to be large, and the rest to diminish quickly, since fewer orthogonal components can summarize the data. Conversely, in a completely uncorrelated feature matrix, we would see a larger tail in the decreasing magnitudes of eigenvalues. Figure 1b shows that for the Basic features (with no fine-tuning), there is a fat tail in both training and test sets due to the presence of a large number of uncorrelated features. After fine-tuning on the training data, we observe a reduction in the tail of the curve, implying that some generality in features has been introduced in the model through the fine-tuning. The test curve follows a similar decrease, justifying the increase in test accuracy. Finally, for entropy regularization, we observe a substantial decrease in the width of the tail of eigenvalue magnitudes, suggesting a larger increase in generality of features in both training and test sets, which confirms our hypothesis.

In conclusion, we demonstrate the effectiveness of regularizing entropy when training deep neural network models with limited amounts of training data. Our work should be useful in improving generalization performance of the large number of applied computer vision tasks that utilize deep neural networks for training with small amounts of data, including medical imaging, fine-grained recognition, and domain adaptation.

Acknowledgements: The authors would like to thank Ryan Farrell and Pei Guo for their helpful comments and discussions.

References

[1] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

[2] Hao Chen, Xiaojuan Qi, Lequan Yu, Qi Dou, Jing Qin, and Pheng-Ann Heng. Dcan: Deep contour-aware networks for object instance segmentation from histology images. *Medical image analysis*, 36:135–146, 2017.

[3] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains.

[4] Brian Chu, Vashisht Madhavan, Oscar Beijbom, Judy Hoffman, and Trevor Darrell. Best practices for fine-tuning visual classifiers to new domains. In *Computer Vision–ECCV 2016 Workshops*, pages 435–442. Springer, 2016.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[8] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[10] James M Joyce. Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*, pages 720–722. Springer, 2011.

[11] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs.

[12] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset, 2014.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[15] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *arXiv preprint arXiv:1702.05747*, 2017.

[16] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[17] Tom Minka et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

[18] Adam Paskze and Soumith Chintala. Tensors and Dynamic neural networks in Python with strong GPU acceleration. `https://github.com/pytorch`. Accessed: [August 1, 2017].

[19] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*, 2017.

[20] Lutz Prechelt. Early stopping-but when? *Neural Networks: Tricks of the trade*, pages 553–553, 1998.

[21] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[22] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[25] Nima Tajbakhsh, Jae Y Shin, Suryakanth R Gurudu, R Todd Hurst, Christopher B Kendall, Michael B Gotway, and Jianming Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312, 2016.

[26] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[27] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016.

Local Affine Approximators for Improving Knowledge Transfer

Suraj Srinivas & François Fleuret
Idiap Research Institute and EPFL
{suraj.srinivas, francois.fleuret}@idiap.ch

Abstract

The Jacobian of a neural network, or the derivative of the output with respect to the input, is a versatile object with many applications. In this paper we discuss methods to use this object efficiently for knowledge transfer. We first show that matching Jacobians is a special form of distillation, where noise is added to the input. We then show experimentally that we can perform better distillation under low-data settings. We also show that Jacobian-penalty regularizers can be used to improve robustness of models to random noise.

1 Introduction

Deep neural networks are non-linear functions with high expressive power, and given a network architecture, it is straightforward to train models from data. However, what if we are interested in training multiple alternate architectures? How can we use already trained models to accelerate the training of others? Methods that can do this are called knowledge transfer methods. A perfect knowledge transfer method would enable us to seamlessly transform one neural network architecture into another, while preserving input-output mapping and generalization. Such interchangability would allow us to more easily explore the space of neural network architectures. This capability could be used for neural network architecture search, model compression, or creating diverse ensembles, among other applications.

This paper deals with the problem of knowledge transfer using a first-order approximation of the neural network. This approach has also been recently explored by Czarnecki *et al.*[1], who considered the general idea of matching Jacobians, and by Zagoruyko *et al.*[2] who proposed knowledge transfer by viewing Jacobians as attention maps. In both of these cases, it was not clear what are the right kinds of penalty to impose on Jacobians. It was also unclear how these methods relate to previous knowledge transfer approaches such as knowledge distillation [3, 4].

In this paper, we show that we can train alternate architectures using much smaller number of data points than the full training set. Specifically for the CIFAR100 dataset, we are able to reach within 2% of the final generalization error by using only $1/5^{th}$ of the entire dataset. The overall contributions of our paper are: (1) We show that matching Jacobians is a special case of distillation, where noise is added to the inputs, (2) Using these regularizers, we improve knowledge transfer performance, particularly for low-data settings, (3) We show that training neural networks with Jacobian penalty improves stability with respect to noise.

2 Related Work

There is a lot of literature which uses the Jacobian in neural networks. Recently Sobolev training [1], showed that using higher order derivatives along with the targets can help in training with low data. This work is very similar to ours. While we also make similar claims, we clarify the relationship of this method with activation matching (knowledge distillation) and show how it can help. Same

is the case with Zagoruyko *et al.*[2], who introduce the idea of matching attention maps. One possible method they consider was computing the Jacobian. This work also finds that combining both activation matching and jacobian matching is helpful.

Jacobian-based regularizers were used in early works [5], where they looked at penalizing the Jacobian norm. The intuition was to make the model more robust to small changes in the input. We find that this conforms to our analysis as well.

There is also plenty of work relating to knowledge transfer between neural networks. Knowledge Distillation [4] first showed that one can use softmax with temperature to perform knowledge transfer with neural nets. Ba and Caruana [3] found that squared error between logits worked better than the softmax method, and they used this method to train shallow nets with equivalent performance to deep nets. A recent paper [6] also found that adding noise to logits helps during teacher-student training. We show that the use of the Jacobian can be interpreted as adding such noise analytically.

3 Local Affine Approximators of Neural Networks

Let us consider the first order Taylor series expansion of a function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ around a small neighbourhood $\{\mathbf{x} + \Delta\mathbf{x} : \|\Delta\mathbf{x}\| \leq \epsilon\}$. It can be written as

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) + \nabla_x f(\mathbf{x})^T(\Delta\mathbf{x}) + \mathcal{O}(\epsilon^2) \approx f(\mathbf{x}) + \nabla_x f(\mathbf{x})^T(\Delta\mathbf{x}) \tag{1}$$

This is the local affine approximation of the function $f(\cdot)$ around the local neighbourhood. This approximation is useful to analyse non-linear objects like deep neural networks. The source of non-linearity for neural nets lie in the elementwise non-linear activations (like ReLU, sigmoid) and pooling operators. *It is easy to see that to construct an affine approximation of the entire neural network, one must construct affine approximations of all such non-linearities in the network.*

Special case: ReLU and MaxPool For the ReLU nonlinearity, the affine approximation is simple to compute, as the derivative $\frac{\mathrm{d}\sigma(z)}{\mathrm{d}z}$ is either 0 or 1 (except at $z = 0$, where it is undefined). Similarly for max-pooling as well. Going back to the definition in Equation 1, for piecewise linear nets there exist $\epsilon > 0$ such that the super-linear terms are exactly zero, *i.e.*; $f(\mathbf{x}+\Delta\mathbf{x}) = f(\mathbf{x})+\nabla_x f(\mathbf{x})^T(\Delta\mathbf{x})$, *i.e.*; $\mathbf{x}$ and $\Delta\mathbf{x}$ lie on the same plane.

Consider piecewise linear functions with the non-differentiability at zero. Also consider neural networks with no external bias units among the model parameters. For such cases we see that the affine approximation reduces to a linear approximation, *i.e.*; the overall 'bias' term is zero. This means the following: let $\mathbf{x}_0$ be a point arbitrarily close to zero such that $f(\mathbf{x}_0+\Delta\mathbf{x}) = f(\mathbf{x}_0)+\nabla_x f(\mathbf{x}_0)^T(\Delta\mathbf{x})$. Then, given $f(\mathbf{x}_0) \rightarrow 0$, $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0 \rightarrow \mathbf{x}$ and the fact that $\mathbf{x}$ and $\mathbf{x}_0$ have the same Jacobian, we have the following - $f(\mathbf{x}) = \nabla_x f(\mathbf{x})^T\mathbf{x}$. *In other words, the output can be obtained by multiplying the Jacobian with the input*. We find that this relation holds approximately even for ReLU nets with a bias unit as they are typically quite small (*e.g.* VGG19). Thus in this case the Jacobian captures almost all of the information required to make a decision. However, this breaks down for ReLU nets with batch normalization (*e.g.* ResNets) as they introduce their own mean subtraction terms.

As a corollary for other non-linearities, even with no external bias units, the overall 'bias' may be non-zero. For example, consider the hard-tanh nonlinearity, which is given by $\sigma(z) = z$ for $-1 \leq z \leq 1$, and saturates to -1 on the left and $+1$ on the right. In such a case, when the non-linearity saturates, the corresponding slope is zero and we get $\sigma(z + dz) = \sigma(z) = \pm 1$ bias depending on where it saturates. This reminds us that in general the Jacobian does not capture all information about the neural network, especially in the case of such saturating non-linearities, or when external bias units are added.

4 Knowledge Transfer

In this section, we consider the problem of knowledge transfer using Local Affine Approximations. This problem may be posed as follows: given a *teacher* network $\mathcal{T}$ which is trained on a dataset $\mathcal{D}$, we wish to enhance the training of a student network $\mathcal{S}$ on $\mathcal{D}$ using hints from $\mathcal{T}$. Recent works [1, 2] sought to match the Jacobians of $\mathcal{S}$ and $\mathcal{T}$ via squared error terms. However, two aspects are not

clear in these formalisms: (i) what penalty term must be used between Jacobians, and (ii) how this idea of matching Jacobians relates to simpler methods such as activation matching [3, 4]. Recall that previous distillation works involved matching the activations of $\mathcal{S}$ and $\mathcal{T}$. To resolve these issues, we make the following claim.

Claim. *Matching Jacobians of two networks is equivalent to matching soft targets with noise added to the inputs during training.*

More concretely, we make the following proposition.

Proposition 1. *Consider the squared error cost function for matching soft targets of two neural networks with k-length targets ($\in \mathbb{R}^k$), given by $\ell(\mathcal{T}(\mathbf{x}), \mathcal{S}(\mathbf{x})) = \sum_{i=1}^{k}(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}))^2$, where $\mathbf{x} \in \mathbb{R}^D$ is an input data point. Let $\boldsymbol{\xi}$ $(\in \mathbb{R}^D) = \sigma$ $\mathbf{z}$ be a scaled version of a unit normal random variable $\mathbf{z}$ $\in \mathbb{R}^D$ with scaling factor $\sigma \in \mathbb{R}$. Then the following is true.*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}\left(\mathcal{T}^i(\mathbf{x}+\boldsymbol{\xi}) - \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\right)^2\right] \;=\; \sum_{i=1}^{k}\left(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})\right)^2 \;+\; \sigma^2 \sum_{i=1}^{k} \|\nabla_x \mathcal{T}^i(\mathbf{x}) - \nabla_x \mathcal{S}^i(\mathbf{x})\|_2^2 + \mathcal{O}(\sigma^4)$$

Notice that in this expression, we have decomposed the loss function into two components - one representing the usual distillation loss on the samples, and the second regularizer term representing the jacobian matching loss. The final error terms are small for small σ and can be ignored. The above proposition is a simple consequence of using the first-order Taylor series expansion around x. Note that the error term is zero for piecewise-linear nets. An analogous statement is true for the case of cross entropy error between soft targets, leading to:

$$\mathbb{E}_{\boldsymbol{\xi}}\left[-\sum_{i=1}^{k}\mathcal{T}_s^i(\mathbf{x}+\boldsymbol{\xi})\log\left(\mathcal{S}_s^i(\mathbf{x}+\boldsymbol{\xi})\right)\right] \;\approx\; -\sum_{i=1}^{k}\mathcal{T}_s^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x})) - \sigma^2\sum_{i=1}^{k}\frac{\nabla_x \mathcal{T}_s^i(\mathbf{x})^T \nabla_x \mathcal{S}_s^i(\mathbf{x})}{\mathcal{S}_s^i(\mathbf{x})} \tag{2}$$

where $\mathcal{T}_s^i(\mathbf{x})$ denotes the same network $\mathcal{T}^i(\mathbf{x})$ but with a softmax or sigmoid (with temperature parameter T if needed) added at the end. We ignore the super-linear error terms for convenience.

In simple terms, these statements show that matching Jacobians is a natural consequence of matching not only the raw CNN outputs at the given data points, but also at the infinitely many data points nearby. We can use a similar idea to derive regularizers for the case of regular neural network training as well. These regularizers seek to make the underlying model *robust* to small amounts of noise added to the inputs.

Proposition 2. *Consider the squared error cost function for training a neural network with k targets, given by $\ell(y(\mathbf{x}), \mathcal{S}(\mathbf{x})) = \sum_{i=1}^{k}(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}))^2$, where $\mathbf{x} \in \mathbb{R}^D$ is an input data point, and $y^i(\mathbf{x})$ is the i^{th} target output. Let $\boldsymbol{\xi}$ $(\in \mathbb{R}^D) = \sigma$ $\mathbf{z}$ be a scaled version of a unit normal random variable $\mathbf{z}$ $\in \mathbb{R}^D$ with scaling factor $\sigma \in \mathbb{R}$. Then the following is true.*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}\left(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\right)^2\right] \;=\; \sum_{i=1}^{k}\left(y^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})\right)^2 \;+\; \sigma^2\sum_{i=1}^{k}\|\nabla_x \mathcal{S}^i(\mathbf{x})\|_2^2 \;+\; \mathcal{O}(\sigma^4)$$

A statement similar to Proposition 2 has been previously derived by [7], who observed that the regularizer term for linear models corresponds exactly to the well-known Tikhonov regularizer. This regularizer was also proposed by [5]. The ℓ_2 weight decay regularizer for neural networks was derived as an extension of this for neural networks. However, this result shows that a more

appropriate extension would be to penalize the norms of the Jacobian rather than weights directly. We can derive a similar result for the case of cross-entropy error as well, which is given by -

$$\mathbb{E}_{\boldsymbol{\xi}}\left[-\sum_{i=1}^{k} y^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x}+\boldsymbol{\xi}))\right] \approx -\sum_{i=1}^{k} y^i(\mathbf{x})\log(\mathcal{S}_s^i(\mathbf{x})) + \sigma^2\sum_{i=1}^{k} y^i(\mathbf{x})\frac{\|\nabla_x \mathcal{S}_s^i(\mathbf{x})\|_2^2}{\mathcal{S}_s^i(\mathbf{x})^2} \quad (3)$$

We notice here again that the regularizer involves $\mathcal{S}_s^i(\mathbf{x})$, which has the sigmoid / softmax nonlinearity applied on top of the final layer of $\mathcal{S}^i(\mathbf{x})$. Deriving all the above results is a simple matter of using first-order Taylor series expansions, and a second-order expansion in the case of Equation 3.

In all cases above we see that the regularizers for cross-entropy error term seem more unstable when compared to those for squared error. We find that this is true experimentally as well. As a result, we use squared error loss for distillation.

Approximating the Full Jacobian One can see that both in the case of Proposition 1 and 2, we are required to match the full Jacobian. This is computationally very expensive to perform. To reduce the computational load, we can approximate the summation of Jacobian terms with the single largest term. However, we cannot estimate this without computing the Jacobians themselves. As a result, we use a heuristic where the output variable involving the correct answer $c \in [1, k]$ is used for computing the Jacobian. Alternately, if we do not want to use the labels, we may instead use the output variable with the largest magnitude.

5 Experiments

We perform two kinds of experiments to show the effectiveness of using Jacobians. First, we perform knowledge transfer on low-data on CIFAR10 and CIFAR100 datasets. Second, we show that penalizing Jacobian norm increases stability of the networks to random noise.

5.1 Knowledge Transfer

For the knowledge transfer experiments, we use VGG-like architectures with batch normalization. The main difference is that we keep only the convolutional layers and have only one fully connected layer rather than three. Our workflow is as follows. First, a 9-layer 'teacher' network is trained on the full CIFAR10 / CIFAR100 dataset. Then, a smaller 4-layer 'student' network is trained - but this time on small subsets of CIFAR10 / 100 datasets rather than the full dataset. We perform experiments to see how performance of various methods differ on changing the size of the dataset for the student network. Experimental details are given in the supplementary section.

Regarding baselines, we compare against regular training of student network from scratch (Row 1). For the distillation experiments, we compare with and without using labels. Baselines with labels have the suffix '+CE' which stands for 'Cross-Entropy' loss using the ground truth labels. In all cases, we use the squared-error distillation loss shown in Proposition 1. We found that the squared loss worked much better than the cross-entropy loss for distillation as did [3]. The loss function we use for the CIFAR10 case is slightly different from that predicted by theory. We use a form of normalized squared error, whose exact form is specified in the supplementary material. However in our CIFAR100 experiments we use the simple squared error loss.

From both Tables 1, 2 we can conclude that (1) it is generally beneficial to do any form of distillation to improve performance, (2) matching Jacobians along with activations outperforms matching only activations in low-data settings, (3) not matching Jacobians is often beneficial for large data.

One curious phenomenon we observe is that having a cross-entropy (CE) error term is often not crucial to maintain good regularization performance. For CIFAR10, we find that not using this term often yields better results. This is perhaps because the CIFAR10 teacher network has high accuracy (90.94%) when compared to the CIFAR100 teacher (64.78%).

For Table 2, we see that when training with activations, Jacobians and regular cross-entropy training (fourth row), we reach an accuracy of 52.43% when training with 100 data points per class. We observe that the overall accuracy of raw training using the full dataset is 54.28%. This implies that

Table 1: Knowledge transfer performance for the CIFAR10 dataset. We find that matching both activations and Jacobians performs the best for low-data settings. The student network is VGG-4 while the teacher is a VGG-9 network which achieves 90.94% accuracy.

# of Data points per class	5	10	50	100	500	1000	5000 (full)
Regular Cross-Entropy (CE) training	28.92	33.8	53.6	61.05	76.01	77.63	84.56
Match Activations + CE	35.8	40.79	59.98	67.22	80.55	81.77	87.65
Match Jacobians + CE	34.47	39.43	58.18	65.05	76.82	78.21	84.76
Match {Activations + Jacobians} + CE	39.78	43.4	62.17	69.15	80.28	82.23	87.29
Match Activations only	36.73	41.61	60.71	68.32	**82.19**	**84.00**	**88.64**
Match {Activations + Jacobians}	**41.43**	**46.03**	**63.5**	**70.17**	80.98	82.75	85.56

Table 2: Knowledge transfer performance for the CIFAR100 dataset. We find that matching both activations and Jacobians along with cross-entropy error performs the best for low-data settings. The student network is VGG-4 while the teacher is a VGG-9 network which achieves 64.78% accuracy.

# of Data points per class	1	5	10	50	100	500 (full)
Regular Cross-Entropy (CE) training	5.69	13.9	20.03	37.6	44.92	54.28
Match Activations + CE	12.13	26.97	33.92	46.47	50.92	**56.65**
Match Jacobians + CE	6.78	23.94	32.03	45.71	51.47	53.44
Match {Activations + Jacobians} + CE	**13.78**	**33.39**	**39.55**	**49.49**	**52.43**	54.57
Match Activations only	10.73	28.56	33.6	45.73	50.15	56.59
Match {Activations + Jacobians}	13.09	33.31	38.16	47.79	50.06	51.33

we are able to reach close to the full training accuracy using $1/5^{th}$ of the data requirement. For the CIFAR10 dataset, we see that the closest performance is achieved by training with only the activations.

5.2 Noise robustness

We now look at the Jacobian regularizers intended to increase robustness to noise. We train 9-layer VGG networks on CIFAR100 with varying amount of the regularization strength, and measure their classification accuracies in presence of noise added to the normalized images. From Table 3 we find that using higher regularization strengths is better for robustness, even when the initial accuracy at the zero-noise case is lower. This aligns well with our theory.

6 Conclusion

In this paper we have considered the problem of matching local affine approximators of deep neural networks for knowledge transfer. We showed that adding the Jacobian loss term to soft-target loss is equivalent to using the soft-target loss with noisy inputs. We experimentally find that adding such a Jacobian loss term typically improves over using only the distillation loss. We also showed that these regularizers can help in making neural networks more robust to noise.

Table 3: Noise robustness tests on VGG-9 nets trained with the Jacobian-penalty regularizers on the CIFAR100 dataset

Noise standard deviation	0	0.1	0.2	0.3	0.4
Regularization = 0	64.78	61.9 ± 0.07	47.53 ± 0.23	29.63 ± 0.16	17.69 ± 0.17
Regularization = 0.01	64.67	61.85 ± 0.15	49.47 ± 0.07	32.24 ± 0.28	19.63 ± 0.17
Regularization = 0.1	**65.62**	$\mathbf{63.36 \pm 0.18}$	53.57 ± 0.23	37.38 ± 0.18	23.99 ± 0.19
Regularization = 0.5	64.26	62.85 ± 0.05	56.57 ± 0.21	44.99 ± 0.28	31.97 ± 0.15
Regularization = 1.0	63.59	62.66 ± 0.13	$\mathbf{57.53 \pm 0.17}$	$\mathbf{47.48 \pm 0.14}$	$\mathbf{35.43 \pm 0.11}$

References

[1] Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev training for neural networks. *NIPS*, 2017.

[2] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *ICLR*, 2017.

[3] LJ Ba and R Caruana. Do deep networks really need to be deep. *Advances in neural information processing systems*, 27:1–9, 2014.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning Workshop*, 2015.

[5] Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 1992.

[6] Bharat Bhusan Sau and Vineeth N Balasubramanian. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*, 2016.

[7] Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 1995.

Supplementary Material

Proofs

Proposition 1. *Consider the squared error cost function for matching soft targets of two neural networks with k-length targets ($\in \mathbb{R}^k$), given by $\ell(\mathcal{T}(\mathbf{x}), \mathcal{S}(\mathbf{x})) = \sum_{i=1}^{k}(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}))^2$, where $\mathbf{x} \in \mathbb{R}^D$ is an input data point. Let $\boldsymbol{\xi}$ ($\in \mathbb{R}^D$) $= \sigma\ \mathbf{z}$ be a scaled version of a unit normal random variable $\mathbf{z}\ \in \mathbb{R}^D$ with scaling factor $\sigma \in \mathbb{R}$. Then the following is locally true.*

$$\mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}\left(\mathcal{T}^i(\mathbf{x}+\boldsymbol{\xi}) - \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\right)^2\right] = \sum_{i=1}^{k}\left(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})\right)^2 + \sigma^2 \sum_{i=1}^{k}\|\nabla_x \mathcal{T}^i(\mathbf{x}) - \nabla_x \mathcal{S}^i(\mathbf{x})\|_2^2 + \mathcal{O}(\sigma^4)$$

Proof. There exists σ and $\boldsymbol{\xi}$ small enough that first-order Taylor series expansion holds true

$$\begin{aligned} & \mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}\left(\mathcal{T}^i(\mathbf{x}+\boldsymbol{\xi}) - \mathcal{S}^i(\mathbf{x}+\boldsymbol{\xi})\right)^2\right] \\ = \quad & \mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}\left(\mathcal{T}^i(\mathbf{x}) + \boldsymbol{\xi}\nabla_x \mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x}) - \boldsymbol{\xi}\nabla_x \mathcal{S}^i(\mathbf{x})\right)^2\right] + \mathcal{O}(\sigma^4) \\ = \quad & \sum_{i=1}^{k}\left(\mathcal{T}^i(\mathbf{x}) - \mathcal{S}^i(\mathbf{x})\right)^2 + \mathbb{E}_{\boldsymbol{\xi}}\left[\sum_{i=1}^{k}(\boldsymbol{\xi})^2\left(\nabla_x \mathcal{T}^i(\mathbf{x}) - \nabla_x \mathcal{S}^i(\mathbf{x})\right)^2\right] + \mathcal{O}(\sigma^4) \end{aligned} \tag{4}$$

To get equation 4, we use the fact that mean of $\boldsymbol{\xi}$ is zero. To complete the proof, we use the diagonal assumption on the covariance matrix of $\boldsymbol{\xi}$.

□

Proofs of other statements are similar. For proof of equation 3, use a second order Taylor series expansion of $\log(\cdot)$ in the first step.

Experimental details

Network Architectures

The architecture for our networks follow the VGG design philosophy. Specifically, we have blocks with the following elements:

- 3×3 conv kernels with c channels of stride 1
- Batch Normalization
- ReLU

Whenever we use Max-pooling (M), we use stride 2 and window size 2.

The architecture for VGG-9 is - $[64-M-128-M-256-256-M-512-512-M-512-512-M]$. Here, the number stands for the number of convolution channels, and M represents max-pooling. At the end of all the convolutional and max-pooling layers, we have a Global Average Pooling (GAP) layer, after which we have a fully connected layer leading up to the final classes.

The architecture for VGG-4 is - $[64 - M - 128 - M - 512 - M]$.

Similar architectures are used for both CIFAR10 and CIFAR100, and the only difference is in the final fully connected layer which has more output classes for CIFAR100.

Loss function

The loss function we use takes the following form in general.

$$\ell(\mathcal{S}, \mathcal{T}) = \alpha \times (\text{CE}) + \beta \times (\text{Match Activations}) + \gamma \times (\text{Match Jacobians})$$

In our experiments, α, β, γ are either set to 1 or 0. In other words, all regularization constants are 1.

Here, 'CE' refers to cross-entropy with ground truth labels. 'Match Activations' refers to squared error term over pre-softmax activations of the form $(y_s - y_t)^2$. 'Match Jacobians' refers to the same squared error term, but for Jacobians. We use the approximation specified in Section 4. For CIFAR100, the loss takes the following form:

$$\text{Match Jacobian} = \|\nabla_x \mathcal{T}^i(\mathbf{x}) - \nabla_x \mathcal{S}^i(\mathbf{x})\|_2^2$$

For CIFAR10, we use a slightly different loss which we found works better:

$$\text{Match Jacobian} = \left|\left| \frac{\nabla_x \mathcal{T}^i(\mathbf{x})}{\|\nabla_x \mathcal{T}^i(\mathbf{x})\|} - \frac{\nabla_x \mathcal{S}^i(\mathbf{x})}{\|\nabla_x \mathcal{S}^i(\mathbf{x})\|} \right|\right|_2^2$$

Optimization

We run optimization for 500 epochs. We use the Adam optimizer, with an initial learning rate of $1e-3$, and a single learning rate annealing (to $1e-4$) at 400 epochs.

Improving One-Shot Learning through Fusing Side Information

Yao-Hung Hubert Tsai **Ruslan Salakhutdinov**
Machine Learning Department, School of Computer Science, Carnegie Mellon University
{yaohungt, rsalakhu}@cs.cmu.edu

1 Introduction

Deep neural networks (DNNs) often struggle when training on classes with very few samples. In this paper, we focus on the extreme case: *one-shot learning* which has only one training sample per category. We treat the problem of one-shot learning to be a transfer learning problem: how to efficiently transfer the knowledge from 'lots-of-examples' to 'one-example' classes. More precisely, we propose to fuse side information for compensating the missing information across classes. In our paper, side information represents the relationship or prior knowledge between categories: for example, unsupervised feature vectors of categories derived from Wikipedia such as Word2Vec vectors (Mikolov et al., 2013), or tree hierarchy label structure such as WordNet structure (Miller, 1995).

We propose to first integrate side information using Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005) between the learned data embeddings and the learned label-affinity kernel, which is inferred from the side information. Since HSIC serves as a statistical dependency measurement, our learned feature representations can be maximally dependent on the corresponding label space. Next, to achieve better adaptation from 'lots-of-examples' to 'one-examples' classes, we introduce an attention mechanism for 'lots-of-examples' classes on the learned label-affinity kernel.

We empirically show that our proposed learning architecture (see Fig. 1) improves over traditional softmax regression networks as well as state-of-the-art attentional regression networks (Vinyals et al., 2016) on one-shot recognition tasks.

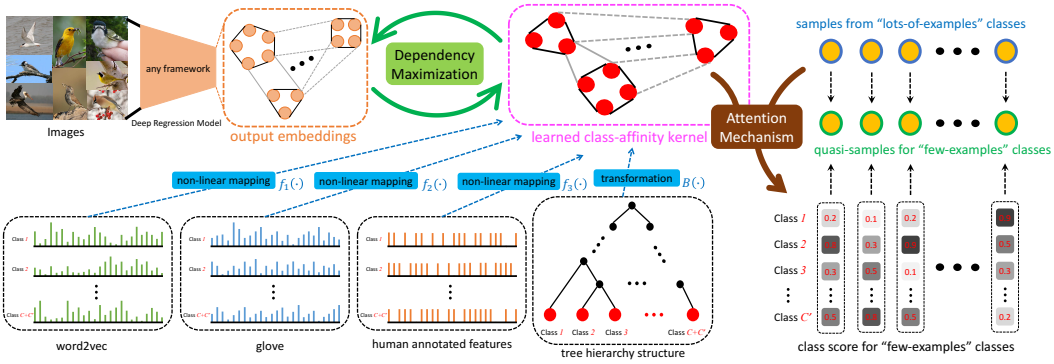


Figure 1: Fusing side information when learning data representation. We first construct a label-affinity kernel through deep kernel learning using multiple types of side information. Then, we enforce the dependency maximization criteria between the learned label-affinity kernel and the output embeddings of a regression model. Samples in 'lots-of-examples' classes are used to generate quasi-samples for 'one-example' classes. These generated quasi-samples can be viewed as additional training data.

2 Proposed Method

2.1 Notation

Let $\mathbf{S}$ denote the support set for the classes with lots of training examples. $\mathbf{S}$ consists of N data-label pairs $\mathbf{S} = \{\mathbf{X}, \mathbf{Y}\} = \{x_i, y_i\}_{i=1}^{N}$, where y_i ranges within C classes. We assume that we have M different kinds of side information $\mathbf{R} = \{R^1, R^2, \cdots, R^M\}$, where R^m can either be supervised/ unsupervised class embeddings or even hierarchical structures inferred from tree-based object structures such as ImageNet (Krizhevsky et al., 2012). Similarly, we have a different support set $\mathbf{S}'$ for 'one-examples' classes that $\mathbf{S}' = \{\mathbf{X}', \mathbf{Y}'\} = \{x'_i, y'_i\}_{i=1}^{N'}$ in which y'_i ranges within C' classes (disjoint from the classes in $\mathbf{S}$). Side information $\mathbf{R}' = \{R'^1, R'^2, \cdots, R'^M\}$ for $\mathbf{S}'$ is also provided. Last, θ_X and θ_R are the model parameters dealing with the data and side information, respectively.

2.2 Dependency Measure on Data and Side Information

The output embeddings $g_{\theta_X}(\mathbf{X})$ and side information $\mathbf{R}$ can be seen as two interdependent random variables, and we hope to maximize their dependency on each other. To achieve this goal, we adopt Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005). HSIC acts as a non-parametric independence test between two random variables, $g_{\theta_X}(\mathbf{X})$ and $\mathbf{R}$, by computing the Hilbert-Schmidt norm of the covariance operator over the corresponding domains $\mathcal{G} \times \mathcal{R}$. Furthermore, let k_g and k_r be the kernels on $\mathcal{G}, \mathcal{R}$ with associated Reproducing Kernel Hilbert Spaces (RKHSs). A slightly biased empirical estimation of HSIC (Gretton et al., 2005) could be written as follows:

$$\mathrm{HSIC}(\mathbf{S}, \mathbf{R}) = \frac{1}{(N-1)^2}\mathrm{tr}(\mathbf{H}\mathbf{K}_\mathbf{G}\mathbf{H}\mathbf{K}_\mathbf{R}), \tag{1}$$

where $\mathbf{K}_\mathbf{G} \in \mathbb{R}^{N\times N}$ with $\mathbf{K}_{\mathbf{G}ij} = k_g(x_i, x_j) = g_{\theta_X}(x_i)^\top \cdot g_{\theta_X}(x_j)$, $\mathbf{K}_\mathbf{R} \in \mathbb{R}^{N\times N}$ with $\mathbf{K}_{\mathbf{R}ij} = k_r(y_i, y_j) = \sum_{m=1}^{M} \frac{1}{M} k_{r^m}(y_i, y_j)$, and $\mathbf{H} \in \mathbb{R}^{N\times N}$ with $\mathbf{H}_{ij} = \mathbb{1}_{\{i=j\}} - \frac{1}{(N-1)^2}$. We consider two variants of $k_{r^m}(\cdot,\cdot)$ based on whether R^m is represented by class embeddings or tree-based label hierarchy. In short, $\mathbf{K}_\mathbf{G}$ and $\mathbf{K}_\mathbf{R}$ respectively stand for the relationships between data and categories, and HSIC provides a statistical dependency guarantee on the learned embeddings and labels.

a) R^m is represented by class embeddings:

Class embeddings can either be supervised features such as human annotated features or unsupervised features such as *word2vec* or *glove* features. Given $R^m = \{r_c^m\}_{c=1}^{C}$ with r_c^m representing class embeddings of class c, we define $k_{r^m}(\cdot,\cdot)$ as:

$$k_{r^m}(y_i, y_j) = f_{m,\theta_R}(r_{y_i}^m)^\top \cdot f_{m,\theta_R}(r_{y_j}^m),$$

where $f_{m,\theta_R}(\cdot)$ denotes the non-linear mapping from R^m. In this setting, we can capture the intrinsic structure by adjusting the categories' affinity through learning $f_{m,\theta_R}(\cdot)$ for different types of side information R^m.

b) R^m is represented by tree hierarchy:

If the labels form a tree hierarchy (e.g., *wordnet* (Miller, 1995) tree structure in ImageNet), then we can represent the labels as a tree covariance matrix $\mathbf{B}$ defined in Bravo et al. (2009), which is proved to be equivalent to the taxonomies in the tree (Blaschko et al., 2013). Specifically, following the definition of Theorem 2 in Bravo et al. (2009), a matrix $\mathbf{B} \in \mathbb{R}^{C\times C}$ is the tree-structured covariance matrix if and only if $\mathbf{B} = \mathbf{V}\mathbf{D}\mathbf{V}^\top$ where $\mathbf{D} \in \mathbb{R}^{2C-1\times 2C-1}$ is the diagonal matrix indicating the branch lengths of the tree and $\mathbf{V} \in \mathbb{R}^{C\times 2C-1}$ denoting the topology.

For any given tree-based label hierarchy, we define $k_{r^m}(\cdot,\cdot)$ to be

$$k_{r^m}(y_i, y_j) = (\mathbf{B}^m)_{y_i,y_j} = (\mathbf{Y}^\top \mathbf{B}^m \mathbf{Y})_{i,j} \ ,$$

where $\mathbf{Y} \in \{0,1\}^{C\times N}$ is the label matrix and $\mathbf{B}^m$ is the tree-structured covariance matrix of R^m. In other words, $k_{r^m}(y_i, y_j)$ indicates the weighted path from the root to the nearest common ancestor of nodes y_i and y_j (see Lemma 1 in (Blaschko et al., 2013)).

In eq. (1), we can try integrating different types of side information R^m with both class-embedding and tree-hierarchy-structure representation. In short, maximizing eq. (1) makes the data representation

kernel $\mathbf{K_G}$ maximally dependent on the side information $\mathbf{R}$ seen from the kernel matrix $\mathbf{K_R}$. Hence, introducing HSIC criterion provides an excellent way of transferring knowledge across different classes. Note that, if $\mathbf{K_R}$ is an identity matrix, then there are no relationships between categories, which results in a standard classification problem.

So far, we have defined a joint learning on the support set $\mathbf{S}$ and its side information $\mathbf{R}$. If we have access to different support set $\mathbf{S}'$ and the corresponding side information $\mathbf{R}'$, we can easily incorporate them into the HSIC criterion; i.e., $\text{HSIC}(\{\mathbf{S}, \mathbf{S}'\}, \{\mathbf{R}, \mathbf{R}'\})$. Hence we can effectively transfer the knowledge both intra and inter sets.

2.3 Quasi-Samples Generation

Our second aim is to use a significant amount of data in 'lots-of-examples' classes to learn the prediction model for 'one-example' classes. We present an attention mechanism over the side information $\mathbf{R}$ and $\mathbf{R}'$ to achieve this goal.

For a given data-label pair $\{x, y\}$ in $\mathbf{S}$, we define its quasi-label $\tilde{y}'$ as follows:

$$\tilde{y}' = P_{\theta_R}(y'|y; \mathbf{R}, \mathbf{R}') = \sum_{i \in \mathbf{S}'} a_r(y, y'_i) y'_i,$$

where $a_r(\cdot, \cdot)$ acts as an attentional kernel from $\mathbf{R}$ to $\mathbf{R}'$, which can be formulated as

$$a_r(y, y'_i) = \frac{e^{k_r(y, y'_i)}}{\sum_{j \in \mathbf{S}'} e^{k_r(y, y'_j)}}.$$

In other words, given the learned label affinity kernel, for each category in 'lots-of-examples' classes, we can form a label probability distribution on the label space for 'one-example' classes; i.e., $\tilde{y}' = P_{\theta_R}(y'|y; \mathbf{R}, \mathbf{R}')$. Moreover, given the other set $\mathbf{S}'$, we can also derive the label probability distribution $P_{\theta_X}(y'|x; \mathbf{S}')$ under any regression model for 'one-example' classes. Our strategy is to minimize the cross entropy between $P_\theta(y'|x; \mathbf{S}')$ and $\tilde{y}'$. In short, we can treat each data-label pair $\{x, y\}$ in 'lots-of-examples' classes to be a quasi-sample $\{x, \tilde{y}'\}$ for 'one-example' classes, as illustrated in Fig. 2.

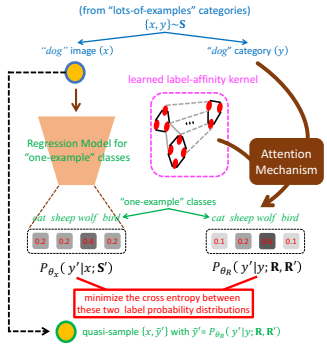


Figure 2: Quasi-samples generation: We take *dog* as an example class from "lots-of-examples" categories. "One-example" categories consist of *cat, sheep, wolf,* and *bird*. Best viewed in color.

2.4 Objectives

The overall training objective is defined as follows:

$$\max \ \alpha \, \text{HSIC}\big(\{\mathbf{S}, \mathbf{S}'\}, \{\mathbf{R}, \mathbf{R}'\}\big) + \frac{1}{|\mathbf{S}|} \sum_{i \in \mathbf{S}} y_i^\top \log P_{\theta_X}\big(y_i|x_i; \mathbf{S}\big) + \alpha \, \tilde{y}_i'^\top \log P_{\theta_X}\big(y_i'|x_i; \mathbf{S}'\big),$$

where α is the trade-off parameter.

For any given test example x'_{test}, the predicted output class is defined as

$$\hat{y}'_{test} = \text{argmax}_{y'} \, P_{\theta_X}(y'|x'_{test}; \mathbf{S}').$$

Table 1: Average performance (%) over 40 random trials for standard one-shot recognition task.

Dataset	softmax_net	HSIC$^{\dagger}_{softmax}$	HSIC$_{softmax}$	attention_net [Vinyals et al. (2016)]	HSIC$^{\dagger}_{attention}$	HSIC$_{attention}$
CUB	26.93 ± 2.41	29.26 ± 2.22	$\mathbf{31.49 \pm 2.28}$	29.12 ± 2.44	33.12 ± 2.48	$\mathbf{33.75 \pm 2.43}$
AwA	66.39 ± 5.38	69.98 ± 5.47	$\mathbf{71.29 \pm 5.64}$	72.27 ± 5.82	$\mathbf{77.86 \pm 4.76}$	76.98 ± 4.99

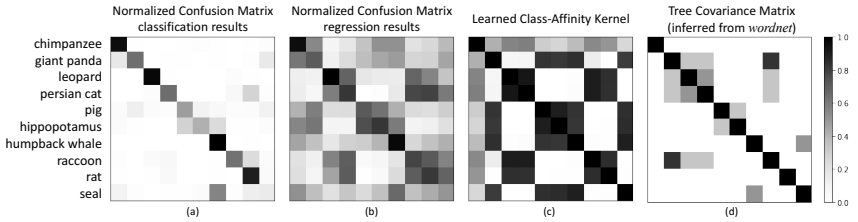


Figure 3: For AwA dataset: (a) normalized confusion matrix for classification, (b) normalized confusion matrix for regression, (c) learned class-affinity kernel in proposed$_{attention}$, and (d) tree covariance matrix.

3 EVALUATION

We evaluate our method (HSIC$_{softmax}$ and HSIC$_{attention}$) on top of two different regression networks: traditional softmax regression (softmax_net) and attentional regression (attention_net) introduced by (Vinyals et al., 2016). Two datasets are adopted for one-shot recognition task: Caltech-UCSD Birds 200-2011 (CUB) (Welinder et al., 2010) and Animals with Attributes (AwA) (Lampert et al., 2014). CUB is a fine-grained dataset in which the categories are both visually and semantically similar, while AwA is a general dataset. Four types of side information are considered: supervised human annotated attributes (att) (Lampert et al., 2014), unsupervised Word2Vec features ($w2v$) (Mikolov et al., 2013), unsupervised Glove features (glo) (Pennington et al., 2014), and the hierarchy tree structures (hie) inferred from *wordnet* (Miller, 1995). We also provide two variants (HSIC$^{\dagger}_{softmax}$ and HSIC$^{\dagger}_{attention}$) when considering no quasi-samples generation technique.

One-Shot Recognition Task: Table 1 lists the average recognition performance for our standard one-shot recognition experiments. HSIC$_{softmax}$ and HSIC$_{attention}$ are jointly learned with all four types of side information: att, $w2v$, glo, and hie. We first observe that the methods with side information achieve superior performance over the methods which do not learn with side information. For example, HSIC$_{softmax}$ improves over softmax_net by 4.56% on CUB dataset and HSIC$_{attention}$ enjoys 4.71% gain over attention_net on AwA dataset. These results indicate that fusing side information can benefit one-shot learning.

Next, we examine the variants of our proposed architecture. In most cases, the construction of the quasi-samples benefits the one-shot learning. The only exception is the 0.88% performance drop from HSIC$^{\dagger}_{attention}$ to HSIC$_{attention}$ in AwA. Nevertheless, we find that our model converges faster when introducing the technique of generating quasi-samples.

Confusion Matrix and the Learned Class-Affinity Kernel: Following the above experimental setting, for test classes in AwA, in Fig. 3, we provide the confusion matrix, the learned label-affinity kernel using HSIC$_{attention}$, and the tree covariance matrix (Bravo et al., 2009). We first take a look at the normalized confusion matrix for classification results. For example, we observe that *seal* is often misclassified as *humpback whale*; and from the tree covariance matrix, we know that *seal* is semantically most similar to *humpback whale*. Therefore, even though our model cannot predict *seal* images correctly, it still can find its semantically most similar classes.

Additionally, it is not surprising that Fig. 3(b), normalized confusion matrix, is visually similar to Fig. 3(c), the learned class-affinity kernel. The reason is that one of our objectives is to learn the output embeddings of images to be maximally dependent on the given side information. Note that, in this experiment, our side information contains supervised human annotated attributes, unsupervised word vectors (Word2Vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014)), and a *WordNet* (Miller, 1995) tree hierarchy.

On the other hand, we also observe the obvious change in classes relationships from *WordNet* tree hierarchy (Fig. 3 (d)) to our learned class-affinity kernel (Fig. 3 (c)). For instance, *raccoon* and *giant panda* are species-related, but they distinctly differ in size and color. This important information is missed in *WordNet* but not missed in human annotated features or word vectors extracted from Wikipedia. Hence, our model bears the capability of arranging and properly fusing various types of side information.

Table 2: Average performance (%) for the different availability of side information.

CUB							
available side information	*none*	*att*	*w2v*	*glo*	*hie*	*att/w2v/glo*	*all*
HSIC$_{softmax}$	26.93 ± 2.41	30.93 ± 2.25	30.67 ± 2.10	30.53 ± 2.42	32.15 ± 2.28	30.58 ± 2.12	31.49 ± 2.28
HSIC$_{attention}$	29.12 ± 2.44	32.86 ± 2.34	33.37 ± 2.30	33.31 ± 2.50	**34.10 ± 2.40**	33.72 ± 2.45	33.75 ± 2.43
AwA							
available side information	*none*	*att*	*w2v*	*glo*	*hie*	*att/w2v/glo*	*all*
HSIC$_{softmax}$	66.39 ± 5.38	70.08 ± 5.27	69.30 ± 5.41	69.94 ± 5.62	73.32 ± 5.12	70.44 ± 6.74	71.29 ± 5.64
HSIC$_{attention}$	72.27 ± 5.82	76.60 ± 5.05	76.60 ± 5.15	**77.38 ± 5.15**	76.88 ± 5.27	76.84 ± 5.65	76.98 ± 4.99

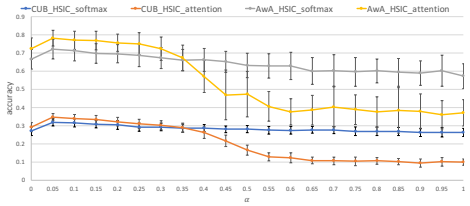


Figure 4: Parameter sensitivity analysis experiment. Our proposed methods jointly learn with all four side information: *att*, *w2v*, *glo*, and *hie*. Best viewed in color.

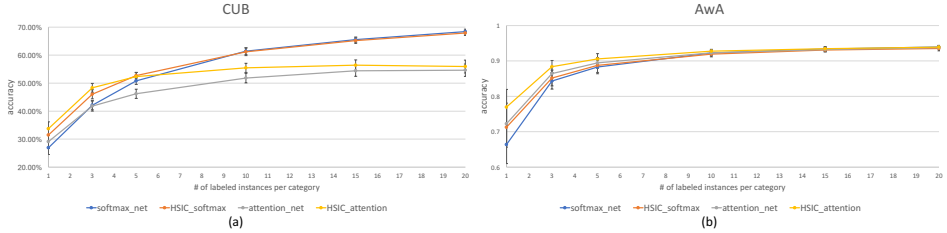


Figure 5: Experiment for increasing labeled instance per category in test classes. Our proposed methods jointly learn with all four side information: *att*, *w2v*, *glo*, and *hie*. Best viewed in color.

Availability of Various Types of Side Information: In Table 2, we evaluate our proposed methods when not all four types of side information are available during training. It is surprising to find that there is no particular rule of combining multiple side information or using a single side information to obtain the best performance. A possible reason would be the non-optima for using kernel average. That is to say, in our current setting, we equally treat contribution of every type of side information to the learning of our label-affinity kernel. Nevertheless, we still enjoy performance improvement of using side information compared to not using it.

Parameter Sensitivity on α: Since α stands for the trade-off parameter for fusing side information through HSIC and quasi-examples generation technique, we studied how it affects model performance. We alter α from 0 to 1.0 by step size of 0.05 for both HSIC$_{softmax}$ and HSIC$_{attention}$ models. Fig. 4 shows that larger values of α does not lead to better performance. When $\alpha \leq 0.3$, our proposed method outperforms softmax_net and attention_net. Note that HSIC$_{softmax}$ and HSIC$_{attention}$ relax to softmax_net and attention_net when $\alpha = 0$. When $\alpha > 0.3$, the performance of our proposed method begins to drop significantly, especially for HSIC$_{attention}$. This is primarily because too large values of α may cause the output embeddings of images to be confused by semantically similar but visually different classes in the learned label-affinity kernel (e.g., Fig. 3 (c)).

From One-Shot to Few-Shot Learning: In Fig. 5, we increase the labeled instances in test classes and evaluate the performance of softmax_net, attention_net, and our proposed architecture. We observe that HSIC$_{softmax}$ converges to softmax_net and HSIC$_{attention}$ converges to attention_net when more labeled data are available in test classes during training. In other words, as labeled instances increase, the power of fusing side information within deep learning diminishes. This result is quite intuitive as deep architecture perform well when training on lots of labeled data.

For the fine-grained dataset CUB, we also observe that *attentional regression* methods are at first outperform *softmax regression* methods, but perform worse when more labeled data are present during training. Note that *softmax regression* networks have one additional softmax layer (one-hidden-layer fully-connected neural network) compared to *attentional regression* networks. Therefore, *softmax regression* networks can deal with more complex regression functions (i.e., regression for the fine-grained CUB dataset) as long as they have enough labeled examples.

Acknowledgements

This work was supported by DARPA award D17AP00001, Google focused award, and Nvidia NVAIL award.

References

Blaschko, M. B., Zaremba, W., and Gretton, A. (2013). Taxonomic prediction with tree-structured covariances. In *ECML-PKDD*.

Bravo, H. C., Wright, S. J., Eng, K. H., Keles, S., and Wahba, G. (2009). Estimating tree-structured covariance matrices via mixed-integer programming. In *AISTATS*.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

Lampert, C. H., Nickisch, H., and Harmeling, S. (2014). Attribute-based classification for zero-shot visual object categorization. *IEEE T-PAMI*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *NIPS*.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. In *NIPS*.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology.

Multiclass Domain Generalization

Aniket Anand Deshmukh
Department of EECS
University of Michigan
Ann Arbor, MI 48109, USA
aniketde@umich.edu

Srinagesh Sharma
Department of EECS
University of Michigan
Ann Arbor, MI 48109, USA
srinag@umich.edu

James W. Cutler
Department of Aerospace Engineering
University of Michigan
Ann Arbor, MI 48109, USA
jwcutler@umich.edu

Clayton Scott
Department of EECS
University of Michigan
Ann Arbor, MI 48109, USA
clayscot@umich.edu

Abstract

Domain generalization is the problem of assigning labels to an unlabeled data set, given several similar data sets for which labels have been provided. In this work, we consider a kernel-based algorithm for domain generalization that was developed in the binary setting. In particular, we show that the generalization error bound for this algorithm extends to the multi-class setting. Our main contribution includes generalization error analysis and a scalable implementation of the approach. We demonstrate improved performance with respect to a pooling strategy on four data sets.

1 Introduction

Transfer learning, domain adaptation, and weakly supervised learning all have the goal of generalizing without access to conventional labeled training data. One particular form of transfer learning that has garnered increasing attention in recent years is *domain generalization* (DG) [2, 3]. In this setting, the learner is given unlabeled data to classify, and must do so by leveraging labeled data sets from similar yet distinct classification problems. In other words, label training data drawn from the same distribution as the test data are not available, but are available from several related tasks. We use the terms “task" and “domain" interchangeably throughout this paper.

Applications of DG are numerous. For example, each task may be a prediction problem associated to a particular individual (e.g., handwritten digit recognition), and the variation between individuals accounts for the variation among the data sets. Domain generalization is needed when a new individual appears, and the only training data come from different subjects.

As another application, below we consider DG for determining the orbits of microsatellites, which are increasingly deployed in space missions for a variety of scientific and technological purposes. Because of randomness in the launch process, the orbit of a microsatellite is random, and must be determined after the launch. Furthermore, ground antennae are not able to decode unique identifier signals transmitted by the microsatellites because of communication resource constraints and uncertainty in satellite position and dynamics. More concretely, suppose c microsatellites are launched together. Each launch is a random phenomenon and may be viewed as a task in our framework. One can simulate the launch of microsatellites using domain knowledge to generate highly realistic training data (feature vectors of ground antennae RF measurements, and labels of satellite ID). One can then

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

transfer knowledge from the simulated training data to label (identify the satellite) the measurements from a real-world launch with high accuracy.

Several approaches to domain generalization have been proposed, including complexity regularization that adapts to the variability of the sampling distribution on tasks [2, 3, 21, 27], learning this sampling distribution directly [5], task matching by optimal transport [8], learning a feature extractor that puts all tasks in a common feature space [26, 25, 23, 28, 15, 13], and using the marginal distribution from which a feature vector is drawn as a feature itself for label prediction [4].

Our work builds on the approach of [4], which develops a kernel-based framework for DG. We review this framework below, and extend their analysis, which addresses the setting of binary labels, to the multiclass setting. While several aspects of the original analysis in [4] carry over to the multiclass case, others do not. In particular, we use an extension of the contraction lemma for Rademacher complexity of Lipschitz loss classes to prove the generalization error bound [22].

A few existing approaches to DG address the multiclass case [8, 14, 24, 19, 16, 10]. Most of these works rely on neural networks and none have statistical performance guarantees. We also note that the approach of [4] can be combined with feature extraction approaches, as was done in [25], and the same is true of our multiclass extension.

Our contributions include: (1) Extending the kernel-based approach to DG from [4] to multiclass DG, (2) Extending the analysis of [4] to multiclass, (3) a scalable implementation based on random Fourier features, and (4) experimental demonstration of the method compared to a pooling approach.

In section 2 we formally state the DG problem and in section 3 we describe the kernel-based learning algorithm. Section 4 contains our theoretical analysis, and experimental results appear in section 5.

2 Formal Problem Statement

Let $\mathcal{X}$ be the feature space and $\mathcal{Y}$ the label space with $|\mathcal{Y}| = c$. Denote by $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$ the set of probability distributions on $\mathcal{X}\times\mathcal{Y}$, $\mathcal{P}_{\mathcal{X}}$ the set of probability distributions on $\mathcal{X}$, and $\mathcal{P}_{\mathcal{Y}|\mathcal{X}}$ the set of conditional distributions of Y given X. Furthermore, let μ be a probability measure on $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$, i.e., whose realizations are distributions on $\mathcal{X}\times\mathcal{Y}$.

With the above notations, DG is defined as follows. We are given training data sets $S_i = ((X_{ij}, Y_{ij}))_{1\le j\le n_i}$ such that $(X_{ij}, Y_{ij}) \sim P^i_{XY}$ and $P^i_{XY} \sim \mu$. The test data set is $S^T = ((X^T_j, Y^T_j))_{1\le j\le n_T}$ such that $(X^T_j, Y^T_j) \sim P^T$ and $P^T \sim \mu$. We assume all (X, Y) pairs are drawn iid from their respective distributions, and that $P_1, \ldots, P_N, P^T$ are iid from μ. The Y^T_j are not visible to the learner, and the goal is to accurately predict $(Y^T_j)_{1\le j\le n_T}$. For any predicted estimate of a label $\hat{Y}$, the accuracy is evaluated using a loss function $\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}$. For greater flexibility in the multiclass case ($c > 2$), the label space for prediction is relaxed to $\mathbb{R}^c$ and a surrogate loss function $\ell : \mathbb{R}^c \times \mathcal{Y} \to \mathbb{R}_+$ is employed.

According to the approach in [4], DG is cast as a supervised learning problem where the input to the classifier is the extended feature space $\mathcal{P}_{\mathcal{X}}\times\mathcal{X}$. A decision function is a function $f : \mathcal{P}_{\mathcal{X}}\times\mathcal{X} \to \mathbb{R}^c$ that predicts $\hat{Y}^T_j = f(\hat{P}^T_X, X^T_j)$, where $\hat{P}_X$ is the associated empirical distribution. The decision function can be separated into its components $f = \begin{bmatrix} g_1 & g_2 & \cdots & g_c \end{bmatrix}$ such that $g_m : \mathcal{P}_{\mathcal{X}}\times\mathcal{X} \to \mathbb{R}$, for $m = 1, 2, \ldots c$. We define the empirical training error as

$$\widehat{\varepsilon}(f) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{j=1}^{n_i}\ell(f(\widehat{P}^i_X, X_{ij}), Y_{ij}), \tag{1}$$

and by denoting $\tilde{X} = (P_X, X)$, the generalization error of a decision function with respect to loss ℓ as

$$\varepsilon(f) = E_{P^T_{XY}\sim\mu}E_{(X^T,Y^T)\sim P^T_{XY}}\ell(f(P^T_X, X^T), Y^T) = E_{P^T_{XY}\sim\mu}E_{(X^T,Y^T)\sim P^T_{XY}}\ell(f(\tilde{X}^T), Y^T). \tag{2}$$

The goal of DG is to learn an f that minimizes this generalization error.

Remarks: (1) Although the generalization error assumes the predictor has access to P_X, at training time as well as at test time P_X is only known through the empirical marginal $\hat{P}_X$. (2) Despite the similarity to standard classification in the infinite sample case, the learning task here is different, because the realizations $(\tilde{X}_{ij}, Y_{ij})$ are neither independent nor identically distributed. (3) Examples

of loss functions l can be found in Lee et. al [18], Crammer and Singer [9], Weston and Watkins [33]. For detailed discussion on different multiclass loss functions and their general forms see [11, 32, 17, 30].

3 Kernel Based Learning Algorithm

The goal of predicting an optimal classifier on the extended feature space can be solved using kernel based algorithms. For a (symmetric positive definite) kernel k, let H_k denote its associated reproducing kernel Hilbert space. Let $\bar{k} : (\mathcal{P}_X \times \mathcal{X}) \times (\mathcal{P}_X \times \mathcal{X}) \to \mathbb{R}$ be a symmetric and positive definite kernel on $\mathcal{P}_X \times \mathcal{X}$, whose construction will be described below. Let ℓ be a loss function. Further let $\hat{P}^i_X$ be the finite sample empirical distribution for sample S_i corresponding to X_{ij}, and let $\tilde{X}_{ij} = (\hat{P}^i_X, X_{ij})$ be the extended data point. We will find a decision function $f \in H^c_{\bar{k}} := H_{\bar{k}} \times \cdots H_{\bar{k}}$ (c times) and has components $g_l \in H_{\bar{k}}, l = 1, 2, ...c$, i.e., $f = \begin{bmatrix} g_1 & g_2 & \cdots & g_c \end{bmatrix}$. Define

$$\hat{f}_\lambda = \underset{f \in H^c_{\bar{k}}}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \ell(f(\tilde{X}_{ij}), Y_{ij}) + \lambda r(f), \tag{3}$$

as the empirical estimate of the optimal decision function. Define the regularizer $r(f)$ as $r(f) := \|f\|^2_{H^c_{\bar{k}}} := \sum_{m=1}^{c} \|g_m\|^2_{H_{\bar{k}}}$. The kernel $\bar{k}$ can be constructed from 3 other kernels k_x, k'_x and κ. Let k_x and k'_x be kernels on $\mathcal{X}$. For example, if $\mathcal{X}$ is $\mathbb{R}^d$, k_x and k'_x could be Gaussian kernels. The so-called kernel mean embedding is the mapping $\Phi : \mathcal{P}_\mathcal{X} \to H_{k'_x}$,

$$\Phi(P) = \int_X k'_x(x, \cdot) dP. \tag{4}$$

Let κ be a kernel-like function on $\Phi(\mathcal{P}_\mathcal{X})$, such as the Gaussian-like function $\kappa(\Phi(P^1_X), \Phi(P^2_X)) = \exp(-\|\Phi(P^1_X) - \Phi(P^2_X)\|^2/2\sigma^2_\kappa)$. Then $\kappa(\Phi(\cdot), \Phi(\cdot))$ is a kernel on $\mathcal{P}_\mathcal{X}$ [6], and we can now define the kernel on the extended feature space via as a product kernel

$$\bar{k}((P^1_x, X_1), (P^2_x, X_2)) = \kappa(\Phi(P^1_X), \Phi(P^2_X)) k_x(X_1, X_2). \tag{5}$$

The empirical estimate of Φ can be computed for $\{X_{ij}\}_{1 \le j \le n}$, $X_{ij} \sim P^i_X$ as $\Phi(\hat{P}^i_X) = \frac{1}{n} \sum_j k'_x(X_{ij}, \cdot)$. The algorithm associated with the minimizer is similar to multiclass extensions of SVMs such as those presented in [18] applied over the extended feature space. The representer theorem applies in modified form for the optimization (3).

4 Generalization Error Analysis

We make the following assumptions to analyze the generalization error. For any kernel k, $\phi_k(x) := k(\cdot, x) \in H_k$ denotes the canonical feature map, $\mathbb{B}_k(R)$ refers to the closed ball of radius R in H_k and $\mathbb{B}^c_k(R) := \prod_{m=1}^{c} \mathbb{B}_k(R)$ refers to the product space of c closed balls.

A I The loss function $\ell : \mathbb{R}^c \times \mathcal{Y} \to R$ is bounded by B_ℓ, and is L_ℓ-Lipschitz in the first variable: For all y, $|\ell(T_1, y) - \ell(T_2, y)| \le L_\ell \|T_1 - T_2\|_2$ for $T_1, T_2 \in \mathbb{R}^c$.

A II Kernels k_x, k'_x, κ are bounded by $B^2_k, B^2_{k'}, B^2_\kappa$ respectively.

A III The canonical feature map $\phi_\kappa : H_{k'_x} \to H_\kappa$ is α-Hölder continuous, i.e., $\forall a, b \in \mathbb{B}_{k'_x}(B_{k'})$:

$$\|\phi_\kappa(a) - \phi_\kappa(b)\|_2 \le L_\kappa \|a - b\|^\alpha_2.$$

The above assumptions are similar to those presented in [4] translated to multiclass data. Condition **A III** holds with $\alpha = 1$ when κ is the Gaussian-like kernel on $H_{k'_x}$. Using the stated assumptions we shall now develop generalization error bounds for multiclass DG. To generalize the analysis, an extension of Talagrand's lemma for bounding the Rademacher complexity is needed. Such an extension was provided by [20, 22] and [7].

Lemma 1. ***(Vector Valued Talagrand's Contraction Lemma)*** *[22] Let $\mathcal{F}$ be a class of functions from $\mathcal{X} \to \mathbb{R}^c$. Let $\{\mu_i\}_{i=1}^N$ and $\{\sigma_{ij}\}_{i=1,j=1}^{N,c}$ be two sets of independent Rademacher random variables. If $\psi : \mathbb{R}^c \to \mathbb{R}$ is L-Lipschitz under $\|\cdot\|_p$ where $p \ge 2$, then*

$$\mathbb{E}_{\mu}\big[\sup_{f\in\mathcal{F}}\sum_{i=1}^{N}\mu_i\psi(f(x_i))\big] \le \sqrt{2}L\mathbb{E}_{\sigma}\big[\sup_{f\in\mathcal{F}}\sum_{i=1}^{N}\sum_{j=1}^{c}\sigma_{ij}g_j(x_i)\big].$$

For simplicity's sake, we assume that $n_i = n$ to state the generalization error bound.

Theorem 1. ***(Estimation error control)*** *Assuming that conditions* ***A I*** *-* ***A III*** *hold then for any $R > 0$, with probability at least $1-\delta$:*

$$\sup_{f\in\mathbb{B}_k^c(R)}|\widehat{\varepsilon}(f)-\varepsilon(f)| \le L_\ell L_\kappa R B_k c(B_{k'})^\alpha\Big(\sqrt{\frac{2\log(2N\delta^{-1})}{n}}+\sqrt{\frac{1}{n}}+\frac{4\log(2N\delta^{-1})}{3n}\Big)^\alpha + \frac{8\sqrt{2}RL_\ell B_k B_\kappa c}{\sqrt{N}} + B_\ell\sqrt{\frac{\log(8\delta^{-1})}{2N}}$$

Proof Sketch Let $\mathcal{E}(f) = |\widehat{\varepsilon}(f)-\varepsilon(f)|$.

$$\begin{aligned}\sup_{f\in\mathbb{B}_k^c(R)}\mathcal{E}(f) &\le \sup_{f\in\mathbb{B}_k^c(R)}\Big|\widehat{\varepsilon}(f)-\frac{1}{Nn}\sum_{i=1}^{N}\sum_{j=1}^{n}\ell(f(\tilde{X}_{ij}),Y_{ij})\Big| + \sup_{f\in\mathbb{B}_k^c(R)}\Big|\frac{1}{Nn}\sum_{i=1}^{N}\sum_{j=1}^{n}\ell(f(\tilde{X}_{ij}),Y_{ij})-\varepsilon(f)\Big| \\ &= (I)+(II)\end{aligned}$$

Term (I) is bounded by application of Lipschitz continuity of ℓ, union bounds for tasks and classes over f and through Hölder continuity in assumption **A III**. Bounding the term (II) is similar to bounding term (II) in Theorem 5 in [4] with modifications for multi-class loss. In addition, the modified Talagrand's lemma 1 is applied to bound the Rademacher complexity [22].

5 Results

We test the proposed algorithm on 4 multiclass datasets and compare it with pooling, where data from all the tasks are pooled together to learn one single classifier. Datasets description are given below and a summary is in Table 1.

Dataset	Training Tasks	Test Tasks	Examples Per Task	Classes
Synthetic	80	20	100	10
Satellite	400	100	77-165	3
HAR	20	10	300	6
MNIST-MOD	80	20	100	10

Table 1: Summary of Datasets

Synthetic Dataset: Features for synthetic data are drawn from the unit square. Based on one of the dimensions, the data are labeled from 0 to 10, e.g., if the feature value is between 0 and 0.1, then it's labeled as 1, if it's in between 0.1 and 0.2, then it's labeled as 2, and so on. After that, the feature vectors are rotated clockwise by an angle randomly drawn from 0 to 180 degrees to get data for one task. The process is repeated 100 times to get data for 100 tasks out of which 80 are train tasks and 20 are test tasks. Fig. 1 shows 3 such tasks for $\theta = 0, 90$ and 180 where the supports don't overlap at all, and Fig. 2 shows 13 tasks where the supports overlap.

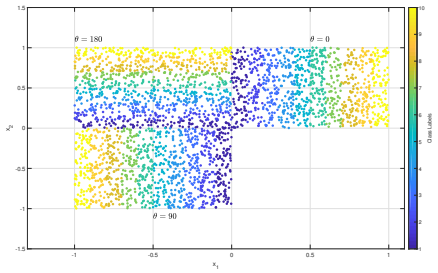


Figure 1: Synthetic Dataset: Three tasks $\theta = \{0, 90, 180\}$

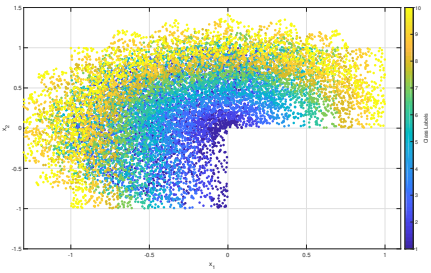


Figure 2: Synthetic Dataset: Thirteen tasks $\theta = \{0, 15, 30, ..., 180\}$

Satellite Dataset: The problem is described in the introduction, and we used the dataset presented by [31] modified for $c = 3$ spacecraft.

HAR Dataset: This is a human activity recognition using smart-phone dataset from UCI repository [1]. Each of 30 volunteers performed six activities (walking, walking upstairs, walking downstairs, sitting, standing, laying) wearing the smart-phone.

MNIST-MOD Dataset: We randomly draw 1000 images from MNIST's train dataset. Then we rotate each of this image by randomly drawn angle from 0 to 180 degrees and repeat this 100 times to get data for 100 tasks. Example for rotated MNIST dataset is shown in Fig. 3.

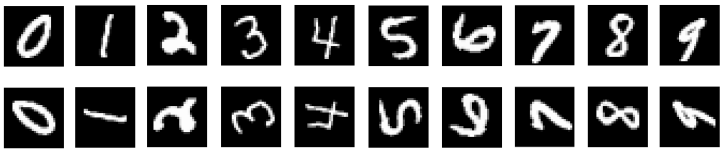


Figure 3: MNIST Data with no rotation (first row) and 90 degree rotation (second row)

We use all Gaussian kernels and a novel random Fourier Feature (RFF) approximation, which extends the usual RFF approximation on Euclidean space $\mathcal{X}$ [29] to the extended feature space $\mathcal{P}_{\mathcal{X}} \times \mathcal{X}$, to speed up the algorithm. We used Liblinear package for the implementation [12]. All hyperparameters were selected using five fold cross-validation and experiments were repeated 10 times. We show results in Table 2. The proposed method performs the best in three datasets and equally well in the one remaining dataset. The more our method outperforms pooling, the more knowledge can be shared between tasks.

Dataset	Pooling	Proposed Method
Synthetic	70.73 (±2.30)	**25.40** (±1.72)
Satellite	11.95 (±0.46)	**8.28** (±0.79)
HAR	1.69 (±0.56)	**1.68** (±0.58)
MNIST-MOD	22.79 (±1.38)	**21.39** (±1.24)

Table 2: Percentage Error

6 Conclusion and Future Work

In this work, we extended the kernel-based algorithm for domain generalization of [4] to the multiclass setting, along with its generalization error bound. We implemented the approach, demonstrating its improved performance with respect to a pooling strategy on four data sets. Future work will focus on improved generalization bounds and extensions to zero shot learning. Our bound depends polynomially on c and for large number of classes, this may not be desirable. We intend to investigate assumptions under which there is a chance to improve this dependency. In extensions, we are interested in zero shot learning where training tasks have c classes and test tasks have $c + 1$ classes.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones.

[2] J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. 28(1):7–39, 1997.

[3] J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

[4] G. Blanchard, G. Lee, and C. Scott. Generalizing from several related classification tasks to a new unlabeled sample. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2178–2186. 2011.

[5] J. Carbonell, S. Hanneke, and L. Yang. A theory of transfer learning with applications to active learning. 90(2):161–189, 2013.

[6] A. Christmann and I. Steinwart. Universal kernels on non-standard input spaces. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 406–414, 2010.

[7] C. Cortes, V. Kuznetsov, M. Mohri, and S. Yang. Structured prediction theory based on factor graph complexity. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2514–2522. Curran Associates, Inc., 2016.

[8] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy. Optimal transport for domain adaptation. 39(9):1853–1865, 2016.

[9] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[10] Z. Ding and Y. Fu. Deep domain generalization with structured low-rank constraint. *IEEE Transactions on Image Processing*, 2017.

[11] Ü. Doğan, T. Glasmachers, and C. Igel. A unified view on multi-class support vector classification. *Journal of Machine Learning Research*, 17(45):1–32, 2016.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[13] M. Ghifary, D. Balduzzi, B. Kleijn, and M. Zhang. Scatter component analysis: A unified framework for domain adaptation and domain generalization. 39(7):1411–1430, 2017.

[14] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015.

[15] T. Grubinger, A. Birlutiu, H. Schöner, T. Natschläger, and T. Heskes. Domain generalization based on transfer component analysis. In I. Rojas, G. Joya, and A. Catala, editors, *Advances in Computational Intelligence. IWANN 2015*, volume 9094 of *Lecture Notes in Computer Science*, pages 325–334. Springer, 2015.

[16] T. Grubinger, A. Birlutiu, H. Schöner, T. Natschläger, and T. Heskes. Multi-domain transfer component analysis for domain generalization. *Neural Processing Letters*, pages 1–11, 2017.

[17] C. Hsu and C. Lin. A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425, 2002.

[18] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465):67–81, 2004.

[19] D. Li, Y. Yang, Y. Song, and T. M. Hospedales. Learning to generalize: Meta-learning for domain generalization. *arXiv preprint arXiv:1710.03463*, 2017.

[20] B. London, B. Huang, B. Taskar, and L. Getoor. Collective stability in structured prediction: Generalization from one example.

[21] A. Maurer. Transfer bounds for linear feature learning. 75(3):327–350, 2009.

[22] A. Maurer. A vector-contraction inequality for rademacher complexities. *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings*, pages 3–17, 2016.

[23] A. Maurer, M. Pontil, and B. Romera-Paredes. Sparse coding for multitask and transfer learning. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 343–351, 2013.

[24] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto. Unified deep supervised domain adaptation and generalization. *arXiv preprint arXiv:1709.10190*, 2017.

[25] K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *Proceedings of the 30th International Conference on International Conference on Machine Learning (ICML'13)*, volume 28 of *Proceedings of Machine Learning Research*, pages I–10–I–18, 2013.

[26] S. J. Pan, I. Tsang, J. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[27] A. Pentina and S. Ben-David. Multi-task and lifelong learning of kernels. In K. Chaudhuri, C. Gentile, and S. Zilles, editors, *Algorithmic Learning Theory: 26th International Conference (ALT'15)*, volume 9355 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2015.

[28] A. Pentina and C. Lampert. A pac-bayesian bound for lifelong learning. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 991–999, 2014.

[29] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[30] H. Ramaswamy and S. Agarwal. Convex calibration dimension for multiclass loss matrices. *Journal of Machine Learning Research*, 17, 2016.

[31] S. Sharma and J. W. Cutler. Robust orbit determination and classification: A learning theoretic approach. *Interplanetary Network Progress Report*, 203:1, 2015.

[32] A. Tewari and P. L. Bartlett. On the consistency of multiclass classification methods. *Journal of Machine Learning Research*, 8(May):1007–1025, 2007.

[33] J. Weston and C. Watkins. Multi-class support vector machines. Technical report, Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, May, 1998.

Neural Skill Transfer from Supervised Language Tasks to Reading Comprehension

Todor Mihaylov [1,3], Zornitsa Kozareva[2], and Anette Frank[1,3]

[1]Institute for Computational Linguistics, Heidelberg University, Germany
[2]Amazon, AWS Deep Learning, Palo Alto, CA
[3]Research Training Group AIPHES
{mihaylov,frank}@cl.uni-heidelberg.de, zornitsa@kozareva.com

Abstract

Reading comprehension is a challenging task in natural language processing and requires a set of skills to be solved. While current approaches focus on solving the task as a whole, in this paper, we propose to use a neural network 'skill' transfer approach. We transfer knowledge from several lower-level language tasks (skills) including textual entailment, named entity recognition, paraphrase detection and question type classification into the reading comprehension model. We conduct an empirical evaluation and show that transferring language skill knowledge leads to significant improvements for the task with much fewer steps compared to the baseline model. We also show that the skill transfer approach is effective even with small amounts of training data. Another finding of this work is that using token-wise deep label supervision for text classification improves the performance of transfer learning.

1 Introduction

Reading comprehension (RC) is a language understanding task, typically evaluated in a question answering setting, where a system is expected to read a given passage of text (document D) and answer questions (Q) about it. Recent work has introduced several datasets for reading comprehension which gained a lot of attention such as the 'CNN/Daily Mail' [13], Children Book Test [14], Who Did What [33], bAbI [50] and before that MCTest [39]. Most recently SQuAD [38], NewsQA [47] and TriviaQA[17] were created using crowd-sourcing. Reading comprehension has been shown [45, 4, 38] to require different sets of skills such as paraphrase detection, recognition of named entities, natural language inference, etc. The common approach to tackling a higher-level task such as Reading Comprehension is to build a complex neural model that reads a large-scale dataset and tries to learn all required skills at once.

We propose learning the 'skills' required for the task of reading comprehension from existing supervised language tasks. We evaluate the performance of several learned lower-level 'skills' for reading comprehension on the SQuAD [38] dataset by integrating them in a simple neural model. This is in contrast to [8] who propose learning sentence compression representations from a large supervised corpus and transfer the learned knowledge to a set of smaller tasks. Our approach is similar to [27] who used weights pre-trained on machine translation to boost the performance of a very good RC system [52]. Instead of solving a single complex task, we propose using the knowledge learned from multiple supervised, possibly low-scale, language tasks as 'skills'. We propose a simple model that allows to inject learned 'skill' representations easily and analyze the learning behavior of this skill transfer model for reading comprehension. We also experiment with training on smaller parts of the training data (2%, 5%, 10%) to examine the impact of 'skill' transfer on smaller datasets.

2 Method

In this work, we tackle the task of reading comprehension using lower-level supporting 'skill' tasks. To do that, we implement a baseline model to represent the relation between a given question and the story context and enrich the representation by reusing encoder weights from the chosen 'skill' tasks.

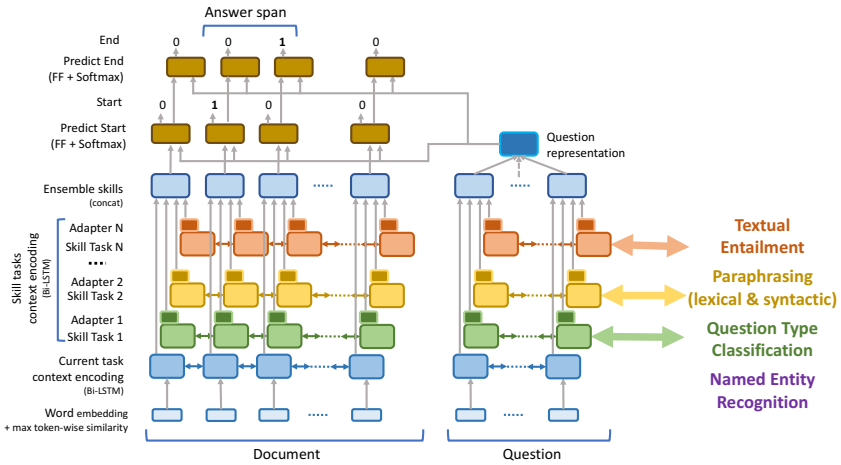


Figure 1: Skillful Reader: Architecture for transferring knowledge from 'skill' language tasks to a reading comprehension model.

Our 'skill' transfer method is visualized in Figure 1 and can be summarized in two main steps:

- Skill Learning: Train context encoder-based (Bi-LSTM) models for several language skill tasks and save the learned encoder weights.
- Neural Skill Transfer: Reuse the learned context encoder skill weights to encode the text context of document and question, in a simple model for the higher-level task (QA/RC).

An overview of our model is shown in Figure 1. It can be considered similar to *progressive neural networks* [42] without the notion of sequential learning of the tasks. We refer to the underlying tasks as skills, following [45], who show that complex tasks like RC require a set of language analysis skills. We show that using such skills, learned from specialized corpora, boosts the performance of a good baseline RC system (i) early in training and (ii) when training on smaller portions (2, 5, and 10 percent) of the original training data.

2.1 Skill Learning

For encoding the skill knowledge from lower-level tasks we first implement simple context encoder models for each low-level task. In this work we implement three types of models for encoding language skill tasks: Sequence Labeling, Text Classification, and Relation Classification.

Sequence Labeling is applied for labeling each token of a given text with a specific category. For this type of encoder model we use a vanilla Bi-directional Long Short-Term Memory [10] architecture, that uses word embeddings as input with a label projection layer with Softmax to predict the sequence labels (2a). While this does not lead to a supreme performance in any sequence-labeling task, it is a stable baseline [25, 21]. We hypothesize that by using a simple architecture for the skill model, we can encode the skill knowledge in the context layer. As a sequence labeling skill, we choose the task of Named Entity Recognition (NER) based on the CoNLL 2012 NER dataset [37]. We use the BIO schema for label encoding, as shown in Figure 2a.

Text Classification is applied in order to categorize a given word token sequence. Given that our RC task is cast as a QA problem, we propose to employ the skill of Question Type Classification, using the TREC Question Classification dataset [23] with 50 classes for training. The task is to classify a given question according to the type of the answer phrase. To learn text classification skills we employ a simple model with Bi-LSTM context encoder, where we apply label supervision on the **token** level. The model is shown in Fig. 2b). That is, instead of retrieving a single vector representation of the

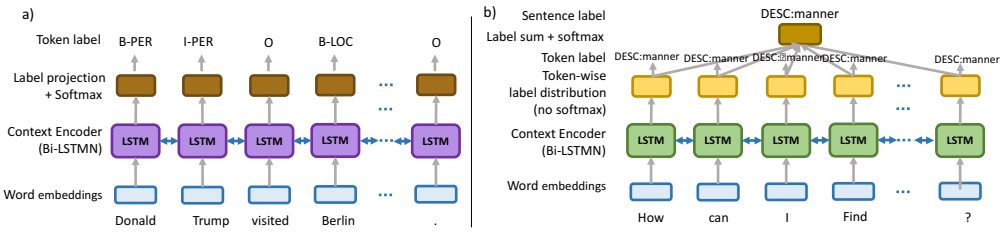


Figure 2: a) Vanilla Bi-LSTM for sequence labeling (NER). b) Text classification (Question Type Classification) with Bi-LSTM context encoder and token-wise label supervision.

sentence (with avg- or max-pooling, etc.) and predicting the label, we project the token context representation $c_{t_{1..n}}$ to the label space (50 classes) $c^{lbl}_{t_{1..n}}$ and sum the label representation predicted for each token, to obtain the label for the sentence $r^{lbl}_{sent} = softmax(\sum c^{lbl}_{t_{1..n}})$. We hypothesize that with lower-level label supervision we can propagate the knowledge expressed by the label to the context representations of specific tokens. This is a form of deep supervision [22], similar to [24].

Relation Classification is used to classify the relation between two arguments represented as text. We implement relation classification skills following the exact *Bi-LSTM max-out* model from Conneau et al. [8], that has been shown to be successful for learning sentence representations.

As a relation classification skill we employ the tasks of Textual Entailment (TE) learned from the Stanford Natural Language Inference (SNLI) corpus [3]. TE is a task that requires a model to classify the entailment relation between two sentences: hypothesis and premise. For instance, the premise *'Dogs like eating food.'* entails the hypothesis *'Animals like eating.'*. Another task that we consider useful for our target task is paraphrase detection over the PPDB 2.0 [35] where the model is required to detect the relation between two phrases in one of the given 6 fine-grained paraphrase classes.

2.2 Model for Reading Comprehension with Skills

We build a simple neural model that uses pre-trained embeddings and word-matching features as input to a bi-directional LSTM context-encoder of document and question and two Bi-LSTM layers for predicting start and end of the answer span. The architecture of the model is shown in Figure 1.

Word embedding input. As an input to the neural model, we use pre-trained 100d Glove [36] word embeddings (WE). We also use two features for each token: the exact word matching feature (em) [49] [6] between each token in the document and the question and the maximum similarity between the word embedding vector of each of the document tokens and each token in the question ($maxsim(w_{d_i}, w_{q_{1..m}}) = \max(\cos(w_{d_i}, w_{q_{1..m}}))$). The WE *maxsim* between two texts has been shown to be helpful for community question answering [5] and discourse relation sense classification [28]. For each token we concatenate the WE and the two features ($w^r_{p_{1..N}} = concat(w^p_{e_i}, maxsim, em)$, *r* means input representation, *p* is a token sequence that can be *d*(document) or *q*(question)) and use them as an input to the context-encoder. For the question, the two features above are set to 1 as in [49].

Context encoding. In particular, we use a Bi-LSTM context encoder represented as $c_{p_{1..N}} = BiLSTM(w^r_{p1..N})$. We refer to a task-specific context-encoder as Enc_{task}.

Context encoder for the current (main) task. For the target task of reading comprehension, we initialize an encoder Enc_{RC} with random weights.

Skill task context encoders. For each skill task, we train a context-encoder model as described in Sec. 2.1. We use the learned weights to initialize the task-specific encoders Enc_{skill}. For the tasks where we employ token label prediction (NER and Question Type Classification), we also concatenate the soft label prediction vectors with the context encoder states: $Enc_{NER/QTC} = concat(c_{p_{1..N}}, c^{lbl}_{p_{1..N}})$.

Adapted representations. Each output from the skill context encoder is projected to a lower dimension using adapter weights [42]: $c^{1..n}_{task} = Enc_{task}(w_{1..n})A_{task} + b^a_{task}$, where A_{task} is a weight matrix for the current task (skill task or target task (RC)) and b^a_{task} is a bias vector.

Ensemble representation. For each token in the document d and question q we concatenate all adapted skill representations c_{task} to the main task representation c_{rc} to obtain the ensemble repre-

sentation $e_p = concat(c_{rc}, c_{ner}, c_{qtc}, c_{te}, c_{ppdb})$, where p is d or q. We represent the question by a weighted representation of its ensemble token vectors: $r_q = sum(e_{q_{1..m}} * softmax(e_{q_{1..m}} W_{qw}))$, where W_{qw} is a weight matrix. We then model interaction between the question representation r_q and each document token e_{d_i} as $r_{d_i 2q} = concat(e_{d_i}, r_q, e_{d_i} * r_q)$.

Answer span prediction. To predict the answer span we predict start and end pointers in the document context. We model the probability of the document tokens being the start of the answer span as $ans_i^{start} = softmax(W_{start} FF(r_{d_i 2q}) + b_{start})$, where W_{start} is a weight matrix and b_{start} is bias. We then model the probability of the document tokens being the end of the answer span as $ans_i^{end} = softmax(W_{end} FF(concat(r_{d_i 2q}, ans_i^{start}, ans_i^{start} * e_{d_i})) + b_{end})$, where W_{end} is a weight matrix and b_{end} is a bias vector. We use dynamic programming to find the answer span (i,j) that maximizes $ans_i^{start} * ans_j^{end}$. FF is a 2-layer (size 100) feed-forward network with $ReLU$[31].

Training details. For all skill tasks and the RC task we use pre-trained Glove word embeddings with size 100. For all tasks, including the target RC task, we train the bi-directional LSTM encoder with output size 256. For the adaption layer we use output size of 100 for the skills and 128 for RC.

3 Related work

Reading comprehension [15] has gained a lot of attention in the last years thanks to large-scale datasets [13][14][33]. More recently the SQuAD [38] dataset offered over 100 thousand crowd-sourced questions to answer questions about Wikipedia. Some of the best performing single models (F1 75-84) on the SQuAD dataset propose token-wise interaction between documents and question Bi-DAF [43], Dynamic-Coattention Networks [52], R-NET [48]. Some models [44][30][46] try to perform reasoning more explicitly using an approach based on memory networks [51, 11]. Some simple neural models [6][49][9] incorporate features to achieve better performance. It has been shown that a big enough dataset [1] can provide enough knowledge to allow a simple neural model [19] to achieve human performance. However, in practice, having a huge dataset is not always an option. So another approach can be to transfer knowledge [18] from another dataset of the same task or from a less related task such as machine translation [27]. Indeed almost all recent neural models use a form of transfer learning by incorporating word embeddings, such as [29][36], as input. Some recent models [34] even use the task of question answering to learn better embeddings. Transfer Learning with neural models has been proposed in NLP initially by [7] and has been encouraged as a way of sharing representations between tasks [2]. It can be performed jointly on multiple tasks [40] which includes learning linguistic tasks in a hierarchical fashion [41] on many levels [12] or even perform the knowledge transfer between tasks from different modalities [20]. In this work we propose a generic and modular approach to learning a set of relevant 'skill' tasks and transferring this knowledge to a target task, here the problem of reading comprehension.

4 Experiments and results

In this work we examine the impact of transferring knowledge from several 'skill' tasks to the task of Reading Comprehension. The assumption is that the transfer of skill knowledge should improve the learning of the target task (RC) and allows for using smaller training sets and fewer training steps. To examine this impact we run several experiments: adding single skill tasks to the RC task, adding all tasks, and ablation of tasks.

Training on the full training set. We use the SQuAD [38] train dataset for training and the publicly available dev set for evaluation. We do not aim for state-of-the art performance but focus on the impact of injecting skill knowledge. In Table 1 we show evaluation results with single tasks and ablation of tasks, w/ and w/o fine tuning of the skill parameters. Figure 3 shows the results in different training steps, with different skills. It shows that individual skills and all skills jointly show a noticeable impact in the early training stages compared to a model without skills.

Training on small parts of the train data. Figure 5 shows results with training on different sizes of the train data. 2% of the train data contains 378 paragraphs, 2512 questions, with 88k tokens in total. We show that with less data (2%, 5% of the full train set), employing skill tasks shows high impact, reaching the best result compared to 'No skills' or 'Random skill weights' setups in only 1000 steps.

Token-wise label supervision. Figure 4 analyzes the impact of token-wise label prediction vs. sentence-wise label prediction with Question Type Classification. We show that token supervision clearly outperforms sentence label supervision in early training phases.

Setup	Skill fine-tuning F-score	EM	No skill fine-tuning F-Score	EM
RC only (no skills)	59.41	46.90	**59.66**	**46.80**
+ only TE	**61.67**	**49.12**	59.40	46.47
+ only QC	60.94	48.68	57.80	44.39
+ only PPDB	60.82	48.71	58.23	45.25
+ only NER	60.65	48.45	58.17	44.70
all (RC + all skills)	60.92	48.70	58.30	45.51
all - PPDB	61.11	48.83	57.87	45.19
all - QC	60.91	48.52	57.28	45.17
all - TE	60.86	48.81	58.48	45.73
all - NER	60.81	48.55	56.99	44.07

Table 1: Results for transferring knowledge from skill tasks. 'Fine-tuning': parameters of the skill tasks are fine-tuned during training.'EM' shows the results for Exact Match with the gold answers. We show evaluation results on the dev set of SQuAD [38]. Trained with 51k steps on the train set.

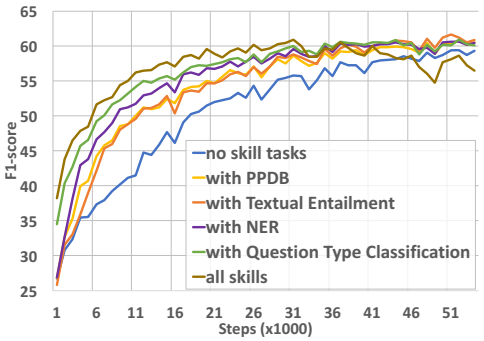


Figure 3: Results for single skill tasks combined with the QA-encoder. (w/ skill fine-tuning)

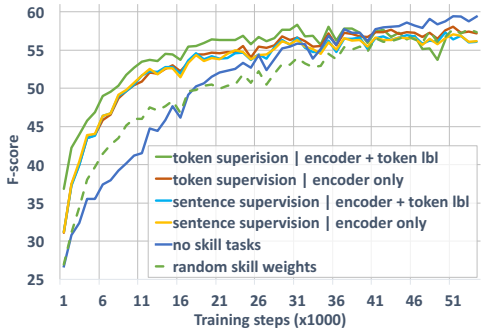


Figure 4: Results with QTC skill, w/ and w/o token label supervision. (w/o fine-tuning)

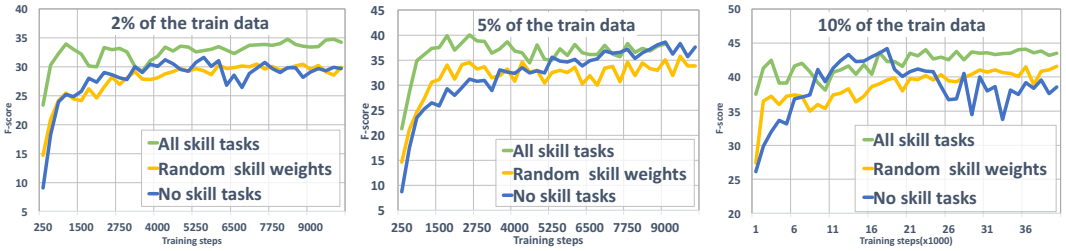


Figure 5: Results for training with different sizes of the training data (2%, 5%, 10%) and evaluated on the dev set. 'Rand. skill weights' is 'All skill tasks' model with random weights. (w/ fine-tuning)

5 Conclusion and future work

In this work, we show the impact of injecting knowledge from supervised language skill tasks into a reading comprehension model. We observe noticeable gains of performance in both, early training stages and when using small training data. While for some domains, currently large training sets are being built, in others such as [39] this is not the case. Beyond performance issues, using skill tasks as proposed in this work can be applied as a tool for analyzing which specific skills are required for reading comprehension (or other tasks) and also the contribution of specific skills for a particular dataset and problem formulation, without having to conduct manual annotation as in [45]. Another finding is that token-wise deep label supervision for QTC is beneficial for reading comprehension in a QA setting. In future work we plan to transfer knowledge from other tasks i.a. Discourse Relations [16] [32], Semantic Role Labeling [26]. We also want to experiment with different models of integrating the learned skills, also for other tasks. We also plan to train all the tasks jointly, in multi-task fashion, where shared parameters are fine-tuned on the skill tasks and the target task.

Acknowledgments

Most of this work is performed during Todor Mihaylov's internship at Amazon, AWS Deep Learning. This work is partly supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1.

References

[1] Ondrej Bajgar, Rudolf Kadlec, and Jan Kleindienst. Embracing data abundance: BookTest Dataset for Reading Comprehension. 2016. URL http://arxiv.org/abs/1610.00956.

[2] Yoshua Bengio. Deep Learning of Representations for Unsupervised and Transfer Learning. *JMLR: Workshop and Conference Proceedings 7*, 7:1–20, 2011.

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. pages 632–642, 2015. doi: 10.18653/v1/D15-1075. URL http://www.aclweb.org/anthology/D15-1075.

[4] Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/Daily Mail reading comprehension task. 2016.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1171. URL http://www.aclweb.org/anthology/P17-1171.

[6] Fisch Adam Chen, Danqi, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. 2016.

[7] Ronan Collobert and Jason Weston. A unified architecture for natural language processing. *Proceedings of the 25th international conference on Machine learning - ICML '08*, 20(1):160–167, 2008. ISSN 07224028. doi: 10.1145/1390156.1390177. URL http://portal.acm.org/citation.cfm?id=1390177{%}5Cnhttp://portal.acm.org/citation.cfm?doid=1390156.1390177.

[8] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. may 2017. URL http://arxiv.org/abs/1705.02364.

[9] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1168. URL http://www.aclweb.org/anthology/P17-1168.

[10] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[11] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. *Arxiv*, 2014. URL http://arxiv.org/abs/1410.5401.

[12] Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. pages 446–456, 2017. URL http://www.aclweb.org/anthology/D17-1046.

[13] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. 2015. URL http://arxiv.org/abs/1506.03340.

[14] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1511.02301.

[15] Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. Deep read: A reading comprehension system. 1999. URL http://www.aclweb.org/anthology/P99-1042.

[16] Yacine Jernite, Samuel R Bowman, and David Sontag. Discourse-Based Objectives for Fast Unsupervised Sentence Representation Learning. URL https://arxiv.org/pdf/1705.00557.pdf.

[17] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics (ACL) 2017. URL http://aclweb.org/anthology/P17-1147.

[18] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst.

[19] Rudolf Kadlec, Martin Schmid, Ondřej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. pages 908–918, 2016. doi: 10.18653/v1/P16-1086. URL http://www.aclweb.org/anthology/P16-1086.

[20] Lukasz Kaiser, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *CoRR*, abs/1706.05137, 2017. URL http://arxiv.org/abs/1706.05137.

[21] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. pages 260–270, 2016. doi: 10.18653/v1/N16-1030. URL http://www.aclweb.org/anthology/N16-1030.

[22] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. pages 562–570, 2015.

[23] Xin Li and Dan Roth. Learning question classifiers. 2002. URL http://www.aclweb.org/anthology/C02-1150.

[24] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. Learning to diagnose with lstm recurrent neural networks. URL https://arxiv.org/pdf/1511.03677.pdf.

[25] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. pages 1064–1074, 2016. doi: 10.18653/v1/P16-1101. URL http://www.aclweb.org/anthology/P16-1101.

[26] Ana Marasović and Anette Frank. SRL4ORL: Improving Opinion Role Labelling using Multi-task Learning with Semantic Role Labeling. nov 2017. URL http://arxiv.org/abs/1711.00768.

[27] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. *CoRR*, abs/1708.00107, 2017. URL http://arxiv.org/abs/1708.00107.

[28] Todor Mihaylov and Anette Frank. Discourse relation sense classification using cross-argument semantic similarity based on word embeddings. In *Proceedings of the CoNLL-16 shared task*, pages 100–107. Association for Computational Linguistics, 2016. doi: 10.18653/v1/K16-2014. URL http://www.aclweb.org/anthology/K16-2014.

[29] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '13, pages 746–751, Atlanta, Georgia, USA, 2013. URL http://www.aclweb.org/anthology/N13-1090.

[30] Tsendsuren Munkhdalai and Hong Yu. Reasoning with Memory Augmented Neural Networks for Language Comprehension. *International Conference on Learning Representations (ICLR) 2017*, pages 1–13, oct 2016. URL http://arxiv.org/abs/1610.06454.

[31] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress, 2010. URL http://www.icml2010.org/papers/432.pdf.

[32] Allen Nie, Erin D. Bennett, and Noah D. Goodman. Dissent: Sentence representation learning from explicit discourse relations. *CoRR*, abs/1710.04334, 2017. URL http://arxiv.org/abs/1710.04334.

[33] Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Who did What: A Large-Scale Person-Centered Cloze Dataset. *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2016*, pages 2230–2235, 2016. URL http://arxiv.org/abs/1608.05457.

[34] Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. MEMEN: Multi-layer Embedding with Memory Networks for Machine Comprehension. 2017.

[35] Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. pages 425–430, 2015. doi: 10.3115/v1/P15-2070. URL http://www.aclweb.org/anthology/P15-2070.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. pages 1532–1543, 2014. doi: 10.3115/v1/D14-1162. URL http://www.aclweb.org/anthology/D14-1162.

[37] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. pages 1–40. Association for Computational Linguistics, 2012. URL http://www.aclweb.org/anthology/W12-4501.

[38] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2016*, 2016. URL http://arxiv.org/abs/1606.05250.

[39] Matthew Richardson, Christopher J C Burges, and Erin Renshaw. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. *Conference on Empirical Methods in Natural Language Processing (EMNLP) 2013*, pages 193–203, 2013.

[40] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. (May), 2017.

[41] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098.

[42] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive Neural Networks. 2016. URL http://arxiv.org/abs/1606.04671.

[43] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL http://arxiv.org/abs/1611.01603.

[44] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. *CoRR*, abs/1609.05284, 2016. URL http://arxiv.org/abs/1609.05284.

[45] Saku Sugawara, Hikaru Yokono, and Akiko Aizawa. Prerequisite skills for reading comprehension: Multi-perspective analysis of mctest datasets and systems. pages 3089–3096, 2017.

[46] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. pages 2440–2448, 2015. URL http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf.

[47] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830, 2016. URL http://arxiv.org/abs/1611.09830.

[48] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. pages 189–198, 2017. doi: 10.18653/v1/P17-1018. URL http://www.aclweb.org/anthology/P17-1018.

[49] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. Making neural qa as simple as possible but not simpler. pages 271–280, 2017. doi: 10.18653/v1/K17-1028. URL http://www.aclweb.org/anthology/K17-1028.

[50] Jason Weston, Antoine Bordes, Sumit Chopra, Tomas Mikolov, and Alexander M. Rush. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. *arXiv Prepr.*, 2015. ISSN 1502.05698. doi: 10.1016/j.jpowsour.2014.09.131.

[51] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *International Conference on Learning Representations (ICLR), 2015*, 2015.

[52] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016. URL http://arxiv.org/abs/1611.01604.

Beyond Fine Tuning: Adding capacity to leverage few labels

Nathan O. Hodas[†] **Kyle Shaffer**[*] **Artem Yankov**[*] **& Courtney D. Corley**[†]
Pacific Northwest National Laboratory
Washington, USA
{kyle.shaffer,artem.yankov,court,nathan.hodas}@pnnl.gov

Aryk Anderson[‡]
Eastern Washington University
Cheney, Washington
aryk.anderson@eagles.ewu.edu

Abstract

In this paper we present a technique to train neural network models on small amounts of data. Current methods for training neural networks on small amounts of rich data typically rely on strategies such as fine-tuning a pre-trained neural network or the use of domain-specific hand-engineered features. Here we take the approach of treating network layers, or entire networks, as modules and combine pre-trained modules with untrained modules, to learn the shift in distributions between data sets. The central impact of using a modular approach comes from adding new representations to a network, as opposed to replacing representations via fine-tuning. Using this technique, we are able surpass results using standard fine-tuning transfer learning approaches, and we are also able to significantly increase performance over such approaches when using smaller amounts of data.

1 Introduction

Training generalizable models using only a small amount of data has proved a significant challenge in the field of machine learning since its inception. This is especially true when using artificial neural networks, with millions or billions of parameters. Conventional wisdom gleaned from the surge in popularity of neural network models indicates that extremely large quantities of data are required for these models to be effectively trained. Indeed the work of Kriszhevsky (8) has commonly been cited as only being possible through the development of ImageNet (16). As practitioners and researchers continue to explore neural networks in more specialized domains, the volume of available labeled data also narrows. Although training methods have improved, it is still difficult to train deep learning models on small quantities of data, such as only tens or hundreds of examples.

The current paradigm for solving this problem has come through the use of pre-trained neural networks. (1) were able to show that transfer of knowledge in networks could be achieved by first training a neural network on a domain for which there is a large amount of data and then retraining that network on a related but different domain via fine-tuning its weights. Though this approach demonstrated promising results on small data, these models do not retain the ability to function as previously trained. That is, these models end up fine tuning their weights to the new learning task, forgetting many of the important features learned from the previous domain.

[*] Seattle, WA
[†] Richland, WA; AA and NOH contributed equally

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

In this paper we present our neuro-modular approach to fine-tuning. The first contribution is demonstrating a novel topological approach to fine-tuning. We quantitatively compare traditional fine-tuning with our modular approach, showing that our approach is more accurate on small amounts of data (<100 examples per class). We also demonstrate how to improve classification in a number of experiments, including CIFAR-100, text classification, and fine-grained image classification, all with limited data.

2 Related Work

Transferring knowledge from a source domain to a target domain is an important challenge in machine learning research. Many shallow methods have been published, those that learn feature invariant representations or by approximating a value without using an instance's label (14; 19; 15; 24; 22; 3). More recent deep transfer learning methods enable identification of variational factors in the data and align them to disparate domain distributions (20; 9; 2; 21). Mesnil et al. (11) presents the Unsupervised and Transfer Learning Challenge and discusses the important advances that are needed for representation learning, and the importance of deep learning in transfer learning. Oquab et al. (13) applied these techniques to mid-level image representations using CNNs. Specifically, they showed that image representations learned in visual recognition tasks (ImageNet) can be transferred to other visual recognition tasks (Pascal VOC) efficiently. Further study regarding the transferability of features by (23) showed surprising results that features from distant tasks perform better than random features and that difficulties arise when optimizing splitting networks between co-adapted neurons. We build on these results by leveraging existing representations to transfer to target domains without overwriting the pre-trained models through standard fine-tuning approaches.

The progressive network architecture of (17) shares a number of qualities with our work. Both the results we present here and the progressive networks allow neural networks to extend their knowledge without forgetting previous information. In addition, Montone et al. (12) discusses a semi-modular approach. Montone et al. also froze the weights of the original network, although it did not demonstrate success on the small data regime. Our approach provides the user with a novel fine-tuning approach for learning on small data. Our modular approach detailed here allows small data to adapt to different domains. Our architecture also complements existing network building strategies, such as downloading pre-trained neural networks to then be fine-tuned for domain adaptation. The modular approach presents a particular advantage when large volumes of training data for the base network may not be available, i.e. when you can not retrain an entire network from scratch. Our approach addresses a key need for large deep learning pre-trained models to be adapted to new tasks using only small amounts of data.

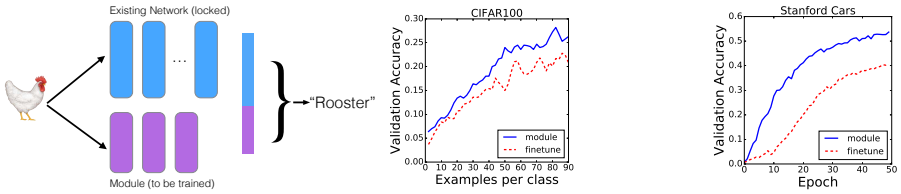
3 Results

Generically, modular neural networks are directed graphs of pre-trained networks linked together with auxiliary, untrained networks. As depicted in Fig. 1a, only the new components of the resulting network is trained. The architecture could take the form of simply placing two networks in parallel (the two-towers approach), shown in Fig. 1a.

More formally, we can describe any neural network as a differentiable function G taking input $\mathbf{x}$, and parameterized by weight matrix θ, $G(\mathbf{x};\theta)$. In our modular architecture, we denote at least one additional network $G'(\mathbf{x};\theta')$. It is crucial to note that the overall architecture of G' need not be identical to that of G, and this allows considerable flexibility for the resulting architecture to learn complementary representations from each sub-network. Each of these networks, G and G' are treated as feature extractors, lacking classifier layers for making final predictions. Our final modular architecture combines the output of the base network G and the modular network, G', via tensor concatenation, which is then fed to a classifier layer for final predictions, as in Equation 1:

$$H = f\big(G(\mathbf{x};\theta) \oplus G'(\mathbf{x};\theta');\theta_f\big). \tag{1}$$

Tuning H (and minimizing the resulting loss) can be broken down into error due to $\frac{\partial G}{\partial \theta}$, $\frac{\partial G'}{\partial \theta'}$, and $\frac{\partial f}{\partial \theta_f}$. If the domain of the new data closely aligns with the original domain of the fully trained G, then we may presume $|\frac{\partial G}{\partial \theta}| \ll |\frac{\partial G'}{\partial \theta'}|$. Thus, for computational expedience, for our modular approach



(a) (b) (c)

Figure 1: (a) An example module setup. By explicitly preserving the original representation learned on pre-trained net, the module is able to learn more robust features using fewer examples than naive fine-tuning. (b) Comparison of modular approach vs. fine-tuning based on amount of training data. (c) Comparison of validation accuracy on the Stanford Cars data. Training on full data

we freeze θ and only tune θ' and θ_f. This allows the network as a whole to retain the original representational power of the pre-trained network.

This allows the modular approach to more robustly handle small data than naive fine-tuning. To demonstrate this, we trained a network on CIFAR-10 data. The CIFAR-10 network was trained until it was 88.9% accurate, using the network in (5) with 3 residual units, for a total of 28 layers. We then compared two approaches. For the first approach, we simply fine tuned the CIFAR-10 network using training data from the CIFAR-100 dataset and replacing the final softmax. Next, we froze the original CIFAR-10 network and added an identical copy as a module, which would be trained on the same batches of data as the first approach. That is, we have: Network 1 – fine-tuning the base network and Network 2 – freezing the base network and fine-tuning a module. Network 1 and Network 2 have an identical number of weights and those weights have the same starting value. These two approaches are presented in eqs. 2 and 3 below.

$$\hat{y} = \sigma(G(\mathbf{x}; \theta_g + \Delta\theta_{g'})) \tag{2}$$

$$\hat{y'} = \sigma(H(\mathbf{x})) \tag{3}$$

where $\hat{y}$ denotes predictions made from a fine-tuned network, $\hat{y}'$ denotes predictions made from our modular architecture, and $\sigma(\cdot)$ is the softmax function.[3] G denotes the fine-tuned network with pre-trained weights θ_g learned from CIFAR-10 and H denotes a modular network as in Eq. 1 above.

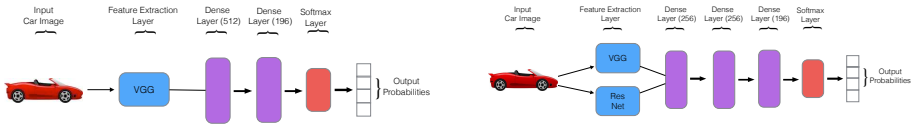
To train, we used the ADAM optimization algorithm (6)). We added an activation L2 regularization of $1e^{-6}$ to the module to help break degeneracy. We used batches of 200, where each batch contained two images per class. Each batch was iterated over five times, before the next batch was used. This iteration allowed simulating multiple epochs over small data. We recorded the results of the performance on the test set after each batch, in Fig. 1b.

We observe that for all amounts of training data, but particularly for small amounts of training data, the modular approach outperforms traditional fine-tuning. Of course, we chose to make the module a complete copy of the original CIFAR-10 network. This ensured we could compare with the same number of weights, same initialization, same data, etc. Further research will certainly reveal more compact module networks that outperform our example. We also tried retraining both the original network and the module, to see if the improvement was due simply to the increased capacity, and noted no improvement over training the module and freezing the original network. The module learns additional filters that boost performance on the transfer task.

To explore modules on a more real-world relevant task, we applied this approach to an imagenet-trained network (16). The Stanford Cars data set (7), which features 16,185 images of 196 classes of cars, is an example of a data set for fine-grained categorization. Rather than train a classifier to distinguish between fundamentally different objects like horses and planes, as required in the Large Scale Visual Recognition Challenge (16)), fine-grained categorization requires the classifier to learn subtle differences in variations of the same entity. For example, a classifier trained on the Stanford Cars data set would have to learn distinguishing features between a BMW X6 SUV from 2012 and an Isuzu Ascender SUV from 2008.

In these Stanford Cars experiments, both models utilize a transfer learning approach by leveraging the feature extraction output layers from the VGG16 model (18). The “fine-tuned" model passes the

[3]In this training setup, there is one softmax per category.



(a) (b)

Figure 2: (a) Network architecture used for Stanford Cars fine-tuned model. (b) ImageNet-based network architecture used for Stanford Cars module model.

VGG16 features to a fully connected layer of length 512 followed by a softmax layer of length 196, as seen in Fig. 2a. Gradient descent via RMSProp is used to train the dense layers (RMSProp performed better than ADAM on this task). The “module" model merges the fixed VGG16 features with a ResNet (4) model, whose output is then fed to two consecutive dense layers of length 256 and finally a softmax layer of length 196. The module model architecture is shown in Fig. 2b. Again, RMSProp is used to train ResNet and the dense layer weights after the two component models are merged, however the VGG feature-extraction weights remain unchanged ensuring that this architecture fully leverages the representational power of this sub-network.

As seen in Fig. 1c, after 50 epochs the module model appears to significantly outperform the fine-tuned model in validation accuracy. However, it should be noted that while the module model carries 19,537,990 trainable parameters the fine-tuned model only has 12,946,116 parameters. Thus, the generic module approach does not require any specific number of additional parameters. Furthermore, no hyperparameter optimization is performed on either model.

We further investigate the effects of our modular network approach by applying this method to a different modeling problem - text classification. Similar to image data, text represents an unstructured data-type that often exhibits long-term and interconnected dependencies that are difficult to model with simpler classifiers. Whereas in the case of images neighboring pixels may represent semantically related concepts or objects, in text words may exhibit long-term semantic or syntactic dependencies that can be modeled sequentially. These characteristics make text classification particularly well-suited to recurrent neural networks such as long short-term memory (LSTM) networks, but these learning methods typically require a great deal of data to be learned efficiently and to avoid overfitting.

To test our methodology, we evaluate a modular recurrent network against two individual recurrent neural networks, and an ensemble of these individual networks on the IMDB sentiment dataset.[4] Previous work has shown deep learning methods to be effective at sentiment classification performance on this dataset (10), however we add to this past work by presenting an analysis that demonstrates the effectiveness of modular networks in the case of extremely small training sets. To this end, we sample only 500 training examples from the original 25,000 available in the full training set, and evaluate on the full 25,000 validation examples. We use the same 500 training examples for each model evaluated in our experiments for consistency, and report accuracy for each model on the full validation set.

We evaluate four models in our text-classification experiments, two individual LSTM networks, an ensemble of these two LSTM networks, and a final model which is our modular recurrent network. The first model consists of three layers - an initial layer that projects sequences of words into an embedding space, a second LSTM layer with 32 units, and a final sigmoid layer for computing the probability of the text belonging to the positive class. Our second model is identical to the first except that we initialize the weights of the embedding layer using pre-trained GloVe word vectors.[5] In particular, we use 100-dimensional vectors learned from a 2014 version of Wikipedia. We also experimented with freezing the weights in the embedding layer of this second model, but found better performance when allowing this layer to be tuned in each epoch of training. For our final baseline, we ensemble the two individual models detailed above by training both networks in tandem on the same training set, and fitting a logistic classifier layer on the prediction probabilities of the two component networks.

To construct our modular network, we take the embedding and LSTM layers from our individual networks, and concatenate the output of both LSTM layers into a single tensor layer in the middle of

[4] http://ai.stanford.edu/~amaas/data/sentiment/

[5] http://nlp.stanford.edu/projects/glove/

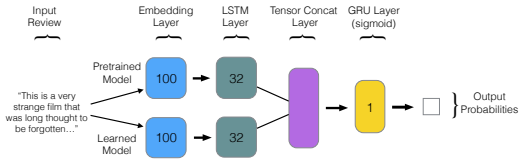


Figure 3: Diagram of architecture for our modular recurrent text classification network. Dimensionality for embedding layers and number of units for all other layers are given in boxes denoting those layers.

MODEL	**Base.**	**Pretrain**	**Ens.**	**Module**
ACCURACY (%)	61.9	64.9	65.3	**69.6**

Table 1: Accuracy results for text classification experiments, *u*sing only 500 training examples. Results are shown for the baseline model (**Base.**), pre-trained (GloVe) model (**Pretrain**), ensembled model (**Ens.**) and modular model (**Module**).

our modular network. Additionally, we modify the output of each of these component LSTM layers by forcing each to output a weight matrix that tracks the state of the LSTM layer across all timesteps throughout the dataset. In this way, we seek to fully leverage the sequential dependencies learned by these layers, and this method outperforms the simpler alternative method of simply outputting the final state of each of the LSTM layers. We then feed this concatenated layer to a gated recurrent unit (GRU) layer with a sigmoid activation function for calculation of class probabilities. We experimented with an LSTM and densely connected layers after the tensor concatenation layer, but found best performance with the GRU. All models were optimized with the ADAM algorithm, and trained for 15 epochs. An outline of this architecture can be seen in Figure 3.

Here, we report results for our classification experiments with the three networks described above. We see an accuracy of 61.9% for our first model which is trained entirely from the data without any pre-training. This is significantly lower than previously reported results, however we are training on only 2% of the available data to test our method's application to small training sets. We see better performance in terms of accuracy (64.9%) from our second model initialized with GloVe vectors. This seems to indicate that despite being trained on more formally written language in Wikipedia, these vectors can still boost performance on a task modeling text that is inherently subjective and opinion-based. The ensemble of these two individual networks further boosts accuracy to 65.3%, demonstrating that even simple combinations of these networks appears to aid model learning, despite receiving very little training data. Finally, we see an accuracy of 69.6% from our modular network, an increase of over 4% accuracy over the next best performing model. Because weight initializations of recurrent networks can greatly affect model performance, we ran the classification experiments with our modular network 10 times, and report the average accuracy across these 10 runs. As can be seen here, our modular approach improves on the best performing individual network, suggesting that this approach is useful for text classification even with very small amounts of training data. Further, the fact that our modular approach improves over an ensemble of the individual networks suggests that our modular architecture learns specific representations that are more advantageous than simply leveraging multiple models via standard ensembling techniques.

4 Conclusions

We have presented a neuro-modular approach to transfer learning. By mixing pre-trained neural networks (that have fixed weights) with networks to be trained on the specific domain data, we are able to learn the shift in distributions between data sets. As we have shown, often the new modules learn features that complement the features previously learned in the pre-trained network. We have shown that our approach out-performs traditional fine-tuning, particularly when the amount of training data is small – only tens of examples per class. Although it has a certain amount of computational complexity, its implementation simplicity and performance benefit over traditional fine-tuning more than justify the cost in many applications.

References

[1] Bengio, Yoshua et al. Deep learning of representations for unsupervised and transfer learning. *ICML Unsupervised and Transfer Learning*, 27:17–36, 2012.

[2] Ganin, Yaroslav and Lempitsky, Victor. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[3] Gong, Mingming, Zhang, Kun, Liu, Tongliang, Tao, Dacheng, Glymour, Clark, and Schölkopf, Bernhard. Domain adaptation with conditional transferable components. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2839–2848, 2016.

[4] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[5] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016.

[6] Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Krause, Jonathan, Stark, Michael, Deng, Jia, and Fei-Fei, Li. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 554–561, 2013.

[8] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[9] Long, Mingsheng, Cao, Yue, Wang, Jianmin, and Jordan, Michael. Learning transferable features with deep adaptation networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 97–105, 2015.

[10] Maas, Andrew L., Daly, Raymond E., Pham, Peter T., Huang, Dan, Ng, Andrew Y., and Potts, Christopher. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

[11] Mesnil, Grégoire, Dauphin, Yann, Glorot, Xavier, Rifai, Salah, Bengio, Yoshua, Goodfellow, Ian J, Lavoie, Erick, Muller, Xavier, Desjardins, Guillaume, Warde-Farley, David, et al. Unsupervised and transfer learning challenge: a deep learning approach. *ICML Unsupervised and Transfer Learning*, 27:97–110, 2012.

[12] Montone, Guglielmo, O'Regan, J Kevin, and Terekhov, Alexander V. The usefulness of past knowledge when learning a new task in deep neural networks. 2015.

[13] Oquab, Maxime, Bottou, Leon, Laptev, Ivan, and Sivic, Josef. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1717–1724, 2014.

[14] Pan, Sinno Jialin and Yang, Qiang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[15] Pan, Sinno Jialin, Tsang, Ivor W, Kwok, James T, and Yang, Qiang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.

[16] Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[17] Rusu, Andrei A, Rabinowitz, Neil C, Desjardins, Guillaume, Soyer, Hubert, Kirkpatrick, James, Kavukcuoglu, Koray, Pascanu, Razvan, and Hadsell, Raia. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[18] Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Sugiyama, Masashi, Nakajima, Shinichi, Kashima, Hisashi, Buenau, Paul V, and Kawanabe, Motoaki. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pp. 1433–1440, 2008.

[20] Tzeng, Eric, Hoffman, Judy, Zhang, Ning, Saenko, Kate, and Darrell, Trevor. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[21] Tzeng, Eric, Hoffman, Judy, Darrell, Trevor, and Saenko, Kate. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4068–4076, 2015.

[22] Wang, Xuezhi and Schneider, Jeff. Flexible transfer learning under support and model shift. In *Advances in Neural Information Processing Systems*, pp. 1898–1906, 2014.

[23] Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, and Lipson, Hod. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[24] Zhang, Kun, Schölkopf, Bernhard, Muandet, Krikamol, and Wang, Zhikun. Domain adaptation under target and conditional shift. In *ICML (3)*, pp. 819–827, 2013.

Sparse Coding on Stereo Video for Object Detection

Sheng Y. Lundquist
Portland State University
New Mexico Consortium
shenglundquist@gmail.com

Melanie Mitchell
Portland State University
Santa Fe Institute

Garrett T. Kenyon
Los Alamos National Laboratory
New Mexico Consortium

Abstract

Deep Convolutional Neural Networks (DCNN) require millions of labeled training examples for image classification and object detection tasks, which restrict these models to domains where such datasets are available. In this paper, we explore the use of unsupervised sparse coding applied to stereo-video data to help alleviate the need for large amounts of labeled data. We show that replacing a typical supervised convolutional layer with an unsupervised sparse-coding layer within a DCNN allows for better performance on a car detection task when only a limited number of labeled training examples is available. Furthermore, the network that incorporates sparse coding allows for more consistent performance over varying initializations and ordering of training examples when compared to a fully supervised DCNN. Finally, we compare activations between the unsupervised sparse-coding layer and the supervised convolutional layer, and show that the sparse representation exhibits an encoding that is depth selective, whereas encodings from the convolutional layer do not exhibit such selectivity. These result indicates promise for using unsupervised sparse-coding approaches in real-world computer vision tasks in domains with limited labeled training data.

1 Introduction

Over the last decade, Deep Convolutional Neural Networks (DCNN) trained with supervised learning have emerged as the dominant paradigm for computer vision. These networks have shown impressive results on computer vision tasks such as image labeling and object detection (e.g., [18, 20]). However, one drawback of DCNNs is that they rely on large collections of training data that have been annotated by humans, e.g., ImageNet [3]. As such, DCNNs are restricted to domains for which large datasets of labeled examples are available.

We explore the use of unsupervised sparse coding within a supervised DCNN to help alleviate the need for an abundance of training labels. Typically, the encodings (defined as the collection of activation values calculated from a layer) in DCNNs are computed via convolutions followed by a nonlinearity, and for a given task the weights are trained via supervised learning. In contrast, sparse coding [15] aims to infer efficient, non-redundant encodings of a given input (e.g., photographs and videos), and the weights are trained via an unsupervised method. Inspired by theories of efficient coding in neural computation [1], sparse coding has been shown to exhibit similar properties to biological neurons in early stages of mammalian visual processing [15, 22]. Modeling the accuracy and precision of biology in visual perception could provide novel insights into computer vision tasks, such as object detection. Additionally, the efficiency in representation exhibited by sparse coding can be advantageous in specialized, non-Von Neumann hardware architectures. Indeed, there has been work in implementations of sparse coding on low-power neuromorphic [9] and quantum [13] hardware.

One domain in which non-redundant encoding should be useful is in multi-view sensing. An efficient encoding should account for correlated offsets between different views of visual features. These

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

offsets represent disparity in stereo images and optic flow in consecutive frames from a video. It follows that an encoding that accounts for such offsets should have some notion of depth [5, 16].

In this paper, we compare two types of convolutional network models that differ only in the first layer: (1) a *sparse-coding* network, in which the weights and activations in the first layer are computed via unsupervised sparse coding, and (2) a *supervised* network, in which the first-layer weights are learned via supervised training and activations are computed using these layer weights. In both network models, the weights and activations in all subsequent layers are computed via supervised convolutional layers. We show that sparse-coding networks is able to achieve better performance on a vehicle detection task on stereo-video data with a minimal amount of training labels. Additionally, we show that the performance of sparse-coding networks is more consistent—i.e., more robust to randomized order of training data and random initializations—than comparable supervised networks. Finally, we show that activations in the first (sparse-coding) layer in our sparse-coding networks are depth selective, which may provide an explanation for the differences in performance we observe in this study.

1.1 Related Work

Lundquist et al. [11] demonstrated that representations of stereo images obtained through sparse coding allow for an encoding that achieves better performance than a convolutional layer in the task of pixel-wise depth estimation. The authors show that the sparse encoding is inherently depth selective, whereas the convolutional encoding is not. In this paper, we extend this work to encode stereo-video clips and compare encodings on a vehicle-detection task.

A closely related study by Coates et al. [2] compared unsupervised and supervised methods with two-layer networks on an image classification task. The authors demonstrated that an unsupervised layer does not outperform a comparable layer with a convolutional encoding. While our experimental results agree with this finding in the case of two layers, we find that limiting the number of labeled training examples or adding an additional supervised layer in the sparse-coding network allows the network to outperform the supervised network.

Recent work by Lotter et al. [10] shares the motivation of utilizing unsupervised learning to alleviate the need for labeled training data. Specifically, the authors use unsupervised learning to predict future frames of a video. They additionally show that their network achieves better performance than standard DCNNs when each is trained on only a limited amount of training data. In contrast to future frame prediction, our work aims to achieve image representation through sparse coding. Additionally, Lotter et al. uses a recurrent neural network [7] as the backbone of their network, whereas we use sparse coding as the unsupervised learning algorithm.

Other work [4, 17] explores the use of unsupervised learning techniques within a supervised network. However, most work in this area does not explore natural scenes (instead, focusing on datasets such as MNIST for handwritten digit recognition). Here, we extend this work to the domain of stereo video captured "in the wild". Additionally, we explicitly compare performance between the use of unsupervised learning versus supervised learning within a DCNN.

There has been other work in unsupervised learning of multi-view data [8, 12, 14]. In contrast, our work aims to explicitly compare unsupervised learning to supervised learning for the task of vehicle detection in multi-view data.

2 Sparse Coding

Sparse coding aims to represent an input (e.g., an image) as a linear combination of basis vectors drawn from a provided set (referred to as a *dictionary*) such that the original signal is recoverable with minimal degradation. Each basis vector is weighted by a scalar coefficient (referred to as an *activation*), and the set of activations are taken to be the encoding of the input. Sparse coding constrains the activations to be sparse (i.e., to have few nonzero activations), such that resulting activations are non-redundant. Here, inferring the sparse set of activations is an optimization problem, unlike encodings calculated via supervised convolutional layers.

Formally, sparse coding aims to minimize the cost function

$$J(\boldsymbol{I}, \boldsymbol{a}, \boldsymbol{\Phi}) = \frac{1}{2}(\overbrace{\|\boldsymbol{I} - \boldsymbol{a} \circledast \boldsymbol{\Phi}\|_2}^{\text{Reconstruction error}})^2 + \lambda \overbrace{\|\boldsymbol{a}\|_1}^{\text{Sparsity term}} . \quad (1)$$

Specifically, the algorithm aims to minimize the difference between a given input $\boldsymbol{I}$ and a reconstruction, where the difference is measured by Euclidean distance (i.e., $\|\cdot\|_2$, or the L_2 norm). The reconstruction $\boldsymbol{a} \circledast \boldsymbol{\Phi}$ is calculated via a linear combination of basis vectors drawn from a dictionary $\boldsymbol{\Phi}$, weighted by activation coefficients $\boldsymbol{a}$. Here, $\circledast$ denotes the transposed convolution operation [21]. The sparsity term constrains the activations $\boldsymbol{a}$ to be sparse, by measuring the sum of the absolute value of $\boldsymbol{a}$ (i.e., $\|\cdot\|_1$, or the L_1 norm)[1]. λ is a hyperparameter that controls the trade-off between the reconstruction error and sparsity.

The process of sparse coding that minimizes Equation 1, given a training set, is broken into two parts: (1) encoding an input by finding a set of activations $\boldsymbol{a}$ for input $\boldsymbol{I}$, and (2) learning a set of basis vectors (i.e., a dictionary) $\boldsymbol{\Phi}$ for the dataset. Encoding input $\boldsymbol{I}$ involves inferring activations $\boldsymbol{a}$ by minimizing Equation 1 with respect to $\boldsymbol{a}$ while holding $\boldsymbol{\Phi}$ fixed, for which we use the biologically informed Locally Competitive Algorithm (LCA) [19]. The final activations $\boldsymbol{a}$ are taken to be the output activations of the corresponding input $\boldsymbol{I}$. A sparse-coding layer can replace a convolutional layer in a DCNN by using activations $\boldsymbol{a}$ as the output of the layer. Learning a dictionary for sparse coding, analogous to learning filter weights in a DCNN, involves minimizing the cost function with respect to $\boldsymbol{\Phi}$ while holding $\boldsymbol{a}$ fixed via backpropagation of Equation 1. In the domain of images, dictionaries learned from sparse coding tend to represent oriented edges [15]. In our method, the input is first encoded using LCA, which is followed by one gradient descent step of minimizing the cost function with respect to $\boldsymbol{\Phi}$ while holding $\boldsymbol{a}$ fixed. Basis vectors are normalized to have unit L_2 norm after each update. Updating $\boldsymbol{\Phi}$ is repeated for multiple input batches until convergence.

3 Experiments

We compare fully supervised networks with networks incorporating an unsupervised sparse-coding layer by testing performance on a vehicle detection task using stereo video. We test the effect of training set size by varying the number of labeled training examples available to the network. We used the KITTI object detection dataset [6] for experiments. The dataset contains approximately 7000 examples, which we split into 6000 for training and 1000 for testing. Each example consists of three stereo frames ordered in time, with bounding box annotations for various objects in the left camera's last frame as ground truth. We normalize the stereo-video inputs to have zero mean and unit standard deviation and we downsampled them to be 256×64 pixels. We concatenated stereo inputs such that the input contains six features, i.e., RGB inputs from both left and right cameras. We kept time in a separate dimension for three-dimensional convolutions (for convolutional layers) or transposed convolution (for sparse-coding layers) across the time, height, and width axes of the input.

We generated the ground truth for our task by sliding a 32×16-pixel non-overlapping window across the left camera's last frame. We considered a window to be a positive instance if the window overlaps with any part of a car, van, or truck bounding box provided by the ground truth. The final output of each network is the probability of a window containing a vehicle, for all windows in the frame. We use the cross entropy between the ground truth and estimated probabilities as the supervised cost function to train all supervised layers within the network.

We tested various encoding schemes along with various weight initializations for the first layer of n-layer networks, as follows:

- **ConvSup**: Convolutional encoding. Weights are initialized randomly and learned via supervised training for car detection.
- **SparseUnsup**: Unsupervised sparse encoding to learn activations and weights.
- **ConvRand**: Convolutional encoding. Weights are initialized randomly and are not updated. This gives a random-weight baseline for the networks.

[1]The L_1 norm is used as a surrogate to the L_0 norm (i.e., the number of nonzero elements), as Equation 1 is non-convex with respect to $\boldsymbol{a}$ if the L_1 norm is replaced with an L_0 norm.

- **ConvUnsup**: Convolutional encoding. Weights are initialized from weights learned via unsupervised sparse coding and are not updated. This control tests the effect of weights versus encoding scheme on performance.

- **ConvFinetune**: Convolutional encoding. Weights are initialized from weights learned via unsupervised sparse coding. The weights were additionally trained via supervised training for car detection. This control is similar to ConvUnsup but tests the effect of additional training on the first-layer weights.

Once the first layer is set to one of the five possible options, the remaining $n - 1$ layers contain convolutional layers learned via backpropagation of the supervised loss. Each model was repeated six times with different random initial conditions and random presentations of the training data to get a range in performance for each network.

All models, experiments, and hyperparameters are available online[2].

4 Results

Table 1 shows the area under precision-versus-recall curve (AUC) of all models trained on all available training data, each tested with two, three, and four layers. Each network was trained on the car-detection task six times, with random weight initialization in higher layers and random ordering of training examples, in order to obtain the range of AUC values ("scores"). The range columns in Table 1 gives the difference between the maximum and minimum scores over the six runs. Here, we find that **SparseUnsup** performs worse than **ConvSup** and **ConvFinetune** with two layers, which agrees with the findings of Coates et al. [2]. However, **SparseUnsup** outperforms **ConvSup** in networks with three or more layers. This difference in performance due to the number of layers is likely because of the lack in capacity of the two layer model to map input to detection.

One key finding is that **SparseUnsup** is much more consistent (i.e., much less susceptible to random initial conditions and ordering of training examples) compared to all other models, as shown by the low range of AUC scores. Interestingly, **SparseUnsup** has lower range in performance than **ConvUnsup**, where both models use unsupervised weights learned via sparse coding and only differ in encoding scheme. This suggests that inferring activations in sparse coding is likely the reason for the additional consistency in performance.

Figure 1 gives the performance of **SparseUnsup** and **ConvSup** while varying the number of labeled training examples that each network was trained on. Here, the unsupervised weights were learned from all available data, without using training labels. We find that **SparseUnsup** achieves better performance than **ConvSup** across all numbers of labeled training examples tested for three and four layer networks. Additionally, **SparseUnsup** achieves better performance with two layer networks when the number of training examples is limited to only 100 training examples. Finally, **SparseUnsup** is much more consistent (as shown by the low range in AUC scores) in performance than that of **ConvSup**, for all models tested.

We compare activation maps for the first layer of **SparseUnsup**, **ConvUnsup**, and **ConvSup** in Figure 2. We find that the sparse-coding activations are selective to certain depths. For example, in Figure 2 for **SparseUnsup**, the top row shows a fast moving edge detector with a large binocular shift that corresponds to image features close to the camera, whereas the bottom row shows a static edge detector with no binocular shift that corresponds to image features far from the camera. In contrast, no convolutional layers show depth selectivity.

To control for the difference in number of nonzero activations, we applied a threshold to convolutional activations to match the number of nonzero activations in sparse coding, as shown in Figure 2 **ConvSup Sparse Control**. Sparse controls for **ConvUnsup** produced no nonzero activations on this input. Here, we show that sparsity-controlled activations from **ConvSup** do not show depth selectivity, as activations are active on image features at different depths. In all, these results may explain the differences in performance between convolutional layers and sparse-coding layers.

[2] `https://github.com/slundqui/TFSparseCode/`

Model	2 layers	Range	3 layers	Range	4 layers	Range
ConvSup	**0.672**	0.045	0.672	0.021	0.681	0.086
SparseUnsup	0.619	**0.004**	**0.681**	**0.009**	**0.693**	**0.014**
ConvRand	0.467	0.009	0.574	0.052	0.592	0.033
ConvUnsup	0.561	0.044	0.609	0.028	0.609	0.033
ConvFinetune	0.660	0.020	0.641	0.081	0.691	0.117

Table 1: Area under precision-versus-recall curve (AUC) for all models tested with varying depths. Models were ran six times with different initial conditions. Each value represents the median AUC score, and the range represents the difference between the highest score and the lowest score. **Chance** scores at .221 AUC.

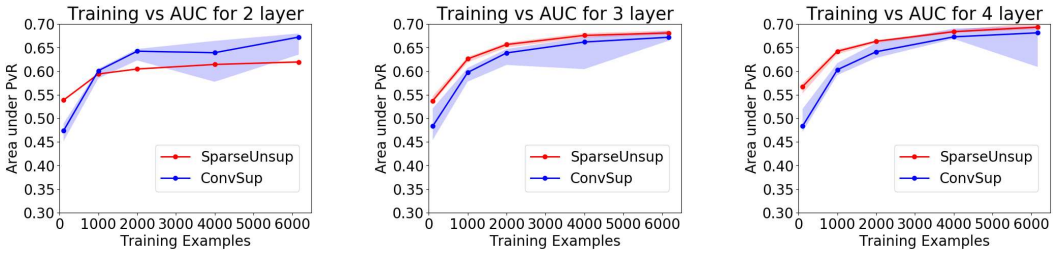


Figure 1: Number of training examples available versus AUC score for **SparseUnsup** (red) and **ConvSup** (blue) for two (left), three (middle), and four (right) layer networks. Each point is the median score over six independent runs, with the area between the maximum and minimum score filled in. Best viewed in color.

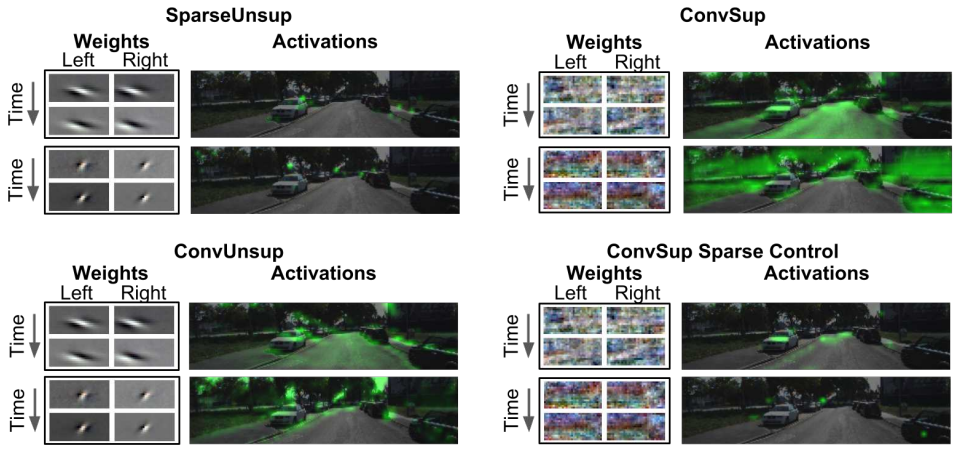


Figure 2: Nonzero activations of example weights overlaid on the input image. Magnitude of pixel values in green correspond to magnitude of activations. **SparseUnsup**: Activations for near tuned (top) and far tuned (bottom) weights for the sparse-coding layer. **ConvSup Sparse Control**: Activations from **ConvSup** with a threshold applied such that the number of activations matched that of sparse coding across the dataset. Best viewed in color.

5 Conclusion

We have shown that a neural network that incorporates unsupervised learning is able to outperform a fully supervised network when there exists limited labeled training data. Additionally, we show that performance of fully supervised networks can vary substantially when compared to networks with a sparse-coding layer. Finally, we compare activations and show that depth selective activations emerge from applying sparse coding to stereo-video data. In all, these results show that unsupervised sparse coding can be useful in domains where there exists a limited amount of available labeled training data.

Acknowledgments

This research is funded by the DARPA Cooperative Agreement Award HR0011-13-2-0015. We would like to thank Dylan M. Paiton for valuable conversations pertaining to this manuscript.

References

[1] Horace B Barlow. Unsupervised learning. *Neural Computation*, 1(3):295–311, 1989.

[2] Adam Coates, Andrew Y Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 215–223, 2011.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*, pages 248–255. IEEE, 2009.

[4] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[5] Steven H Ferris. Motion parallax and absolute distance. *Journal of experimental psychology*, 95(2):258, 1972.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Patrik O Hoyer and Aapo Hyvärinen. Independent component analysis applied to feature extraction from colour and stereo images. *Network: computation in neural systems*, 11(3):191–210, 2000.

[9] Phil Knag, Jung Kuk Kim, Thomas Chen, and Zhengya Zhang. A sparse coding neural network asic with on-chip learning for feature extraction and encoding. *IEEE Journal of Solid-State Circuits*, 50(4):1070–1079, 2015.

[10] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[11] Sheng Y Lundquist, Dylan M Paiton, Peter F Schultz, and Garrett T Kenyon. Sparse encoding of binocular images for depth inference. In *IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)*, pages 121–124. IEEE, 2016.

[12] Roland Memisevic and Christian Conrad. Stereopsis via deep learning. In *Advances in Neural Information Processing Systems (NIPS) Workshop on Deep Learning*, volume 1, page 2, 2011.

[13] N.T.T. Nguyen and G.T. Kenyon. Solving sparse representations for object classification using quantum d-wave 2x machine. In *International workshop on Post-Moore's Era Supercomputing (MPES)*, 2016.

[14] Bruno A Olshausen. Sparse coding of time-varying natural images. In *International Conference on Independent Component Analysis and Blind Source Separation*, pages 603–608. Citeseer, 2000.

[15] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[16] Ning Qian. Binocular disparity and the perception of depth. *Neuron*, 18(3):359–368, 1997.

[17] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *International conference on Machine learning (ICML)*, pages 759–766. ACM, 2007.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[19] Christopher Rozell, Don Johnson, Richard Baraniuk, and Bruno Olshausen. Locally competitive algorithms for sparse approximation. In *IEEE International Conference on Image Processing (ICIP)*, volume 4, pages IV–169. IEEE, 2007.

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[21] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2535. IEEE, 2010.

[22] Mengchen Zhu and Christopher J Rozell. Visual nonclassical receptive field effects emerge from sparse coding in a dynamical system. *PLoS computational biology*, 9(8):e1003191, 2013.

Meta-Learning for Semi-Supervised Few-Shot Classification

Mengye Ren **Eleni Triantafillou*** **Sachin Ravi*** **Jake Snell** **Kevin Swersky**

Joshua B. Tenenbaum **Hugo Larochelle** **Richard S. Zemel**

Abstract

In this work, we advance the few-shot classification paradigm towards a scenario where unlabeled examples are also available within each episode. We consider two situations: one where all unlabeled examples are assumed to belong to the same set of labeled classes of the episode, as well as the more challenging situation where examples from other distractor classes are also provided. To address this paradigm, we propose novel extensions of Prototypical Networks that are augmented with the ability to use unlabeled examples when producing prototypes. These models are trained in an end-to-end way on episodes, to learn to leverage the unlabeled examples successfully. We also propose a new split of ImageNet, consisting of a large set of classes, with a hierarchical structure. Our experiments confirm that our Prototypical Networks can learn to improve their predictions due to unlabeled examples, much like a semi-supervised algorithm would.

1 Introduction

The availability of large quantities of labeled data has enabled deep learning methods to achieve impressive breakthroughs in several tasks related to artificial intelligence, such as speech recognition, object recognition and machine translation. However, current deep learning approaches struggle in tackling problems for which labeled data are scarce.

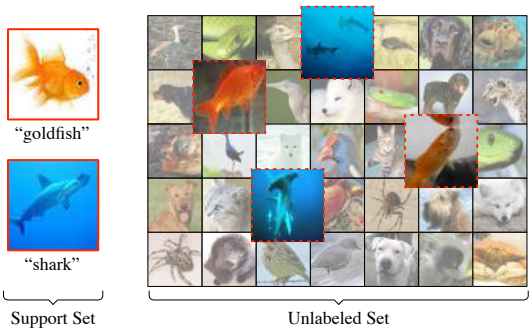


Figure 1: Consider a setup where the aim is to learn a classifier to distinguish between two previously unseen classes, goldfish and shark, given not only labeled examples of these two classes, but also a larger pool of unlabeled examples, some of which may belong to one of these two classes of interest.

For this reason, recently there has been an increasing body of work on few-shot learning, which considers the design of learning algorithms that specifically allow for better generalization on problems with small labeled training sets. Here we focus on the case of few-shot classification, where the given classification problem is assumed to contain only a handful of labeled examples per class. One approach to few-shot learning follows a form of meta-learning[2] [1, 2], which performs transfer learning from a pool of various classification problems generated from large quantities of available labeled data, to new classification problems from classes unseen at training time. Meta-learning may take the form

*These authors contributed equally.

[2]See the following blog post for an overview: `http://bair.berkeley.edu/blog/2017/07/18/learning-to-learn/`

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

of learning a shared metric [3, 4], a common initialization for few-shot classifiers [5, 6] or a generic inference network [7, 8].

However, this progress has been in a limited scenario, which differs in many dimensions from how humans learn new concepts. In this paper we aim to generalize the few-shot setting in two ways. First we consider a scenario in which the new classes are learned in the presence of additional unlabeled data. Second, we consider the situation where the new classes to be learned are not viewed in isolation. Instead, many of the unlabeled examples are from different classes; the presence of such *distractor* classes introduces an additional and more realistic level of difficulty to the few-shot problem. We propose and study three novel extensions of Prototypical Networks [4], a state-of-the-art approach to few-shot learning, to the semi-supervised setting. We demonstrate in our experiments that our semi-supervised variants successfully learn to leverage unlabeled examples and outperform purely supervised Prototypical Networks.

2 Semi-Supervised Few-Shot Learning

We denote our training set as a tuple of labeled and unlabeled examples: $(\mathcal{S}, \mathcal{R})$. The labeled portion is the usual support set $\mathcal{S}$ of the few-shot learning literature, containing a list of tuples of inputs and targets. In addition to classic few-shot learning, we introduce an unlabeled set $\mathcal{R}$ containing only inputs: $\mathcal{R} = \{\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2, \ldots, \tilde{\boldsymbol{x}}_M\}$. As in the purely supervised setting, our models are trained to perform well when predicting the labels for the examples in the episode's query set $\mathcal{Q}$. Figure 2 shows a visualization of training and test episodes.

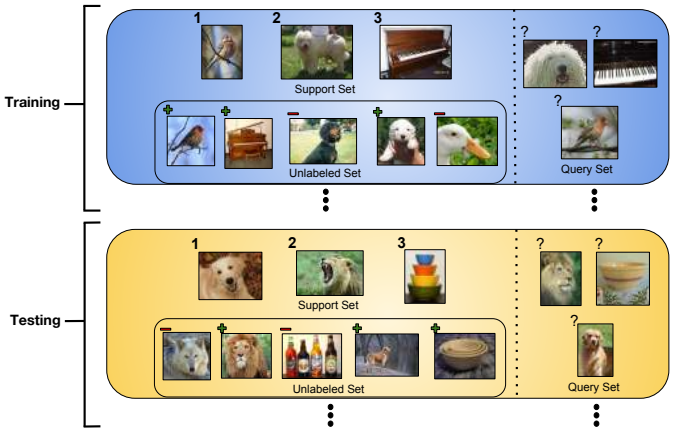


Figure 2: Example of the semi-supervised few-shot learning setup. Training episodes consist of a support set $\mathcal{S}$, an unlabeled set $\mathcal{R}$, and a query set $\mathcal{Q}$. The items in $\mathcal{R}$ may either be pertinent to the labeled classes (with + signs) or they may be *distractor* items (with - signs).

2.1 Prototypical Networks with Soft k-Means

We first consider a simple way of leveraging unlabeled examples for refining prototypes, by taking inspiration from semi-supervised clustering. One natural choice would be to borrow from the inference performed by soft k-means. We prefer this version of k-means over hard assignments since hard assignments would make the inference non-differentiable. We start from the regular Prototypical Network [4]'s prototypes $\boldsymbol{p}_c$. Then, the unlabeled examples get a partial assignment $(\tilde{z}_{j,c})$ to each cluster based on their Euclidean distance to the cluster locations. Finally, refined prototypes are obtained by incorporating these unlabeled examples.

This process can be summarized as follows:

$$\tilde{\boldsymbol{p}}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c} + \sum_j h(\tilde{\boldsymbol{x}}_j) \tilde{z}_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c}}, \quad \text{where} \quad \tilde{z}_{j,c} = \frac{\exp\left(-||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_c||_2^2\right)}{\sum_{c'} \exp\left(-||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_{c'}||_2^2\right)} \tag{1}$$

We could perform several iterations of refinement, as is usual in k-means. However, we have experimented with various number of iterations and found results to not improve beyond a single refinement step.

2.2 Prototypical Networks with Soft k-Means with a Distractor Cluster

The soft k-means approach described above implicitly assumes that each unlabeled example belongs to either one of the N classes in the episode. However, it would be much more general to have a model robust to the existence of examples from other classes, which we refer to as distractor

classes. A simple way to address this is to add an additional cluster whose purpose is to capture the distractors.

$$\boldsymbol{p}_c = \begin{cases} \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c}}{\sum_i z_{i,c}} & \text{for } c = 1...N \\ \mathbf{0} & \text{for } c = N+1 \end{cases} \tag{2}$$

Here we take the simplifying assumption that the distractor cluster has a prototype centered at the origin. We also consider introducing length-scales r_c to represent variations in the within-cluster distances, specifically for the distractor cluster:

$$\tilde{z}_{j,c} = \frac{\exp\left(-\frac{1}{r_c^2}||\tilde{\boldsymbol{x}}_j - \boldsymbol{p}_c||_2^2 - A(r_c)\right)}{\sum_{c'} \exp\left(-\frac{1}{r_c^2}||\tilde{\boldsymbol{x}}_j - \boldsymbol{p}_{c'}||_2^2 - A(r_{c'})\right)}, \text{ where } A(r) = \frac{1}{2}\log(2\pi) + \log(r) \tag{3}$$

For simplicity, we set $r_{1...N}$ to 1 in our experiments, and only learn the length-scale of the distractor cluster r_{N+1}.

2.3 Prototypical Networks with Soft k-Means and Masking

Modeling distractor unlabeled examples with a single cluster is likely too simplistic , since distractor examples may very well cover more than a single natural object category. To address this problem, we incorporate a soft-masking mechanism on the contribution of unlabeled examples. We start by computing normalized distances $\tilde{d}_{j,c}$ between examples $\tilde{\boldsymbol{x}}_j$ and prototypes $\boldsymbol{p}_c$:

$$\tilde{d}_{j,c} = \frac{d_{j,c}}{\frac{1}{M}\sum_j d_{j,c}}, \text{ where } d_{j,c} = ||h(\tilde{\boldsymbol{x}}_j) - \boldsymbol{p}_c||_2^2 \tag{4}$$

Then, soft thresholds β_c and slopes γ_c are predicted for each prototype, by feeding to a small neural network various statistics of the normalized distances for the prototype:

$$[\beta_c, \gamma_c] = \text{MLP}\left([\min_j(\tilde{d}_{j,c}), \max_j(\tilde{d}_{j,c}), \text{var}_j(\tilde{d}_{j,c}), \text{skew}_j(\tilde{d}_{j,c}), \text{kurt}_j(\tilde{d}_{j,c})]\right) \tag{5}$$

This allows each threshold to use information on the amount of intra-cluster variation to determine how aggressively it should cut out unlabeled examples.

Then, soft masks $m_{j,c}$ for the contribution of each example to each prototype are computed, by comparing to the threshold the normalized distances, as follows:

$$\tilde{\boldsymbol{p}}_c = \frac{\sum_i h(\boldsymbol{x}_i) z_{i,c} + \sum_j h(\tilde{\boldsymbol{x}}_j)\tilde{z}_{j,c} m_{j,c}}{\sum_i z_{i,c} + \sum_j \tilde{z}_{j,c} m_{j,c}}, \text{ where } m_{j,c} = \sigma\left(-\gamma_c\left(\tilde{d}_{j,c} - \beta_c\right)\right) \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function.

3 Experiments

3.1 Datasets

We evaluate the performance of our models on three datasets: two benchmark few-shot classification datasets and a novel large-scale dataset that we hope will be useful for future few-shot learning work.

Omniglot [9] is a dataset of 1,623 handwritten characters from 50 alphabets. Each character was drawn by 20 human subjects. We follow the few-shot setting proposed by [3], in which the images are resized to 28×28 pixels and rotations in multiples of 90° are applied, yielding 6,492 classes in total. These are split into 4,800 training classes and 1,692 classes for test. 10% of the training images are used as labeled examples.

***mini*ImageNet** [3] is a modified version of the ILSVRC-12 dataset [10], in which 600 images for each of 100 classes were randomly chosen to be part of the dataset. We rely on the class split used by [5]. These splits use 64 classes as training, 16 for validation, and 20 for test. All images are of size 84×84 pixels. 40% of the training images are used as labeled examples.

***tiered*ImageNet** is our proposed dataset for few-shot classification. Like *mini*Imagenet, it is a subset of ILSVRC-12. However, *tiered*ImageNet represents a larger subset of ILSVRC-12 (608

ProtoNet Model	Err.	Err. w/ D
Supervised	5.16%	5.16%
Semi-Supervised Inference	2.35%	4.70%
Soft k-Means	2.56%	4.59%
Soft k-Means+Cluster	**2.18%**	2.71%
Masked Soft k-Means	2.46%	**2.62%**

Table 1: Omniglot 1-shot Results

ProtoNet Model	*mini* / *tiered* 1-shot Acc.	*mini* / *tiered* 5-shot Acc.	*mini* / *tiered* 1-shot Acc. w/ D	*mini* / *tiered* 5-shot Acc. w/ D
Supervised	43.36% / 46.60%	59.03% / 67.18%	43.36% / 46.60%	59.03% / 67.18%
Semi-Supervised Inference	48.68% / 50.38%	62.94% / 70.26%	46.16% / 46.87%	62.32% / 68.38%
Soft k-Means	48.25% / 53.41%	**65.72% / 71.31%**	46.72% / 50.18%	61.94% / 68.83%
Soft k-Means+Cluster	**50.87% / 55.82%**	63.75% / 70.79%	48.60% / 49.87%	61.51% / 70.16%
Masked Soft k-Means	50.57% / 52.76%	63.78% / 70.08%	**50.04% / 50.93%**	**62.50% / 71.00%**

Table 2: *mini*ImageNet and *tiered*ImageNet 1/5-shot Results

classes rather than 100 for *mini*ImageNet). Analogous to Omniglot, in which characters are grouped into alphabets, *tiered*ImageNet groups classes into broader categories corresponding to higher-level nodes in the ImageNet [11] hierarchy. There are 34 categories in total, with each category containing between 10 and 30 classes. These are split into 20 training categories, 6 validation categories, and 8 testing categories. (details of the dataset can be found in the Supplementary Materials). This ensures that all of the training classes are sufficiently distinct from the testing classes, unlike *mini*ImageNet and other alternatives such as *rand*ImageNet proposed by [3]. For example, "pipe organ" is a training class and "electric guitar" is a test class in the [5] split of *mini*Imagenet, even though they are both musical instruments. 10% of the training images are used as labeled examples.

In each dataset we compare our three semi-supervised models with two baselines. The first baseline, referred to as Supervised in our tables, is an ordinary Prototypical Network that is trained in a purely supervised way on the labeled split of each dataset. The second baseline, referred to as Semi-Supervised Inference, uses the embedding function learned by this supervised Prototypical Network, but performs semi-supervised refinement of the prototypes at inference time using a step of Soft k-Means refinement. For *tiered*ImageNet, We report the final test accuracy trained with both training and validation set.

3.2 Results

Results for Omniglot are given in Table 1, and *mini*ImageNet and *tiered*ImageNet in Table 2. Across all three benchmarks, at least one of our proposed models outperform the baselines, demonstrating the effectiveness of our semi-supervised meta-learning procedure. In particular, Soft k-means+Cluster performs the best on 1-shot non-distractor settings, as the extra cluster seems to provide a form of regularization that pushes the clusters farther apart. Soft k-Means performs well on 5-shot non-distractor settings, as it considers the most unlabeled examples. Masked Soft k-Means shows the most robust performance in distractors settings, in both 1-shot and 5-shot tasks. For 5-shot, Masked soft k-Means reaches comparable performance compared to the upper bound of the best non-distractor performance.

From Figure 3, we observe clear improvement in test accuracy when the number grows from 0 to 25. Note that our models were trained with $M = 5$ and thus are showing an ability to extrapolate in generalization. This confirms that, through meta-training, the models learned to acquire a better representation that will be more helpful after semi-supervised refinement.

Note that the wins obtained in our semi-supervised learning are super-additive. Consider the case of the simple k-Means model on 1-shot without Distractors. Training only on labeled examples while incorporating the unlabeled set during test time produces an advantage of 3.8% (50.4-46.6), while incorporating the unlabeled set during training but not during test produces a win of 1.1% (47.7-46.6). Incorporating unlabeled examples during both training and test yields a win of 6.8% (53.4-46.6).

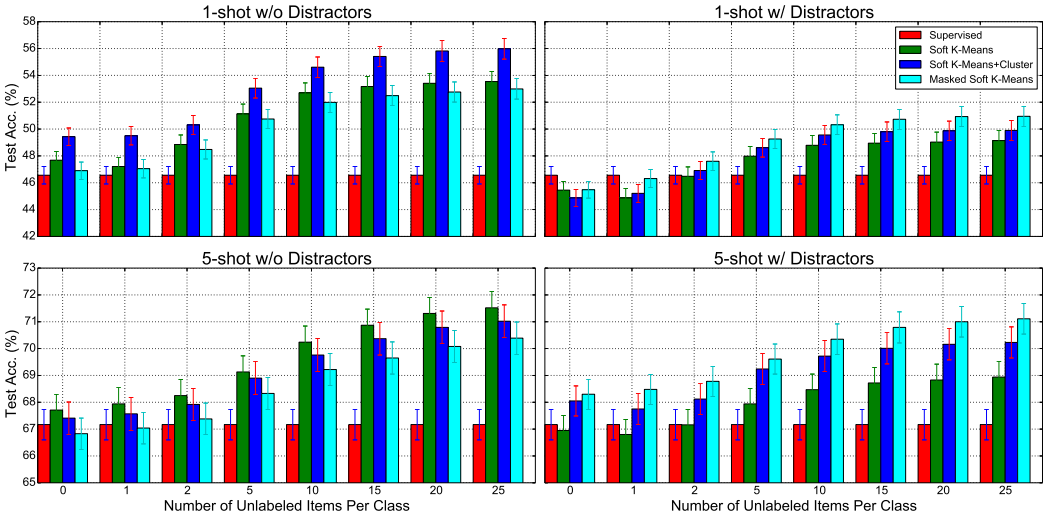


Figure 3: Model Performance on *tiered*ImageNet with different number of unlabeled items during test time.

4 Conclusion

In this work, we propose a novel semi-supervised few-shot learning paradigm, where an unlabeled set is added to each episode. We also extend the setup to more realistic situations where the unlabeled set has classes not belonging to the labeled classes. We also introduce a larger dataset split, *tiered*ImageNet, with hierarchical levels of labels. We propose several novel extensions of Prototypical Networks, and they show consistent improvements under semi-supervised settings compared to our baselines. As future work, we are working on incorporating fast weights [12, 6] into our framework so that examples can have different embedding representation given the contents in the episode.

References

[1] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[2] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[3] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638, 2016.

[4] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30*, 2017.

[5] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations*, 2017.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *34th International Conference on Machine Learning*, 2017.

[7] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. One-shot learning with memory-augmented neural networks. In *33rd International Conference on Machine Learning*, 2016.

[8] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *CoRR*, abs/1707.03141, 2017.

[9] Brenden M. Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua B. Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the 33th Annual Meeting of the Cognitive Science Society, CogSci 2011, Boston, Massachusetts, USA, July 20-23, 2011*, 2011.

[10] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[12] Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4331–4339, 2016.

[13] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A *tiered*Imagenet Dataset Details

Each high-level category in *tiered*ImageNet contains between 10 and 30 ILSVRC-12 classes (17.8 on average). In the ImageNet hierarchy, some classes have multiple parent nodes. Therefore, classes belonging to more than one category were removed from the dataset to ensure separation between training and test categories. Test categories were chosen to reflect various levels of separation between training and test classes. Some test categories (such as "working dog") are fairly similar to training categories, whereas others (such as "geological formation") are quite different. The list of categories is shown below and statistics of the dataset can be found in Table 3. A visualization of the categories according to the ImageNet hierarchy is shown in Figure 4. The full list of classes per category will also be made public, however for the sake of brevity we do not include it here.

Table 3: Statistics of the *tiered*ImageNet dataset.

	Train	Val	Test	Total
Categories	20	6	8	34
Classes	351	97	160	608
Images	448,695	124,261	206,209	779,165

Train Categories: n02087551 (hound, hound dog), n02092468 (terrier), n02120997 (feline, felid), n02370806 (ungulate, hoofed mammal), n02469914 (primate), n01726692 (snake, serpent, ophidian), n01674216 (saurian), n01524359 (passerine, passeriform bird), n01844917 (aquatic bird), n04081844 (restraint, constraint), n03574816 (instrument), n03800933 (musical instrument, instrument), n03125870 (craft), n04451818 (tool), n03414162 (game equipment), n03278248 (electronic equipment), n03419014 (garment), n03297735 (establishment), n02913152 (building, edifice), n04014297 (protective covering, protective cover, protection).

Validation Categories: n02098550 (sporting dog, gun dog), n03257877 (durables, durable goods, consumer durables), n03405265 (furnishing), n03699975 (machine), n03738472 (mechanism), n03791235 (motor vehicle, automotive vehicle),

Test Categories: n02103406 (working dog), n01473806 (aquatic vertebrate), n02159955 (insect), n04531098 (vessel), n03839993 (obstruction, obstructor, obstructer, impediment, impedimenta), n09287968 (geological formation, formation), n00020090 (substance), n15046900 (solid).

B Extra Experimental Results

Figure 5 shows test accuracy values with different number of unlabeled items during test time. We observe clear improvement in test accuracy when the number grows from 0 to 25. Note that our models were trained with $M = 5$ and thus are showing an ability to extrapolate in generalization.

This confirms that, through meta-training, the models learned to acquire a better representation that will be more helpful after semi-supervised refinement. Figure 6 shows our mask output value distribution of the masked soft k-means model on Omniglot. The mask values have a bi-modal distribution, corresponding to distractor and non-distractor items.

C Hyperparameter Details

For Omniglot, we adopted the best hyperparameter settings found for ordinary Prototypical Networks in [4]. In these settings, the learning rate was set to 1e-3, and cut in half every 2K updates starting at update 2K. We trained for a total of 20K updates. For *mini*Imagenet and *tiered*ImageNet, we trained with a starting learning rate of 1e-3, which we also decayed. We started the decay after 25K updates, and every 25K updates thereafter we cut it in half. We trained for a total of 200K updates. We used ADAM [13] for the optimization of our models. For the MLP used in the Masked Soft k-Means model, we use a single hidden layer with 20 hidden units with a tanh non-linearity for all 3 datasets. We did not tune the hyparameters of this MLP so better performance may be attained with a more rigorous hyperparameter search.

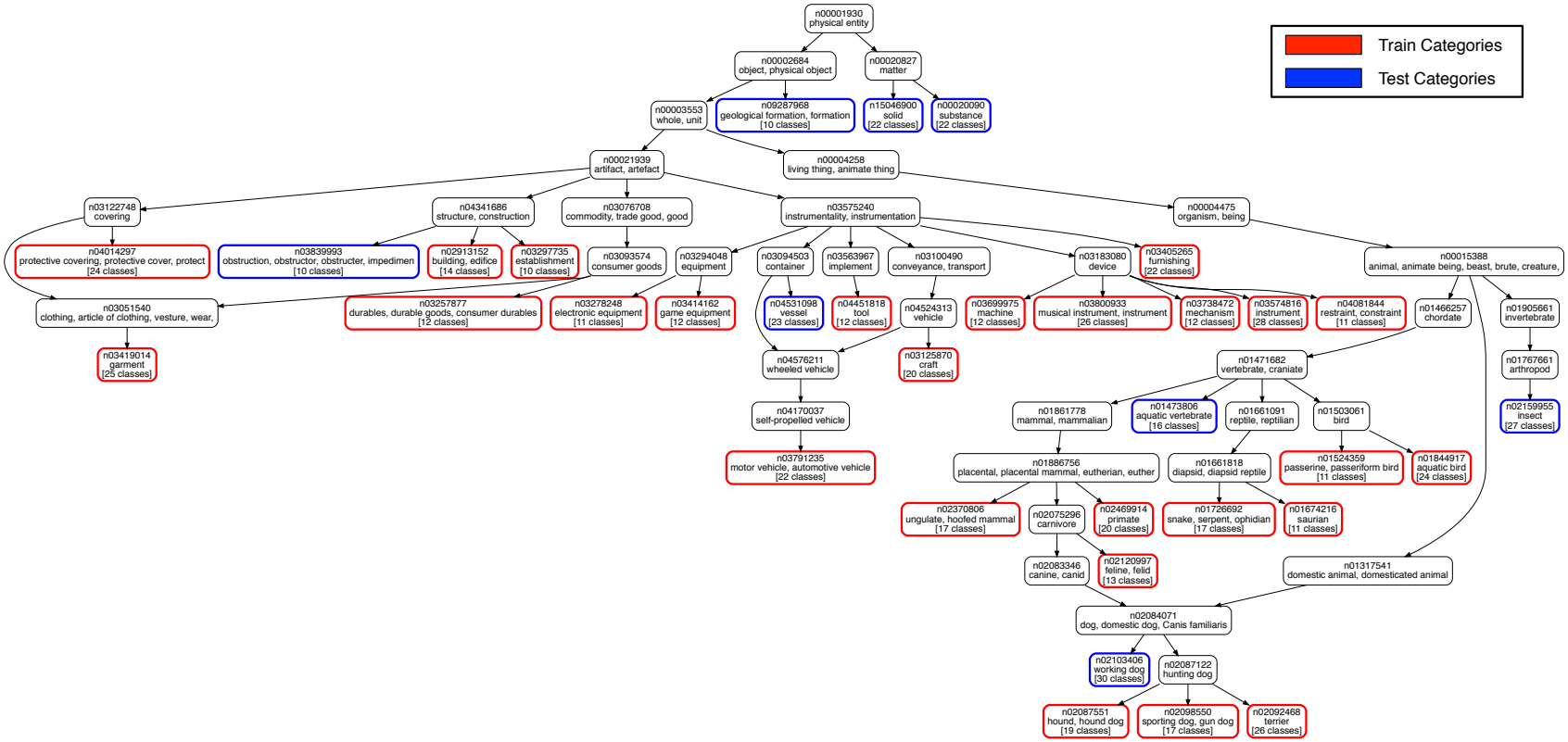


Figure 4: Hierarchy of *tiered*Imagenet categories. Training categories are highlighted in red and test categories in blue. Each category indicates the number of associated classes from ILSVRC-12. Best viewed zoomed-in on electronic version.

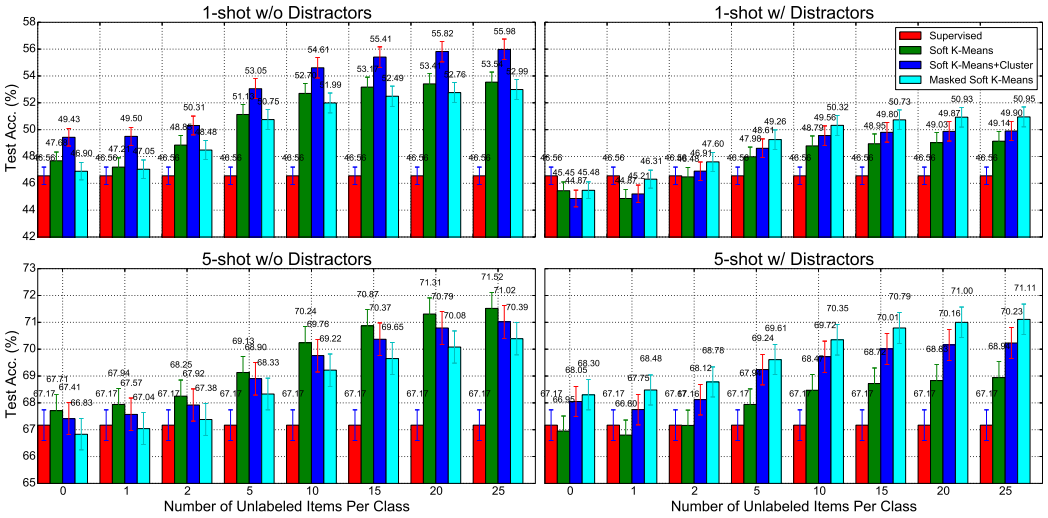


Figure 5: Model Performance on *tiered*ImageNet with different number of unlabeled items during test time. We include test accuracy numbers in this chart.

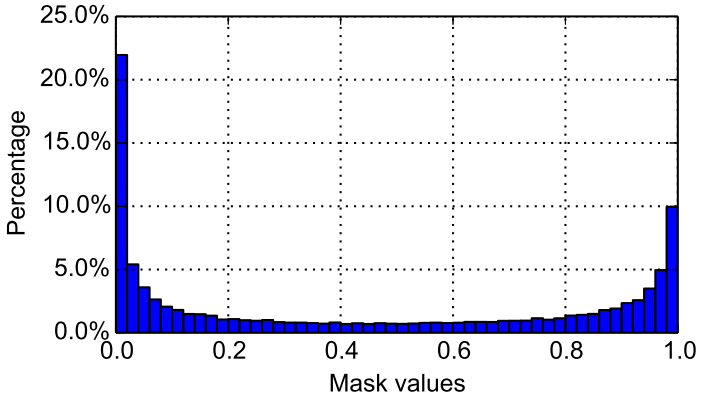


Figure 6: Mask values predicted by masked soft k-means on Omniglot.

Video to Skill Tagging using Transcripts under Weak Supervision

Zhe Cui
Department of Electrical and Computer Engineering
University of Maryland, College Park
College Park, MD 20742
zcui@umd.edu

Shivani Rao
LinkedIn, Inc.
Sunnyvale, CA 94085
sirao@linkedin.com

Abstract

Delivering micro content (small pieces of learning materials) from a course catalog has become important as it helps learners engage quickly with a learning platform on a need-to-know period. In order to recommend videos that can teach skills relevant to a learner, we propose an approach for tagging videos with skills using transcripts. Since, there is no labeled data to train a video-to-skill model directly, we leverage a bunch of techniques that fall under the umbrella of weak supervision. We evaluate our approach using offline metrics like Precision, Recall, F-1 score and online metrics via large-scale A/B testing.

1 Introduction

Online Learning has taken off with a plethora of e-learning products such as Coursera, LinkedIn Learning, and Udacity, to name a few. More recently, learners are engaging with online learning platforms in an informal way, i.e., they are seeking relevant videos on a need-to-know period as opposed to taking on the task of sitting through an entire course. In order to deliver relevant videos to the learner (whose learning interests are expressed in terms of skills), we need to map videos to skills, so that we can match learners and videos in the skill space. The LinkedIn Learning platform [2] that we focus on in this paper hosts thousands of courses and each course comprises of several videos, resulting in hundreds of thousands of videos, and it is impossible and expensive to have Subject Matter Experts (SMEs) hand-label all the videos. Even at the course-level, the SME provided skill-tags are sparse and incomplete and cover only 2% of the skills on the platform.

In this paper, we propose an approach that utilizes raw transcripts to tag videos with skills for the purpose of recommending micro contents to learners on LinkedIn Learning platform. One of the byproducts of our proposed algorithm is that, it also improves the quality of course-to-skill mapping as well, i.e., the skill coverage grows by 3X for the mapping. We present our experimental findings using offline (e.g., Precision/Recall) and online metrics (e.g., A/B testing).

2 Related Work

In industry settings, getting enough labeled training data has always been a bottle neck, but academic research focus on this problem has been more recent [19]. This form of learning through videos is called micro learning, and the videos are called micro content in educational research community [15, 16]. Some of the techniques that have been developed to circumvent this problem include active learning [18, 7], semi-supervised learning [5, 17, 10], and transfer learning [14, 4] [1]. In the field of deep learning, unsupervised pre-training [8] has been used as proven technique to improve

[1] Due to page limit, please refer to the cited paper for a full survey of related work.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

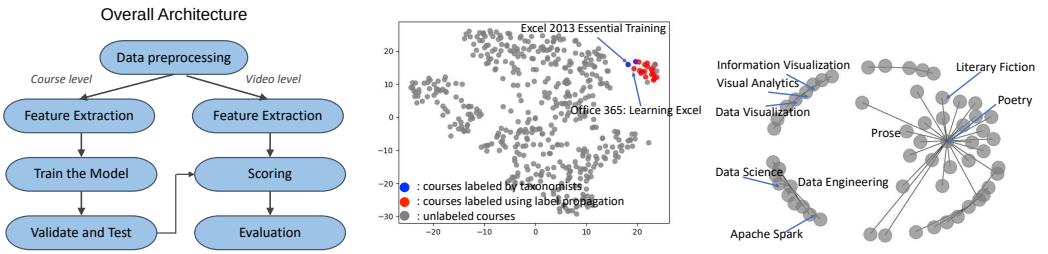


Figure 1: Left: proposed model architecture. Middle: Visualization of course representations (with PCA and t-SNE [13]). We propagate skill labels to courses that are close neighbors of a labeled course in this embedded space. The skill label is propagated from two courses related to excel (blue) to other courses (e.g., Office 365.). Right: subset of skill graph. There are three clusters shown: Data Engineering, Poetry, and Commerical Design. Each of them are closely related to similar skills e.g., Data Science to Data Engineering, Literary Fiction to Poetry.

performance of classification tasks especially in the context of transfer learning and semi-supervised learning. Our work is inspired by all of these contributions, and stands to show how they can be applied to solve a data problem in the industrial setting.

3 Modeling

The problem we are trying to solve can be described as follows, "Given video transcripts, infer skills on the learning platform". Since the set of skills is derived from a standardized dictionary [3], our approach of extracting skills is to build a separate text classification model for each skill. Thus for each skill, the feature matrix remains the same across models, and the labeled column changes. While extracting tags or assigning skills to textual information is not new, there are some unique challenges for skill tagging of video transcripts in the context of learning from limited labeled data.

- *Videos do not have skill labels.* The task of labeling videos with skills is an unsupervised learning task, given no labeled training data available. Thus we create a model that can be trained on course level, by combining all the transcripts of the videos in that course into one document (Fig. 1 Left). This turns it into a transfer learning problem [14]. Since the model is trained entirely on textual features, we can use the same model to predict skills for video transcripts.
- *For each course, the number of skills tagged by SMEs is sparse (2-3 per course).* In order to alleviate the issue, we use a heuristic version of the label propagation technique to expand the labels [20].
- *Existing course level skill labels only cover 2% of the total skills.* The purpose of our proposed approach is not just learn a robust video to skill mapping, but also discover *new* skills that are relevant to a course or a video, that was probably missed about by SMEs. In order to discover new skills, we expand labeled skills using an external source that we call the skill graph [2]. based on correlated skill graph to generate more labeled skills for training.
- *Text features are very sparse.* We use unsupervised pre-training on the courses, by learning a doc2vec [11] representation of transcripts to create document embeddings.

The overall video to skill model flow is shown in Fig. 1 Left. The main idea is gained from transfer learning: **train at the course-level, and make predictions at video level**. Specifically, each course level transcript is obtained by concatenating video level transcripts and represented using latent representations (doc2vec [12], details below). We present key procedures of the proposed model below.

3.1 Label Augmentation

As mentioned before, the text classification problem is faced with a challenge of limited labeled data. We use two approaches to increase the labeled data for each of the per-skill models.

- **Course based expansion:** The expansion of skill labels for courses is based on course similarities. An example is shown in Fig. 1 Middle. Using the document embeddings of courses, each course pair can be measured by a similarity metric (here we use kNN kernel as the metric). The skill label "Excel" has been expanded to other courses that are close in the graph. This step can be considered as a heuristic version of label propagation [20].
- **Skill based expansion:** We leverage an external source of skill graph [1] to increase the labeled training data. A subgraph is shown in Fig. 1 Right. The weight of the edges represents similarity [6]. In order to ensure the quality of the augmented labels, for each skill, we retain only the skills that have similarity score in the 99th percentile (top 1%). This step can be considered as mixture of noisy labeled data with high precision labeled data from the SMEs.

3.2 Model Training

With all the labeled data (including those obtained with label expansion), we build a "per-skill" model: training the model separately for each associated skill. In spite of applying all the label generation schemes discussed in the last section, the percentage of labeled data is still small. Since the model needs to score and rank skills, it is preferred to get predictions with quantitative measures to rank them. Hence, we chose logistic regression [9] which gives a score for each skill prediction between 0 and 1.

3.3 Scoring Pipeline

Once we have trained per-skill models, we can score any video or course to predict which skills are relevant. In what follows, we present two key considerations while designing our scoring pipeline:

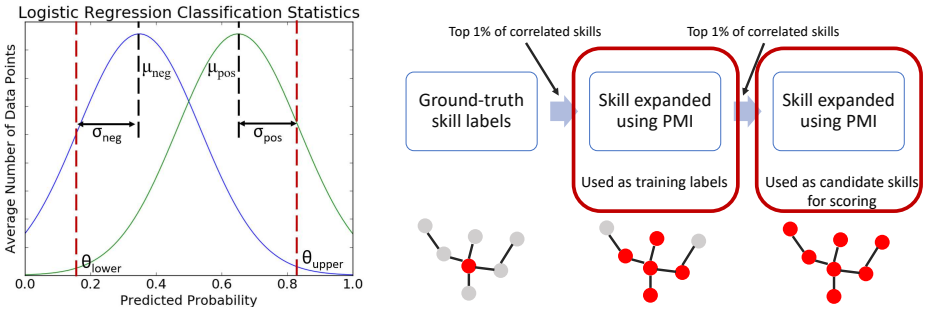


Figure 2: Left: one sample model with predicted distribution of positive and negative labels. Truncate at θ_{upper} to retain only confident skills. Right: skill expansion based on skill-graph, for the purpose of generating candidates. During training, we only did 1-hop expansion (See Figure 1 Right) to generate labels. For the purpose of scoring, we use 2-hops to generate candidates for scoring.

- *Preserve confident predictions.* The goal is to retain only skill predictions that the model is confident in. Typically, logistic regression uses 0.5 as default threshold, but for our model, we set the threshold to θ_{upper} and θ_{lower}, which can be set heuristically as $\theta_{upper} = \mu_{pos} + 2 * \sigma_{pos}$ and $\theta_{lower} = \mu_{neg} - 2 * \sigma_{neg}$ respectively.(Fig. 2 Left[2]). With $probability$ outside the range $(\theta_{lower}, \theta_{upper})$, the predictions are considered to be confident.
- *Candidate Pool Generation.* There are thousands of different skills, which means thousands of models trained for predictions. If we were to predict every course and video against all the skills, it would be rather inefficient (e.g: there is no point in scoring a piano course transcript for the skill Java) and we would also run into having lots of false positive predictions. Hence, for each of the courses/videos we only evaluate skills that are possibly relevant or related to a given skill that is already tagged by SMEs. We utilize the skill graph for this, but this time, we use a 2-hops to retrieve our candidates. Fig. 2 Right gives an illustration of such a two-step procedure using the skill graph.

[2]To keep confidentiality, we remove the actual numbers on Y-axis of the left figure.

4 Evaluation

While the primary purpose of the proposed model is to map videos to skills, which helps video recommendations, to test the efficiency of the approach, we evaluate both course to skill and video to skill mappings. For video to skill mapping, we are collecting feedback from our SMEs as we speak (For some examples of video-transcript to skill mapping, please see Appendix). However, we do have mechanisms to do large-scale online A/B testing with our course-to-skill models as we will describe in this section.

4.1 Datasets and Metrics

We apply our proposed model to over $5k$ courses and over $100k$ videos. Typically, one course is labeled manually with 1 or 2 skills by SMEs. With skill expansion using the skill graph, the number of unique skills is tripled. The metrics we use include: Precision@K, Recall@K, F-1 score@K, as well as Click-Through-Rate (CTR), which is online metric that measures the number of engaged learners.

4.2 Research Questions

We present our evaluations according to the following research questions:

How does changing the threshold θ_{upper} in logistic regression model affect the offline metrics?

Fig. 3 Left shows a skill model of Precision, Recall and F-1 score changes across different probability thresholds. The Recall stays high most of the time, while Precision keeps increasing as threshold goes higher. From a practical perspective, a course cannot teach too many skills, so it is reasonable to err on the side of caution and focus on Precision, not Recall.

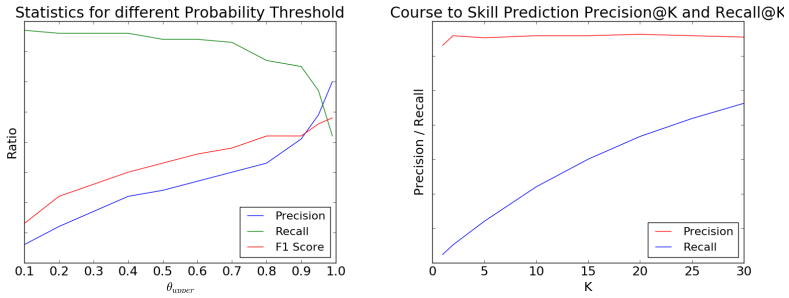


Figure 3: Left: Precision and Recall with different probability thresholds on predictions from one course to skill model. Right: Precision@K and Recall@K for top K retained skills of the proposed approach.

How does the performance change w.r.t. K, the number of top skills (K) we retain?

Fig. 3 Right gives an overview of how Precision and Recall changes w.r.t. the number of top predicted skills we retain. It's worth noting that while Recall is an important metric in most classification and retrieval scenarios, it can be artificially low for course to skill mapping where SME provided labels are sparse and incomplete. This is due to the fact that SMEs are unable to retrieve all the relevant skills for a given course. Typically each course has 2 or 3 labeled skills from SME, and the number of predicted skills for each course may increase up to 40 or more. For example, "Python" course may get assigned more than 80 skills since it is a fundamental course for several types of skills such as "Web development", "back-end development", "Data Science", "Machine Learning"..

Is it possible that the same skill tags will be assigned to all the videos in a course?

Since we train the model based on course level skill labels, it's possible that all the videos within the same course get the same labels. Thus measuring the skill variance for videos within a course gives us valuable information about how the model is able to differentiate between video level features. Specifically, skills that are predicted for majority of the videos in a course can be bubbled up to a course level skill mapping, and the skills that assigned to only some videos will be kept as actual video labels.

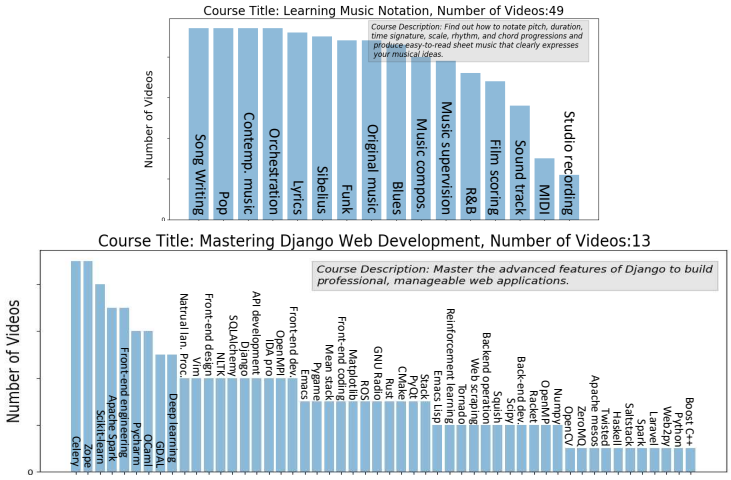


Figure 4: The number of predicted skills appears in each video within the same course. Course on the top has a small variance while variance of predicted skills of the bottom one is larger.

Fig. 4 shows video to skill variances for two typical courses. It shows the number of videos tagged with each skill. The “Learning Music Notation" course has very low skill-variance, whereas the “Django Web Development" course has a higher variance across videos.

How does the predicted course-to-skill mapping compare to the existing course-to-skill mapping in Online A/B testing?

In order to measure the performance of our predictions, we have run a large-scale online A/B testing experiment on one of our engagement channels – the jobs page. Job seekers arrive on this page to view jobs and possibly apply for jobs. For existing learners, this page serves as another channel where they can be directed to the learning platform by showing them course recommendations relevant to the job they are viewing. The jobs themselves map to skills, and with courses mapping to skills as well, jobs can be matched with courses in the skills space. By keeping the job-to-skill model the same, we created two job-to-course models using two course-to-skill mappings: (a) j001 - job-to-skill (b001) matched with course-to-skill model (where skills are tagged by SMEs) and (b) j002 - job-to-skill (b001) matched with course-to-skill model (where skills are tagged by our approach). We doubled the job-to-course coverage from j002 to j001 (meaning 2x more jobs had course recommendations due to the higher skill coverage of the proposed course-to-skill model). On the jobs page, we saw 65% increase in impressions, while keeping the Click Through Rate (CTR) unchanged (which means, that we increased coverage (read-Recall) without hurting Precision). Last but not the least, with j002, there is 4.6% increase in learner engagement (unique learners that watched course videos) on the learning platform page.

5 Conclusion

In this paper we propose an approach to extract skills from video transcripts with no labeled data. We drew inspiration from transfer learning, in that, we trained a course-to-skill model by using textual features only and apply this model to predict skills for the videos. While we started with the goal of extracting skills from video transcripts, our approach also improves course to skill mapping. While we are still collecting feedback on the quality of our video-to-skill recommendations from SMEs, we used offline metrics and online A/B testing experiments to gauge the quality of course-to-skill predictions made by our model. For future work, we plan to investigate with more sophisticated approaches to learning document representations from text. We also plan to use different sources of data for our skill-to-skill mapping. We plan to use the prediction score to intelligently derive a course-skill and video-skill pairs for which we need to collect feedback from SMEs (tending towards an active learning approach). While, we acknowledge that this paper itself does not propose many novel ideas, we argue that the work done here shows the power of weak supervision approaches by applying them to an industry problem, where labeled data is indeed hard to come by.

References

[1] 2017. LinkedIn Economic Graph Research. (2017). https://engineering.linkedin.com/data/economic-graph-research.

[2] 2017. LinkedIn Learning. (2017). https://www.linkedin.com/learning/.

[3] 2017. LinkedIn Skills List. (2017). https://www.linkedin.com/directory/topics-a/.

[4] Isabelle Augenstein, Andreas Vlachos, and Diana Maynard. 2015. Extracting relations between non-standard entities using distant supervision and imitation learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 747–757.

[5] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks* 20, 3 (2009), 542–542.

[6] Thomas M Cover and Joy A Thomas. 2012. *Elements of information theory*. John Wiley & Sons.

[7] Gregory Druck, Burr Settles, and Andrew McCallum. 2009. Active learning by labeling features. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 81–90.

[8] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11, Feb (2010), 625–660.

[9] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. 2013. *Applied logistic regression*. Vol. 398. John Wiley & Sons.

[10] Samuli Laine and Timo Aila. 2016. Temporal Ensembling for Semi-Supervised Learning. *arXiv preprint arXiv:1610.02242* (2016).

[11] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1188–1196.

[12] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1188–1196.

[13] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[14] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

[15] Cristobal Romero and Sebastian Ventura. 2007. Educational data mining: A survey from 1995 to 2005. *Expert systems with applications* 33, 1 (2007), 135–146.

[16] Cristóbal Romero and Sebastián Ventura. 2010. Educational data mining: a review of the state of the art. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40, 6 (2010), 601–618.

[17] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*. 2234–2242.

[18] Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison* 52, 55-66 (2010), 11.

[19] Weak Supervision: The New Programming Paradigm for Machine Learning 2016. Weak Supervision: The New Programming Paradigm for Machine Learning. (2016). https://hazyresearch.github.io/snorkel/blog/ws_blog_post.html.

[20] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. (2002).

Appendix

Here we provide two examples of video transcripts, corresponding course title and descriptions, as well as predicted skills using the proposed model.

5.1 Example 1: A Java Course

This example shows video transcripts from a Java course and corresponding predicted skills. The order of the skills is based on the predicted probabilities.

Course Title: Learning Java Applications (2012)

Course Description: An introduction to developing Java applications for various runtime environments.

Video Transcripts: Now we'll write the code to connect to our database, so make sure first that your database server is running and that Tomcat is running inside of Eclipse and that you have installed the appropriate MySQL driver. So I'm in data.jsp. I am going to double-click the tab so that I can get a full view of just my code. You can always double-click it again to go back to the standard view. The first thing we are going to do is write some import statements. So open bracket, a percent sign and that symbol, we will close that out with a percent sign and a greater than sign. And inside of this tag I'm going to type page, space, import, an equal, and in quotes I am going to type in the classes that I want to import, so java.sql.*. And then I'm going to copy and paste this code right after itself to eliminate whitespace. So I'll paste that using the keyboard shortcut Command+C or Ctrl+C to copy and Command+V or Ctrl+V to paste. And I'm going to change the import code from java.sql.* to java.io.*. And I will repeat the process and paste the code again and change the import to com.mysql.* and then finally after that, still on the same line, I am going to write a declaration that this is an XML document. So open bracket, question mark, xml, space, version= and in quotes, the version is going to be 1.0, and then we will close it out with question mark and a greater than sign. So what we have done so far is just declare the classes that we want to import and use in Java and then outside of our Java code we have declared that this is an XML file. So let's go to the next line, and we are going to create the tours tag. So <tours>, and on the next line we will close that out with /tours tag. So we have an opening closing tag called to tours. And on the next line, inside of our tours tag, I'm going to write some Java code. So I will need to type a less than sign and a percent sign and Eclipse will autocreate the closing tag for me. And in here I'm going to initialize some variables that we will be using. So connection is the data type, and we will call this connection lowercase. We will just initialize it to null. We will do the same thing for a Statement. Statement is the class, statement lowercase is the name of the variable, set it equal to null, and the same thing for ResultSet. Capital R and capital S for the class name. And then result for the name, and again, we will initialize to null here. Now let's go down a few lines, and we need to connect to the database in a try catch block. So try and then some brackets and below the brackets, catch. The error is going to be of type SQLException. We will name it e. I will put the brackets there, and we will just type out .println and in the parentheses, pass in the string, "error connecting to database." No return to the try block. In there, we are going to create a new instance of the driver class by typing Class, capital C, .forName, and the name will pass in as a string so quotes and then pass in com.mysql.jdbc.Driver, with a capital D. And then after the close parenthesis type a dot and then newInstance. On the next line we will set the value of our connection. So connection all lowercase. We will set that equal to DriverManager, capital D, capital M, .getConnection, and then we pass in three parameters that are all strings. First the database url, then the username for the database and the password for the database. So in quotes url is going to be jdbc:mysql://localhost:3306. Remember that's the port for your MySQL server. And if you change it to something else in your settings, make sure you put that number here. And then /tours. That's the name of our database. After that string, type a comma. The next string is the username of our database. I kept mine at the default, which is root. And finally, the third parameter is also a string that's the password for the database, and I kept mine at the default as well, which is root. Make sure to add a semicolon after the end of the statement. And then here I am just going to type out.println, and in parentheses I will pass in the string "connected to database." So this is going to happen if there's no problem. So when we test this we should see either "connected to database" or "error connecting to database." So let's save the file and then test this using Tomcat. I am going to click OK to restart the server. And then in Eclipse's built-in browser I should see a message that says "connected to database." So that successfully connected to the database. If you have any problems with this, make sure that you're running both your MySQL server and Tomcat, and then check to see that the jar file is in the correct location so that you can load in the drivers. Finally, of course, make sure to check all of your code against mine, including the port number for your MySQL server.

Predicted Skills: Refactoring, Object oriented design, IntelliJ IDEA, Subversion, Standard widget toolkit, Facelets, Apache ant, Java architecture for XML binding, JSON, Maven, Java concurrency,

Spring security, Log4j, Smart GWT, Spring boot, Libgdx, Text driven development, Play framework, Mockito, JavaFX, JavaServer Faces, Junit, Jboss seam, Eclipse RCP, Dropwizard, Zend framework, LAMP, OSGi, MyBatis, Netty, Guice, Winforms, Multithreading, Primefaces, Thymeleaf, PureMVC, Vaadin, Eclipselink, Aspect oriented programming, Wicket.

5.2 Example 2: A Music Course

This example shows video transcripts from a music course and corresponding predicted skills. Similarly, the order of the skills is based on the predicted probabilities.

Course Title: Drum Setup and Mic'ing in the Studio

Course Description: GRAMMY-winning recording engineer Ryan Hewitt explains his techniques for capturing drum sounds. "Drum Doctor" Ross Garfield consults with Ryan on a real-world studio setup for a session with A-list drummer Josh Freese.

Video Transcripts: There's a million ways to mic a drum kit. It's one of the most complicated instruments to record, 'cause there's so many things happening on it over a vast space in your studio. So what we're going to talk about right now is the way I mike drums for, you know, my typical rock, modern, pop kind of set up where you want a tight, punchy, controlled sound. So we're going to use a lot of close mics on each individual drum, we'll have a set of room mics that we'll play with, and then we'll have what I call my stunt mics, you know, the mics that are sort of the filler sound, if you will. Those are the ones I crush up, make them sound really nasty, and get a lot of character for the drum kit. One of the important things I think about with micing drums is what the mic is literally seeing. I mean, you know, hearing, but seeing. If you think about a microphone as having like a flashlight beam, you know, these are cardioid microphones here, a condenser and a dynamic, and they see like a flashlight. They're going to look in a particular direction. So if you take this guy, he's going to see, you know, something like this, as its concentrated sound source. It'll still pick up some things around the side, you know, 'cause the cardioid pattern is really like this, but the main thrust of it is going to be like a big flashlight beam. So when you think about where these are pointing, you'll know where the sound it's going to pick up is coming from. So it's sort of crucial to think about that when choosing microphones, and then putting them in front of the instrument that you are recording. Even though we're doing tight micing, we don't want the microphones to be so close to the source that they're picking up a very small portion of the drum, you know, or the drum head, the side of the drum. We want it to have a slightly more broad picture of that drum so that it's getting the maximum tonal quality and characteristic and volume of that drum, while balancing that with the spill, with the leakage that's coming from the other parts of the drum kits. So if you have a mic on the tom, you need to be careful of the bleed from the cymbals, and the bleed from the snare drum, stuff like that. Leakage is inevitable. We just want it to sound as good as possible. So that's where mic choice comes in, and aiming the mic properly at that particular drum. So the microphones that I'm going to use today are the microphones that are available at this studio. EastWest has a great selections of microphones, and I'm pretty lucky to have all the ones I like to use. That said, there are many, many, many mics you can substitute in all of these applications. We're going to use some really expensive stuff, but there are analogous microphones that are a lot less expensive that will sound very good. Is it going to sound as good as a FET 47? I don't know. Maybe not. But it's going to sound good for what you have. And it's important to remember that if you have a good microphone on a really good-sounding drum, it's going to sound good. This is my current favorite, the Beyer M 88. So I can start with this one, and if I'm not stoked on it I can try something else and move on to the next one. But as I say, if you have one microphone that's going to work on the kick drum, we'll make it work. You can follow some basic guidelines that we'll discuss, and find the sweet spot for that microphone to make it sound the best you can. Micing a drum kit is a difficult undertaking because of all the individual pieces. The important thing to remember is that all these pieces make up a whole. And we want to hear the drum kit as a whole. So we'll have the close mics, we'll have some far mics, we'll have the stunt mics, and we have to make them all work together. So with that in mind, let's mic up the drum kit.

Predicted Skills: R&B, Music remixing, Blues, Original music, Music supervision, MIDI, Studio recording, Audio mastering, Audio recording, Film scoring, Pop, Audio engineering, Lyrics, Music publishing, Professional audio, Sound reinforcement, Music production, Songwriting, A&R administration, Song production.

Bootstrapping Chatbots for Novel Domains

Petr Babkin, Md Faisal Mahbub Chowdhury, Alfio Gliozzo, Martin Hirzel, Avraham Shinnar
IBM Research AI
Yorktown Heights, NY
{petr.babkin, mchowdh, gliozzo, hirzel, shinnar}@us.ibm.com

Abstract

We tackle the problem of automatically generating chatbots from Web API specifications using embedded natural language metadata, focusing on the intent classification subtask. One of the main challenges for such a use case comes from the lack of a sufficiently representative training sample for utterance classification, which hinders the traditional supervised model's ability to generalize to unseen inputs. We apply several unsupervised and distantly-supervised techniques to refine the model's representation and to augment its coverage. These include sentence similarity estimation in the embedding space and decision tree learner-based feature selection leveraging structural regularities in API specifications. In addition, we harness linguistic resources such as web corpora, PPDB, and WordNet. We use the resulting representation for prediction and perform a small-scale evaluation, where our approach compares favorably to the supervised baseline.

1 Introduction

Conversational bots serving Web APIs are becoming ubiquitous as nearly every major eCommerce company nowadays employs a chatbot front-end to their services. In turn, virtually every major player in the AI arena offers a product to support this [1, 2, 10, 11]. But with this interesting use case comes the technical challenge of scaling the dialogue and language models to novel APIs with minimal hand-labeling, feature engineering, and re-training for each individual domain.

Recently, there has been much progress in developing conversational systems based, most notably, on generative sequence-to-sequence recurrent architectures [7, 13, 21, 24]. Alas, these approaches require dialogue corpora (sets of question-answer or utterance-logical form pairs) to train them, which are not always available for an arbitrary domain of interest. That becomes a serious limitation when the dialogue domain is not pre-defined but is specified dynamically, e.g., by a supplied Web API [23]. While the dynamic generation of rudimentary chatbots is feasible, they come nowhere close in flexibility and robustness to models trained on large volumes of data. For the purpose of this paper we will focus on the intent classification subtask. Traditionally, the lack of training data for an NLP task is addressed by resorting to unsupervised and weakly supervised methods such as expectation-maximization and instance-based learning. Various techniques such as bootstrapping have been successfully employed on top, in some cases matching the performance of fully-supervised methods [6]. In recent years, pre-trained word embeddings have become a standard to improving the robustness of models trained on limited amounts of data on a wide array of NLP tasks, including sentence classification [12]. Another option that gained much popularity in the recent decade is transfer learning-based domain adaptation [17]. In this paper we pursue several of these techniques, contributing a novel feature selection method to the mix.

2 The Domain

The data at our disposal is a collection of 566 Swagger [16] files describing publicly available Web APIs for services ranging from language translation to ride sharing. Internally, each Swagger is a

Workshop at NIPS on Learning with Limited Labeled Data (LLD), December 2017

JSON-formatted specification of available endpoints, methods, parameters and return values. While the specification is intended to be machine-readable, it does contain natural-language annotation fields such as descriptions. Figure 1 is an excerpt detailing the interface of the language identification endpoint from a language-translation API [9].

```
1  "swagger": "2.0",
2  "info": {"version": "2.0.0", "title": "Language Translator"}, ...
3  "basePath": "/language−translator/api",
4  "paths": {
5    "/v2/models": {"get": ..., "post": ...},
6    "/v2/models/{model_id}": {"get": ..., "delete": ...},
7    "/v2/translate": {"get": ...},
8    "/v2/identifiable_languages": {"get": ...},
9    "/v2/identify": {"get": {
10     "operationId": "identifyLanguageGet",
11     "summary": "Identifies the language of the input text",
12     "description": "The return value is the two−letter language code for the identified language.",
13     "parameters": [{
14       "name": "text",
15       "description": "Input text in UTF−8 format."}],
16     "responses": {"200": {
17       "identifiedLanguages": {
18         "description": "A ranking of identified languages with confidence scores.",
19         "type": "array",
20         "items": {
21           "identifiedLanguage": {
22             "language": {
23               "description": "The code for an identified language.",
24               "type": "string" },
25            "confidence": {
26              "description": "The confidence score for the identified language.",
27              "type": "number" }, ...
```

Figure 1: Excerpt from the Translation API Swagger.

3 Approach

Our baseline is a proprietary compiler [23] that generates dialogue workspaces for Watson Conversation Service (WCS) [10]. It deterministically extracts endpoint and parameter names from Swagger and supplies them as training examples to the WCS classifiers for *intents* and *entities*, respectively.

While the resulting dialogue system is able to classify user utterances with reasonable accuracy, it only understands a limited vocabulary and forces the user to use stylized language where the input words have to match the keywords it learned. This limitation makes for unnatural user interaction. For example, consider the verbose utterance (a) supported by the system and its more implicit equivalent (b) that is beyond the capability of the underlying ngram-based model.

identify the language of text 'vive' (a)
is vive **spanish**? (b)

The underlined words in (a) match the keywords extracted from Lines 9 and 14 of the Swagger in Figure 1, reflecting the names of the service endpoint (identify) and its required parameter (text). In contrast, the utterance (b) contains none of the overt keywords and requires subtle inference that the word spanish is related to language. Furthermore, the baseline model is insensitive to syntax, allowing utterances to match a wrong command based on word overlap, as in utterance (c):

what languages **can** you identify? (c)

While Swagger files contain more information, using it effectively is challenging. First, the amount of textual data describing each endpoint is prohibitively low to be used for training directly, as the resulting model will simply overfit to linguistic idiosyncrasies of individual examples rather than learning useful generalizations pertaining to underlying intents. Second, different endpoints within the same API are topically related and thus share much of the vocabulary (e.g., in the Translation API, the description of most endpoints involves the word "language", and in the Lyft ride sharing

API, many commands have to do with rides). The shared vocabulary makes topic modeling-based approaches such as LSI [4] infeasible, which warrants the use of a more fine-grained representation.

To combat the data deficit we use word embeddings and a parsing model pre-trained on large corpora [8]. In addition, we bootstrap our model with information from external linguistic resources such as PPDB [5] and WordNet [14]. Finally, we opt for a finer-grained feature representation that exploits structural regularities in Swagger. The following sections detail the specifics of our approach.

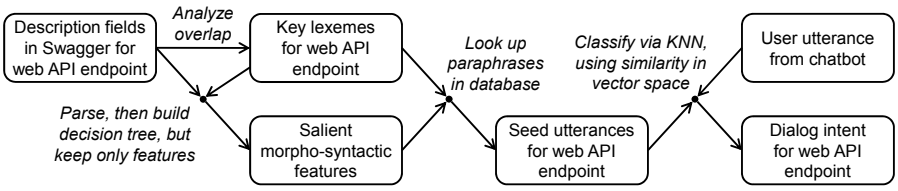


Figure 2: Overview of our approach

3.1 Estimation of Semantic Similarity under Vector Space Model

The first step to making our model more robust is to abstract away from discrete string matching. This can be accomplished by introducing word embeddings, which enable a fuzzier notion of *semantic relatedness* under vector space model (VSM). Thus, we can replace the binary notion of whether an item matches a certain keyword with a continuous similarity measure given by the cosine of the angle between the corresponding dense vectors. We used 300-dimensional vectors pre-trained with the GloVe algorithm [19] on the Common Crawl corpus that come with the gensim Python library [20]. By applying this model, for the words "spanish" and "language" we get a significant similarity score close to 0.4 (for reference, the similarity between "spanish" and "english" is around 0.8 and between "language" and "languages" is close to 0.9 under the same model, whereas the similarity among unrelated words is usually below 0.1)

Next comes a question of how to compute similarity among clauses and complete sentences. There are a number of approaches to embedding composition including averaging of individual word embeddings, their weighted addition, multiplication, dilation, circular convolution (cf. [15] for a good overview) and even subtraction [3]. Some more advanced, syntax-aware approaches include compositional vector grammars [22] and recursive tree autoencoders [25]. After experimenting with a few different models we found the following simple similarity metric works best:

$$sim(\vec{u}, \vec{s}) = \sum_i \max_j \cos(\vec{u}_i, \vec{s}_j) \tag{1}$$

where vectors $\vec{u}$ (user utterance) and $\vec{s}$ (seed utterance) represent sentences such that their ith or jth component is a GloVe embedding of the ith or jth word, respectively. This equation is not symmetric in that it sums over the component embeddings of the user utterance, for each aiming to find the closest component of the seed utterance. This is motivated by the need to assign higher scores to those seeds that match not only based on intent (as in simple max or averaging) but also on arguments. We use this metric in K-nearest neighbors classification of incoming user utterances.

3.2 Selecting Optimal Representation

Computing the similarity from raw annotations without preprocessing resulted in suboptimal performance. When using a VSM, we want to make sure that only the invariant features fundamentally characteristic of the underlying intent are considered while mere wording idyosyncracies are ignored. The approach we took with respect to selecting an optimal representation was a) to determine salient lexemes and their salient morpho-syntactic configurations, and b) reduce raw annotations to "compressed" form while preserving their discriminative attributes. A simple way to identify potentially salient words is to look for lexemes that overlap across different fields describing the same endpoint.

The lemmas "language" and "identify" are most prevalent in the /v2/identify endpoint. However, one should be wary that due to substantial lexical overlap across classes and the low volume of available data, lexical features alone might not suffice. Thus, we can consider the key lexemes in conjunction with their functional role in the sentence, which are likely to be different across classes.

Table 1: Lexically overlapping but functionally distinct descriptors of two different endpoints.

/v2/identify	/v2/identifiable_languages
VBZ dobj NN **identifies** the **language** of the input text	NNS relcl VBN lists all **languages** that can be **identified** by the api
VBN amod NNS a ranking of **identified languages** with confidence scores	NNS relcl VB a list of all **languages** that the service can **identify**
VBN amod NN a code for an **identified language**	JJ amod NN the code for an **identifiable language**
VBN amod NN the confidence score for the **identified language**	JJ amod NN the name of the **identifiable language**

For example, consider syntactic dependencies indicated in Table 1. The same word that acts as a head verb (identifies) or a verbal modifier (identified) in the descriptions of the first endpoint, takes on the adjectival modifier (identifiable) and the relative clause verb roles in texts describing the second one. We capture these fine-grained distinctions by decomposing each key lexeme into its base form, morphological features (e.g., part of speech, number, etc.), and syntactic features (e.g., dependency relation, subordinate and superordinate constituents). Next, to select the most salient subset of features we employ the information gain-based decision tree algorithm provided by the *sklearn* library [18]. To illustrate, Table 2 shows the two features selected by this procedure in order to perfectly discriminate between the two classes in the example. Importantly, the resulting feature hierarchy is not used for classification but rather as a "filter": stripping down the seed examples to key lexemes exhibiting feature configurations that it identified to be salient to each class.

Table 2: The resulting salient feature space w.r.t. the lexeme "identify" obtained for the two classes.

	verb (VB, VBZ, VBN)	adjective (JJ)
adjectival modifier (amod)	/v2/identify	/v2/identifiable_languages
direct object (dobj)	/v2/identify	/v2/identifiable_languages
relative clause (relcl)	/v2/identifiable_languages	/v2/identifiable_languages

3.3 Data Augmentation and Boosting

After the seed utterances have been reduced to their minimal representations in terms of the identified set of discriminative features, we enrich them using external linguistic resources, namely the paraphrase database (PPDB) [5] to widen the system's coverage. PPDB was collected with a high precision method relying on parallel corpora and statistical co-occurence measures, which ensures high quality paraphrases. Functionally, this database is not simply a thesaurus of synonyms but also includes phrasal and syntactic paraphrases. For example, using PPDB, a clause "give me a ride" from the ride sharing domain can be paraphrased in multiple interesting ways including a subtle "pick me up". In addition, we conducted a limited series of trials with incorporating user feedback — by adding correctly labeled utterances to the pool of seed examples as a form of boosting our classifier. However, we observed the impact on the performance to be too unstable and, due to the limited time for the project, did not pursue this direction further.

4 Results

We performed a small-scale evaluation on three different Web API Swaggers — Watson Translation, Lyft, and Petstore. For each, we elicited approximately 30 – 40 utterances from 4 annotators who use Web API Swaggers in their work. They were encouraged to provide concise yet informal commands to the bot as implicit as they would to a friend in a chat. As part of the annotation, they indicated the intended endpoint (from the list in the API Swaggers) for every typed utterance. Agreement scores were not formally measured.

As expected, simple synonymy and lexical paraphrase got picked up easily by the system. For example, "remove model" in the translation API was correctly recognized as endpoint "delete

translation model", and "abort mission to new york" from the Lyft API was resolved to the "cancel ride" endpoint. A significant fraction of more implicit renderings that do not contain command identifying verbs were also correctly recognized; for example, "what language is this: saionara?" as "identify language" and "i don't want this ride" as "cancel ride". Interestingly, in such subtle cases, certain stop words such as propositions came out as crucial: "guten tag from german to english" (translate) as the seed descriptions contained source and destination languages to translate "from" and "to". In addition, type vs. token distinction played a role as language names were found similar to the word "language".

However, there is a limit to the semantic subtlety that our approach can handle. For example, a very open ended "what languages do you speak anyway?" utterance (intended: list identifiable languages) and even more implicit ones such as "i wanna go somewhere else" (intended: "change destination") require more sophisticated semantic reasoning going beyond sheer similarity. Quantitatively, the accuracies varied greatly across domains. Nevertheless, our approach consistently improved over the baseline (which is described in Section 3). Figure 3 shows an absolute improvement over the baseline of 0.33 for Translation and 0.14 for Lyft. Judging the improvement on the Petstore test set proved difficult due to a lack of reliable numbers of baseline performance (the baseline compiler wrongly merged multiple intents together in this Swagger). The absolute accuracy of 0.44 (across 19 classes), while not ideal, is still significantly better than random ($1/19 \approx 0.05$).

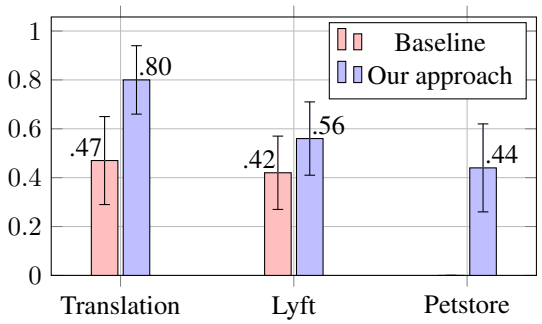


Figure 3: Classification accuracy comparison

In addition, for the Translation domain, we compared against the performance on raw (i.e. unprocessed) utterances, which as one would expect resulted in performance not only inferior to our main approach but even slightly worse than the deterministic baseline (0.37). Although we did not perform a formal ablation study, the runner-up configuration (max similarity and no PPDB) achieved 0.67 score. Examination of confusion matrices revealed that most errors occur between semantically similar classes, e.g., "get rides" and "get ride types". Among miscellaneous observations, "give rating" was sometimes confused with "cancel ride" probably in part based on the negative sentiment of the corresponding utterance such as "very bad driver, 0 stars" that was mistakenly considered as related to a slightly negatively charged word "cancel". In addition, utterances invoking "get cost estimate" such as "how much for the ride", or "how much do i pay" were confused with a broad variety of endpoints whose corresponding utterances also contain question words, for example: "when will my ride arrive" (get ETA), "who are the passengers" (get profile), and "what types of rides are available from this location" (get ride types).

5 Conclusion

Learning NLP models from very limited training data remains a challenge that does not have a universal solution. In this project we experimented with multiple unsupervised and weakly supervised techniques and, in addition, harnessed our knowledge of the nature and format of our data. We achieved moderate success in a hard problem of multiclass sentence classification with just a handful of training examples. In particular, we found representation augmentation in combination with automatic feature selection to be especially beneficial in a limited data setting. But while our approach worked really well in one domain, this did not trivially transfer to others. This underscores the importance of a reliable proxy metric for estimating and tuning model performance prior to the evaluation on a labeled test set. We leave a formal investigation on that subtopic for future work in addition to other potentially fruitful avenues of training on noisy data and domain adaptation.

References

[1] Amazon. Lex, 2017. https://aws.amazon.com/lex/ (Retrieved November 2017).

[2] api.ai, 2010. https://api.ai/ (Retrieved November 2017).

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 2787–2795, 2013.

[4] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[5] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 758–764, 2013.

[6] Alfio Gliozzo, Carlo Strapparava, and Ido Dagan. Improving text categorization bootstrapping via unsupervised learning. *Transactions on Speech and Language Processing (TSLP)*, 6(1):1:1–1:24, October 2009.

[7] Dilek Hakkani-Tür, Gokhan Tur, Asli Celikyilmaz, Yun-Nung Vivian Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Conference of the International Speech Communication Association (INTERSPEECH)*, pages 715–719, 2016.

[8] Matthew Honnibal and Mark Johnson. An improved non-monotonic transition system for dependency parsing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1373–1378, 2015.

[9] IBM. Watson Language Translator service, 2015. https://www.ibm.com/watson/developercloud/language-translator.html (Retrieved October 2017).

[10] IBM. Watson Conversation service, 2016. https://www.ibm.com/watson/developercloud/conversation.html (Retrieved October 2017).

[11] kitt.ai, 2016. https://kitt.ai (Retrieved November 2017).

[12] Amit Mandelbaum and Adi Shalev. Word embeddings and their use in sentence classification tasks, 2016. https://arxiv.org/abs/1610.08229.

[13] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. Using recurrent neural networks for slot filling in spoken language understanding. *Transactions on Audio, Speech, and Language Processing (TASLP)*, 23(3):530–539, March 2015.

[14] George A. Miller. WordNet: A lexical database for English. *Communications of the ACM (CACM)*, 38(11):39–41, November 1995.

[15] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 236–244, 2008.

[16] OpenAPI Initiative. OpenAPI specification (fka swagger restful api documentation specification), 2014. https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md (Retrieved November 2017).

[17] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Transactions on Knowledge and Data Engineering (ToKaDE)*, 22:1345–1359, October 2010.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

[19] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[20] Radim Řehůřek and Petr Sojka. Software framework for topic modelling with large corpora. In *Workshop on New Challenges for NLP Frameworks*, pages 45–50, 2010.

[21] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808, 2015.

[22] Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. Parsing with compositional vector grammars. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 455–465, 2013.

[23] Mandana Vaziri, Louis Mandel, Avraham Shinnar, Jérôme Siméon, and Martin Hirzel. Generating chat bots from web API specifications. In *Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward!)*, pages 44–57, 2017.

[24] Oriol Vinyals and Quoc V. Le. A neural conversational model. In *Deep Learning Workshop*, 2015.

[25] Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over recursive structures. In *International Conference on Machine Learning (ICML)*, pages 1604–1612, 2015.

Classification and Disease Localization in Histopathology Using Only Global Labels: A Weakly-Supervised Approach

Pierre Courtiol[*], **Eric W. Tramel**[*], **Marc Sanselme, and Gilles Wainrib**
Owkin, Inc.
New York City, NY
{firstname.lastname}@owkin.com

Abstract

Analysis of histopathology slides is a critical step for many diagnoses, and in particular in oncology where it defines the gold standard. In the case of digital histopathological analysis, highly trained pathologists must review vast whole-slide-images of extreme digital resolution ($100,000^2$ pixels) across multiple zoom levels in order to locate abnormal regions of cells, or in some cases single cells, out of millions. The application of deep learning to this problem is hampered not only by small sample sizes, as typical datasets contain only a few hundred samples, but also by the generation of ground-truth localized annotations for training interpretable classification and segmentation models. We propose a method for disease localization in the context of weakly supervised learning, where only image-level labels are available during training. Even without pixel-level annotations, we are able to demonstrate performance comparable with models trained with strong annotations on the Camelyon-16 lymph node metastases detection challenge. We accomplish this through the use of pre-trained deep convolutional networks, feature embedding, as well as learning via top instances and negative evidence, a multiple instance learning technique from the field of semantic segmentation.

1 Introduction

Histopathological image analysis (HIA) is a critical element of diagnosis in many areas of medicine, and especially in oncology, where it defines the gold standard metric. Recent works have sought to leverage developments in machine learning (ML) to aid pathologists in disease detection tasks, but the majority of these techniques require localized annotation masks as training data. These annotations are even more costly to obtain than the original diagnosis, as pathologists must spend time to assemble pixel-by-pixel segmentation maps of diseased tissue at extreme resolution, thus HIA datasets with annotations are very limited in size. Additionally, such localized annotations may not be available when facing new problems in HIA, such as new disease subtybe classification, prognosis estimation, or drug response prediction. Thus, the critical question for HIA is: can one design a learning architecture which achieves accurate classification with no additional localized annotation? A successful technique would be able to train algorithms to assist pathologists during analysis and could also be used to identify previously unknown structures and regions of interest.

Indeed, while histopathology is the gold standard diagnostic in oncology, it is extremely costly, requiring many hours of focus from pathologists to make a single diagnosis (Litjens et al., 2016; Weaver, 2010). Additionally, as correct diagnosis for certain diseases requires pathologists to identify a few cells out of millions, these tasks are akin to "finding a needle in a haystack." In digital

[*]Equal contribution to this work.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

pathology with whole-slide-imaging (WSI) (Yagi & Gilbertson, 2005; Snead et al., 2016), highly trained and skilled pathologists review digitally captured microscopy images from prepared and stained tissue samples in order to make diagnoses. Digital WSI are massive datasets, consisting of images captured at multiple zoom levels. At the greatest magnification levels, a WSI may have a digital resolution upwards of 100 thousand pixels in both dimensions. However, since localized annotations are very difficult to obtain, one is often left with image-level global labels. Thus, obtaining predictions localized within the WSI becomes a difficult task commonly referred to as *weakly-supervised learning*.

In this paper, we propose CHOWDER[2], an approach for the interpretable prediction of general localized diseases in WSI with only weak, whole-image disease labels and without any additional expert-produced localized annotations, i.e. per-pixel segmentation maps, of diseased areas within the WSI. We accomplish this through the use of a novel aggregation technique performed on features extracted by a pre-trained DCNN on the tile-level. Notably, while the approach we propose makes use of a pre-trained model, the entire procedure is a true end-to-end classification technique. Thus, pre-trained DCNN feature-extraction layers can be fine-tuned to the context of haematoxylin and eosin (H&E) stained WSI. We demonstrate, using only whole-slide labels, performance comparable to top-10 ranked methods trained with strong, pixel-level labels on the Camelyon-16 challenge dataset, while also producing disease segmentation that closely matches ground-truth annotations. We also present results for diagnosis prediction on WSI obtained from The Cancer Genome Atlas (TCGA), where strong annotations are not available and diseases may not be strongly localized in the tissue sample.

2 Baseline: Global Feature Aggregation

Given their scale, it is necessary to process WSI in a piecemeal fashion, taking in image data for a given magnification level tile-by-tile. Following the pre-processing pipeline described in Appendix A, extracting tile-level features produces a bag of feature vectors which one attempts to use for classification against the known image-wide label. The dimension of these local descriptors is $M^{\mathrm{S}} \times P$, where P is the number of features output from the pre-trained image DCNN and M^{S} is the number of sampled tiles. Approaches such as Bag-of-visual-words (BoVW) or VLAD (Jégou et al., 2010) could be chosen as a baseline aggregation method to generate a single image-wide descriptor of size $P \times 1$, but would require a huge computational power given the dimensionality of the input. Instead, we will try two common approaches for the aggregation of local features, specifically, the `MaxPool` and `MeanPool`.

After applying these pooling methods over the axis of tile indices, one obtains a single feature descriptor for the whole image. Other pooling approaches have been used in the context of HIA, including Fisher vector encodings (Song et al., 2017) and $p-$norm pooling (Xu et al., 2017). However, as the reported effect of these aggregations is quite small, we don't consider these approaches when constructing our baseline approach. After aggregation, a classifier can be trained to produce the desired diagnosis labels given the global WSI aggregated descriptor. For our baseline method, we use a logistic regression for this final prediction layer of the model.

3 CHOWDER Method

In experimentation, we observe that the baseline approach of the previous section works well for *diffuse* disease, which is evidenced in the results of Table 1 for `TCGA-Lung`. Here, *diffuse* implies that the number of disease-containing tiles, pertinent to the diagnosis label, are roughly proportional to the number of tiles containing healthy tissue. However, if one applies the same approach to different WSI datasets, such as `Camelyon-16`, the performance significantly degrades. In the case of `Camelyon-16`, the diseased regions of most of the slides are highly localized, restricted to a very small area within the WSI. When presented with such imbalanced bags, simple aggregation approaches for global slide descriptors will overwhelm the features of the disease-containing tiles. Instead, we propose an improvement of the WELDON method (Durand et al., 2016), adapting it for use in HIA. As in Durand et al. (2016), rather than creating a global slide descriptor by aggregating

[2]Classification of HistOpathology with Weak supervision via Deep fEature aggRegation

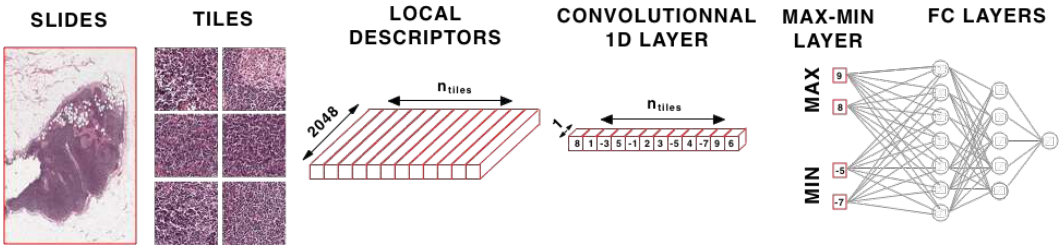


Figure 1: Description of the CHOWDER architecture (for $R = 2$) for WSI classification via MLP on operating on top positive and negative instances shown for a single sample mini-batch sample.

all tile features, instead a *multiple instance learning* (MIL) approach is used that combines both top-instances as well as negative evidence. A visual description of approach is given in Fig. 1.

First, a set of one-dimensional embeddings for the $P = 2048$ ResNet-50 features are calculated via J one-dimensional convolutional layers strided across the tile index axis. For tile t with features $\mathbf{k}_t$, the embedding according to kernel j is calculated as $e_{j,t} = \langle \mathbf{w}_j, \mathbf{k}_t \rangle$. Notably, the kernels $\mathbf{w}_j$ have dimensionality P. This one-dimensional convolution is, in essence, a shortcut for enforcing a fully-connected layer with tied weights across tiles, i.e. the same embedding for every tile (Durand et al., 2016). In our experiments, we found that the use of a single embedding, $J = 1$, is an appropriate choice for WSI datasets when the number of available slides is small (< 1000). In this case, choosing $J > 1$ will decrease training error, but will *increase* generalization error. Avoiding overtraining and ensuring model generality remains a major challenge for the application of WSL to WSI datasets.

After feature embedding, we now have a $M^{\mathrm{S}} \times 1$ vector of local tile-level (*instance*) descriptors. As in Durand et al. (2016), these instance descriptors are sorted by value. Of these sorted embedding values, only the top and bottom R entries are retained, resulting in a vector of $2R \times 1$ entries to use for diagnosis classification. This can be easily accomplished through a `MinMax` layer on the output of the one-dimensional convolution layer. The purpose of this layer is to take not only the top instances region but also the negative evidence, i.e. the region which best support the absence of the class. During training, the back-propagation runs only through the $2R$ selected tiles, the positive and negative evidence. When applied to WSI, the `MinMax` serves as a powerful tile selection procedure.

In the WELDON architecture, the last layer consists of a sum applied over the $2R \times 1$ output from the `MinMax` layer. However, we find that this approach can be improved for WSI classification. We investigate the possibility of richer interactions between the top and bottom instances by instead using an MLP as the final classifier. In our implementation of CHOWDER, we use an MLP with two fully connected layers of 200 and 100 neurons with sigmoid activations.

Lastly, while the CHOWDER architecture is trained predict WSI classification, the trained model may be modified during inference to permit tumor localization, as well. Specifically, for a set of tiles drawn from a given WSI, the output of the feature embedding layer is used as an indicator of tile-level classification. In the specific case of $J = 1$, the magnitude of the scalar value output for each tile is used to represent the relative strength of classification of the tile-level instance between binary classes (e.g. "healthy", "tumor"). For high-quality localization maps, a large number of overlapping tiles may be drawn and their inferred instance-level values fused together into a single map.

4 Experimental Results

For pre-processing, we fix a single tile scale for all methods and datasets. We chose a fixed zoom level of 0.5 μm/pixel, which corresponds to $\ell = 0$ for slides scanned at 20x magnification, or $\ell = 1$ slides scanned at 40x magnification. Next, since WSI datasets often only contain a few hundred images, far from the millions images of ImageNet dataset, strong regularization required prevent over-fitting. We applied ℓ_2-regularization of 0.5 on the convolutional feature embedding layer and dropout on the MLP with a rate of 0.5. However, these values may not be the global optimal, as we did not apply any hyper-parameter optimization to tune these values. To optimize the model parameters, we use Adam (Kingma & Ba, 2014) to minimize the binary cross-entropy loss over 30 epochs with a mini-batch size of 10 and with learning rate of 0.001.

To reduce variance and prevent over-fitting, we trained an ensemble of E CHOWDER networks which only differ by their initial weights. The average of the predictions made by these E networks establishes the final prediction. Although we set $E = 10$ for the results presented in Table 1, we used a larger ensemble of $E = 50$ with $R = 5$ to obtain the best possible model and compare our method to those presented in Table 1. We also use an ensemble of $E = 10$ when reporting the results for WELDON. As the training of one epoch requires about 30 seconds on our available hardware, the total training time for the ensemble took just over twelve hours. While the ResNet-50 features were extracted using a GPU for efficient feed-forward calculations, the CHOWDER network is trained on CPU in order to take advantage of larger system RAM sizes, compared to on-board GPU RAM. This allows us to store all the training tiles in memory to provide faster training compared to a GPU due to reduced transfer overhead.

TCGA. The public Cancer Genome Atlas (TCGA) provides approximately 11,000 tissue slides images of cancers of various organs[3]. For our first experiment, we selected 707 lung cancer WSIs (`TCGA-Lung`), which were downloaded in March 2017. Subsequently, a set of new lung slides have been added to TCGA, increasing the count of lung slides to 1,009. Along with the slides themselves, TCGA also provides labels representing the type of cancer present in each WSI. However, no local segmentation annotations of cancerous tissue regions are provided. The pre-processing step extracts 1,411,043 tiles and their corresponding representations from ResNet-50. The task of these experiments is then to predict which type of cancer is contained in each WSI: adenocarcinoma or squamous cell carcinoma. We evaluate the quality of the classification according to the area under the curve (AUC) of the receiver operating characteristic (ROC) curve generated using the raw output predictions.

As expected in the case of diffuse disease, the advantage provided by CHOWDER is slight as compared to the `MeanPool` baseline, as evidenced in Table 1. Additionally, as the full aggregation techniques work quite well in this setting, the value of R does not seem to have a strong effect on the performance of CHOWDER as it increases to $R = 100$. In this setting of highly homogenous tissue content, we can expect that global aggregate descriptors are able to effectively separate the two classes of carcinoma.

Camelyon-16. For our second experiment, we use the `Camelyon-16` challenge dataset[4], which consists of 400 WSIs taken from sentinel lymph nodes, which are either healthy or exhibit metastases of some form. In addition to the the WSIs themselves, as well as their labeling (`healthy`, `contains-metastases`), a segmentation mask is provided for each WSI which represents an expert analysis on the location of metastases within the WSI. Human labeling of sentinel lymph node slides is known to be quite tedious, as noted in Litjens et al. (2016); Weaver (2010). Teams participating in the challenge had access to, and utilized, the ground-truth masks when training their diagnosis prediction and tumor localization models. For our approach, we set aside the masks of metastasis locations and utilize only diagnosis labels. Furthermore, many participating teams developed a post-processing step, extracting handcrafted features from predicted metastasis maps to improve their segmentation. No post-processing is performed for the presented CHOWDER results, the score is computed directly from the raw output of the CHOWDER model. We also conduct a set of experiments on `Camelyon-16` using random train-test cross-validation (CV) splits, respecting the same training set size as in the original competition split.

The `Camelyon-16` dataset is evaluated on two different axes. First, the accuracy of the predicted label for each WSI in the test set is evaluated according to AUC. Second, the accuracy of metastasis localization is evaluated by comparing model outputs to the ground-truth expert annotations of metastasis location. This segmentation accuracy is measured according to the free ROC metric (FROC), which is the curve of metastasis detection sensitivity to the average number of also positives. As in the Camelyon challenge, we evaluate the FROC metric as the average detection sensitivity at the average false positive rates 0.25, 0.5, 1, 2, 4, and 8.

In Table 1, we see the classification performance of our proposed CHOWDER method, for $E = 10$, as compared to both the baseline aggregation techniques, as well as the WELDON approach. In the case of WELDON, the final MLP is not used and instead a summing is applied to the `MinMax` layer. The value of R retains the same meaning in both cases: the number of both high and low

[3] `https://portal.gdc.cancer.gov/legacy-archive`

[4] `https://camelyon16.grand-challenge.org`

Method	Camelyon		TCGA
	CV	*Competition*	
BASELINE			
`MaxPool`	0.749	0.655	0.860
`MeanPool`	0.802	0.530	0.903
WELDON			
$R = 1$	0.782	0.765	—
$R = 10$	0.832	0.670	—
$R = 100$	0.809	0.600	—
$R = 300$	0.761	0.573	—
CHOWDER			
$R = 1$	0.809	0.821	0.900
$R = 5$	**0.903**	**0.858**	—
$R = 10$	0.900	0.843	**0.915**
$R = 100$	0.870	0.775	0.909
$R = 300$	0.837	0.652	—

Rank	Team	AUC
1	HMS & MIT	0.9935
2	HMS-MGH	0.9763
3	HMS-MGH	0.9650
	...	
10	DeepCare Inc.	0.8833
	CHOWDER	**0.8706**
11	Indep. DE	0.8654
	...	

Rank	Team	FROC
1	HMS & MIT	0.8074
2	HMS-MGH	0.7600
3	HMS-MGH	0.7289
	...	
17	SIT	0.3385
	CHOWDER	**0.3103**
18	Warwick-QU	0.3052
	...	

Table 1: *Left:* Classification (AUC) results for the `Camelyon-16` and `TCGA-Lung` datasets for CHOWDER, WELDON, and the baseline approach. For `Camelyon-16`, we present two scores, one for the fixed competition test split of 130 WSIs, and one for a cross-validated average over 3 folds (*CV*) on the 270 training WSIs. For `TCGA-Lung`, we present scores as a cross-validated average over 5 folds. *Right:* Final leader boards for `Camelyon-16` competition. All competition methods had access to the full set of strong annotations for training their models. In contrast, our proposed approach only utilizes image-wide diagnosis levels and obtains comparable performance as top-10 methods.

scoring tiles to pass on to the classification layers. We test a range of values R for both WELDON and CHOWDER. We find that over all values of R, CHOWDER provides a significant advantage over both the baseline aggregation techniques as well as WELDON. We also note that the optimal performance can be obtained without using a large number of discriminative tiles, i.e. $R = 5$.

For CHOWDER, we also propose tumor localization via the full set of outputs from the convolutional feature embedding layer. These are then sorted and thresholded according to value τ such that tiles with an embedded value larger than τ are classified as diseased and those with lower values are classified as healthy. We show an example of disease localization produced by CHOWDER in Appendix A. Here, we see that CHOWDER is able to very accurately localize the tumorous region in the WSI even though it has only been trained using global slide-wide labels and without any local annotations. While some potential false detections occur outside of the tumor region, we see that the strongest response occurs within the tumor region itself.

We also present in Table 1 our performance as compared to the public Camelyon leader boards for $E = 50$. We are able to obtain an effective 11^{th} and 18^{th} place rank for classification and localization, respectively, *but without using any of the ground-truth disease segmentation maps*. This is a remarkable result, as the winning approach of Wang et al. (2016) required tile-level disease labels derived from expert-provided annotations in order to train a full 27-layer GoogLeNet (Szegedy et al., 2015) architecture for tumor prediction.

5 Discussion

We have shown that using state-of-the-art techniques from MIL in computer vision, such as the top instance and negative evidence approach of (Durand et al., 2016), one can construct an effective technique for diagnosis prediction *and* disease location for WSI in histopathology without the need for expensive localized annotations produced by expert pathologists. By removing this requirement, we hope to accelerate the production of computer-assistance tools for pathologists to greatly improve the turn-around time in pathology labs and help surgeons and oncologists make rapid and effective patient care decisions. This also opens the way to tackle problems where expert pathologists may not know precisely where are the relevant zones in the images, for instance for prognosis estimation or prediction of drug response tasks. The ability of our approach to discover associated regions of interest without prior localized annotations hence appears as a novel discovery approach for the field of pathology. Moreover, using the suggested localization from CHOWDER, one may considerably speed up the process of obtaining ground-truth localized annotations.

References

Francesco Ciompi, Oscar Guessing, Babak Ehteshami Bejnordi, Gabriel Silva de Souza, Alexi Baidoshvili, Geert Litjens, Bram van Ginneken, Iris Nagtegaal, and Jeroen van der Laak. The importance of stain normalization in colorectal tissue classification with convolutional networks. arXiv Preprint [cs.CV]:1702.05931, 2017.

Thibault Durand, Nicolas Thome, and Matthieu Cord. WELDON: Weakly supervised learning of deep convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 4743–4752, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Reconition*, 2010.

Adnan Mujahid Khan, Nasir Rajpoot, Darren Treanor, and Derek Magee. A nonlinear mapping approach to stain normalization in digital histopathology images using image-specific color deconvolution. *IEEE Trans. Biomedical Engineering*, 61(6), 2014.

Diederik P. Kingma and Jimmy Ba. ADAM: A method for stochastic optimization. arXiv Preprint [cs.LG]:1412.6980, 2014.

Geert Litjens, Clara I. Sanchez, Nadya Timofeeva, Meyke Hermsen, Iris Nagtegaal, Iringo Kovacs, Christina Hulsbergen van de Kaa, Peter Bult, Bram van Ginneken, and Jeroen van der Laak. Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific Reports*, (6), 2016.

Dennis Nikitenko, Michael A. Wirth, and Kataline Trudel. Applicability of white-balancing algorithms to restoring faded colour slides: An empirical evaluation. *Journal of Multimedia*, 2008.

N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

David R J Snead, Yee-Wah Tsang, Aisha Meskiri, Peter K Kimani, Richard Crossman, Nasir M Rajpoot, Elaine Blessing, Klaus Chen, Kishore Gopalakrishnan, Paul Matthews, Navid Momtahan, Sarah Read-Jones, Shatrughan Sah, Emma Simmons, Bidisa Sinha, Sari Suortamo, Yen Yeo, Hesham El Daly, and Ian A Cree. Validation of digital pathology imaging for primary histopathological diagnosis. *Histopathology*, 68(7): 1063–1072, 2016.

Yang Song, Ju Jia Zou, Hang Chang, and Weidong Cai. Adapting fisher vectors for histopathology image classification. In *Proc. IEEE Int. Symp. on Biomedical Imaging*, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H. Beck. Deep learning for identifying metastatic breast cancer. arXiv Preprint [q-bio.QM]:1606.05718, 2016.

D. L. Weaver. Pathology evaluation of sentinel lymph nodes in breast cancer: Protocol recommendations and rationale. *Mod. Pathol.*, 23(Suppl 2):S26–S32, 2010.

Yan Xu, Zhipeng Jia, Liang-Bo Wang, Yuqing Ai, Fang Zhang, Maode Lai, and Eric I-Chao Chang. Large scale tissue histopathology image classification, segmentation, and visualization via deep convolutional activation features. *BMC Bioinformatics*, 18(281), 2017.

Yukako Yagi and John R. Gilbertson. Digital imaging in pathology: The case for standardization. *Journal of Telemedicine and Telecare*, 11(3):109–116, 2005.

A WSI Pre-Processing

Tissue Detection. It is possible that large regions of a WSI may contain no tissue at all. To extract only tiles with content relevant to the task, we use the same approach as Wang et al. (2016), namely, Otsu's method (Otsu, 1979) applied to the the hue and saturation channels of the image after transformation into the HSV color space to produce two masks which are then combined to produce the final tissue segmentation. Subsequently, only tiles within the foreground segmentation are extracted for training and inference.

Color Normalization. According to Ciompi et al. (2017), stain normalization is an important step in HIA since the result of the H&E staining procedure can vary greatly between any two slides. We utilize a simple histogram equalization algorithm consisting of left-shifting RGB channels and subsequently rescaling them to $[0, 255]$, as proposed in Nikitenko et al. (2008). In this work, we place a particular emphasis on the tile aggregation method rather than color normalization, so we did not make use of more advanced color normalization algorithms, such as Khan et al. (2014).

Tiling. The tiling step is necessary in histopathology analysis. Indeed, due to the large size of the WSI, it is computationally intractable to process the slide in its entirety. For example, on the highest resolution zoom level, denoted as *scale 0*, for a fixed grid of non-overlapping tiles, a WSI may possess more than 200,000 tiles of 224×224 pixels. Because of the computational burden associated with processing the set of all possible tiles, we instead turn to a uniform random sampling from the space of possible tiles. Additionally, due to the large scale nature of WSI datasets, the computational burden associated with sampling potentially overlapping tiles from arbitrary locations is a prohibitive cost for batch construction during training.

Instead, we propose that all tiles from the non-overlapping grid should be processed and stored to disk prior to training. As the tissue structure does not exhibit any strong periodicity, we find that sampling tiles along a fixed grid without overlapping provides a reasonably representative sampling while maximizing the total sampled area.

Given a target scale $\ell \in \{0, 1, \ldots, L\}$, we denote the number of possible tiles in WSI indexed by $i \in \{1, 2, \ldots, N\}$ as $M^{\mathrm{T}}_{i,\ell}$. The number of tiles sampled for training or inference is denoted by $M^{\mathrm{S}}_{i,\ell}$ and is chosen according to

$$M^{\mathrm{S}}_{i,\ell} = \min\left(M^{\mathrm{T}}_{i,\ell}\,,\ \max\left(M^{\mathrm{T}}_{\min},\ \frac{1}{2}\cdot \bar{M}^{\mathrm{T}}_{\ell}\right)\right), \tag{1}$$

where $\bar{M}^{\mathrm{T}}_{\ell} = \frac{1}{N}\sum_i M^{\mathrm{T}}_{i,\ell}$ is the empirical average of the number of tiles at scale ℓ over the entire set of training data.

Feature Extraction. We make use of the ResNet-50 (He et al., 2016) architecture trained on the ImageNet natural image dataset. In empirical comparisons between VGG or Inception architectures, we have found that the ResNet architecture provides features more well suited for HIA. Additionally, the ResNet architecture is provided at a variety of depths (ResNet-101, ResNet-152). However, we found that ResNet-50 provides the best balance between the computational burden of forward inference and richness of representation for HIA.

In our approach, for every tile we use the values of the ResNet-50 pre-output layer, a set of $P = 2048$ floating point values, as the feature vector for the tile. Since the fixed input resolution for ResNet-50 is 224×224 pixels, we set the resolution for the tiles extracted from the WSI to the same pixel resolution at every scale ℓ.

A Further Results

A.1 ROC Curves for Camelyon Competition Split

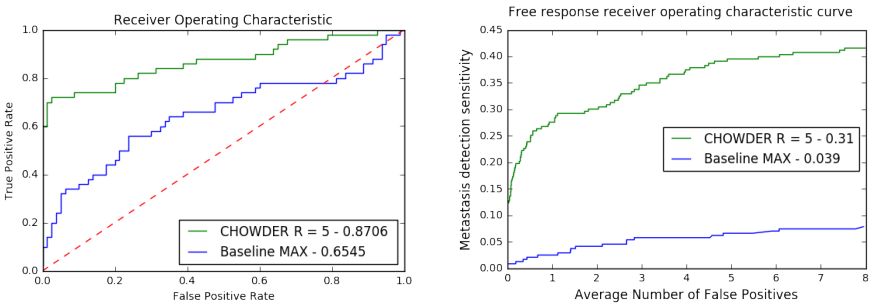


Figure 2: Performance curves for `Camelyon-16` dataset for both classification and segmentation tasks for the different tested approaches. *Left:* ROC curves for the classification task. *Right:* FROC curves for lesion detection task.

A.2 Tumor Localization Examples

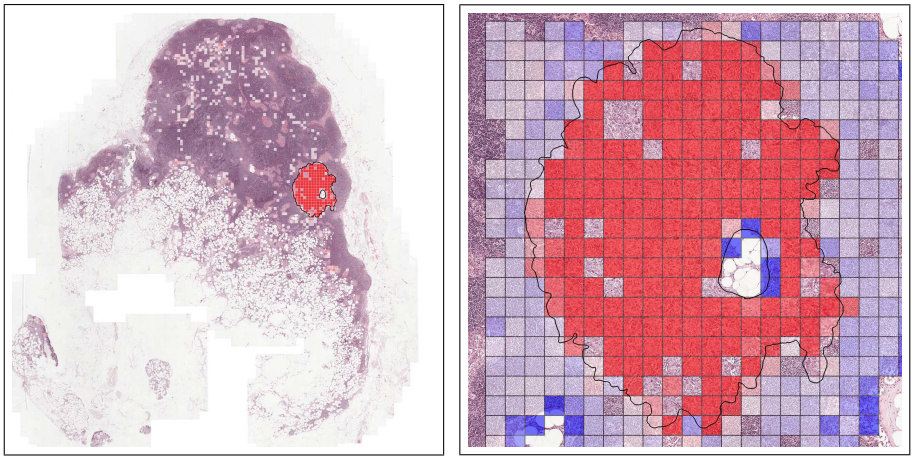


Figure 3: Visualization of metastasis detection on test image 27 of the `Camelyon-16` dataset using our proposed approach. **Left:** Full WSI at zoom level 6 with ground truth annotation of metastases shown via black border. Tiles with feature embeddings larger than 0 are classified as hits and color coded according to their magnitude, with red mapped to strong metastasis detection. **Right:** Detail of metastases at zoom level 2 overlaid with classification output of our proposed approach. Here, the output of all tested tiles are shown, with the feature embeddings mapped to a blue-white-red colormap, with blue and red mapped to strong healthy and metastasis detection, respectively. Tiles without color were not included when randomly selecting tiles for inference.

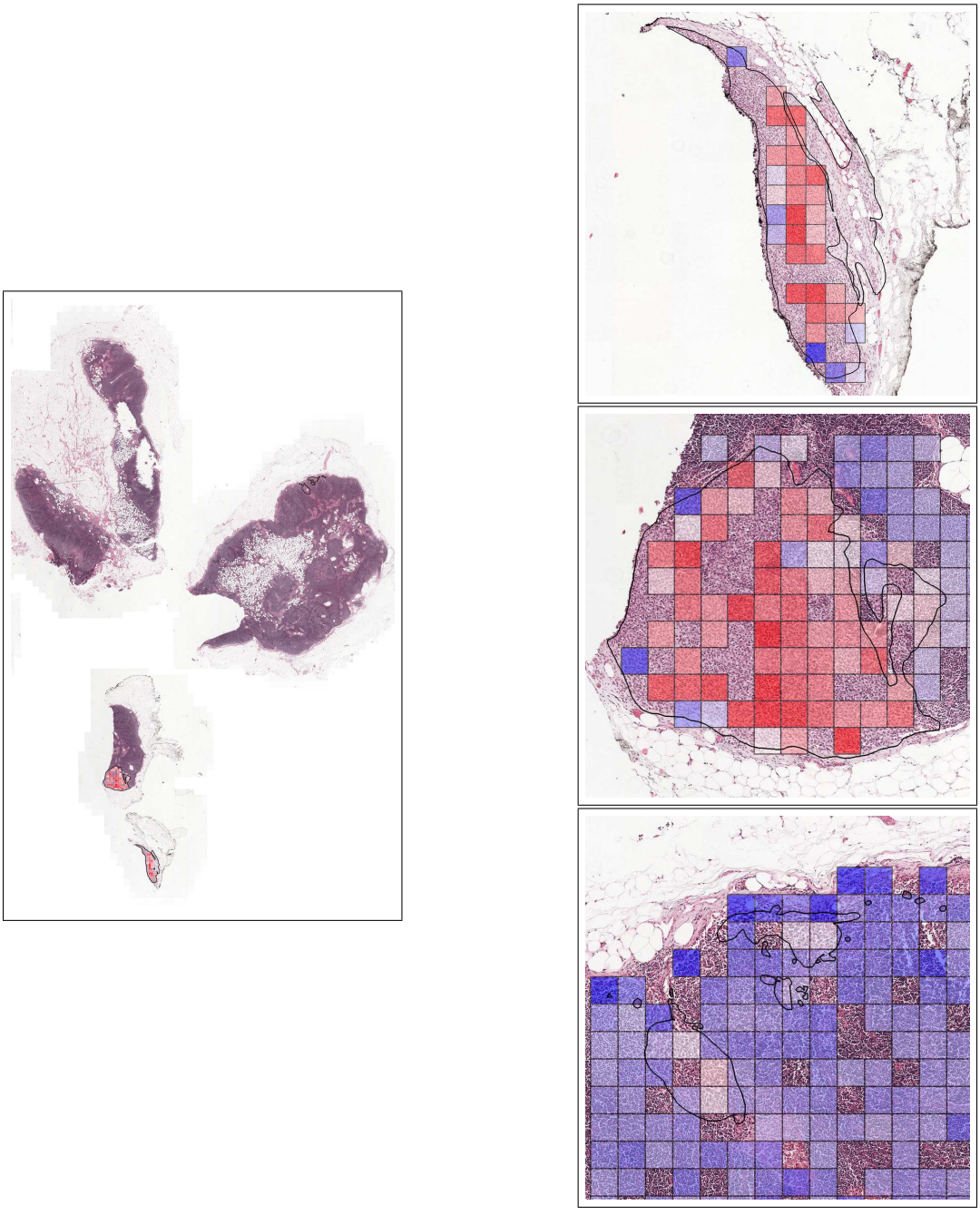


Figure 4: Visualization of metastasis detection on test image 2 of the `Camelyon-16` dataset using our proposed approach. **Left:** Full WSI at zoom level 6 with ground truth annotation of metastases shown via black border. Tiles with feature embeddings larger than 0 are classified as hits and color coded according to their magnitude, with red mapped to strong metastasis detection. **Right:** Detail of metastases at zoom level 2 overlaid with classification output of our proposed approach. Here, the output of all tested tiles are shown, with the feature embeddings mapped to a blue-white-red colormap, with blue and red mapped to strong healthy and metastasis detection, respectively. Tiles without color were not included when randomly selecting tiles for inference.

Detecting Cyber-bullying from Sparse Data and Inconsistent Labels

Sabina Tomkins **Lise Getoor** **Yunfei Chen** **Yi Zhang**
UC Santa Cruz
satomkin@ucsc.edu {getoor,ychen,yiz}@soe.ucsc.edu

Abstract

Cyber-bullying threatens the emotional, psychological and even physical health of up to 72% [6] of youth. A first step in combating this problem is to automatically detect incidents of cyber-bullying. Detecting incidents of cyber-bullying in text is typically cast as a binary classification problem and requires laborious manual labeling. Unfortunately this hand labeling poses two issues: it is expensive and prone to inconsistency. In this work we address both of these issues. We propose a linguistic model which uses domain knowledge to drastically reduce the number of parameters compared to standard bag-of-words approaches and is thus more appropriate for learning from limited labeled data. Rather than discard inconsistent labels we evaluate several methods for learning from them, demonstrating that incorporating uncertainty allows for better generalization.

1 Introduction

Bullying has long presented physical, emotional, and psychological risks to children, youth, and adults nationwide. As such there is an extensive body of knowledge aimed at understanding and preventing bullying. Far less is known about the newest form of interpersonal aggression: cyber-bullying. Cyber-bullying occurs in an electronic environment [9], from online forums, to social media platforms such as Twitter and Facebook. As it can occur at any time or location, cyber-bullying poses new risks, while also influencing well-being in the classroom [9, 6, 7]. It also introduces new questions of governance and enforcement, as it is less clear in an online environment who can and should police harmful behavior. It is critical to understand cyber-bullying as a new kind of human behavior, and as an extension of bullying which happens in the physical world.

One necessary first step in understanding and preventing cyber-bullying is to detect it, and a particular goal is to be able to automatically flag potentially harmful *social media* messages. However, social media messages introduce unique problems. As they are unusually short, and rife with misspellings and slang, when treated with traditional text cleaning these messages are often stripped to only one or two words. This sparsity, coupled with the cost of obtaining high quality labels, makes them especially ill-suited for bag-of-word approaches which depend on sufficient training data to generalize well. Not only is labeled data costly, but it can be error-prone as annotators are generally third parties who are not directly involved with the incidents of cyber-bullying. Thus their labeling is subjective, and even labels with high inter-annotator agreement may be incorrect.

We propose a novel modeling framework which utilizes collective reasoning and domain knowledge to detect cyber-bullying. Rather than throwing out annotations with low inter-annotator agreement, we propose a set of probabilistic linguistic models which can directly incorporate uncertainty. To overcome the challenges with learning from low-quality tweets, we develop a linguistic model which uses domain knowledge. We demonstrate that models which use domain knowledge are better able to detect cyber-bullying in tweet messages than those that rely on n-grams. Furthermore, we show that

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

modeling uncertainty in the training data can improve the F-Measure of all models, demonstrating that a probabilistic approach is well suited for this domain.

2 Detecting Cyber-Bullying

The problem of determining if a message contains bullying content is typically formulated as a binary classification task, which takes as input a vector of observed variables $\boldsymbol{x}$ and outputs a vector of labels $\boldsymbol{y}$. Typically, $\boldsymbol{y}$ is treated as binary, however in this work we investigate the ability to learn from continuously valued labels representing the degree to which a message contains bullying content. To do so, we relax our labels into the $[0, 1]$ interval for training, and demarcate these training labels $\tilde{\boldsymbol{y}}$.

In this section we introduce three linguistic models with which to detect cyber-bullying. Furthermore we propose three methods for determining the labels used in training $\tilde{\boldsymbol{y}}$. We construct our models with Probabilistic Soft Logic, with which we can encode domain knowledge as well as dependencies between the target variables.

2.1 Probabilistic Soft Logic

We propose a probabilistic approach which can learn from noisy social media interactions, where text is typically short and feature vectors sparse. We define a joint probability distribution over bullying messages using a *hinge-loss* Markov random field (HL-MRF) [1]. HL-MRFs are a general class of conditional, continuous Markov random fields, which provide the advantage of highly-efficient inference while maintaining expressivity.

A HL-MRF describes the following conditional probability density function over vectors of observed, $\boldsymbol{x}$, and unobserved, $\boldsymbol{y}$, continuous random variables:

$$P(\boldsymbol{y}|\boldsymbol{x}) \propto exp\left(-\sum_{j=1}^{m} w_j \phi_j(\boldsymbol{y}, \boldsymbol{x})\right)$$

where ϕ_j is a *hinge-loss* potential, $\phi_j = \max\{l_j(\boldsymbol{x}, \boldsymbol{y}), 0\}^p, p \in \{1, 2\}$, l_j is a linear function of $\boldsymbol{x}$ and $\boldsymbol{y}$ and w_j is the positive weight associated with ϕ_j.

To specify a HL-MRF, we use the templating language Probabilistic Soft Logic (PSL). In PSL, domain knowledge is encoded as weighted rules that capture dependences between both the input and output variables. These rules translate into the weighted potential functions ϕ. For example in our model, we have a rule which says that if two messages are similar, and one contains bullying content, then the other one may be likely to as well. To express this rule we introduce the predicate Similar, which takes two messages as arguments and whose truth value is the similarity between those messages. Additionally, we introduce the predicate BullyingContent, which takes a message as an argument and whose truth value indicates bullying. Using these predicates, and a weight w_{sim}, we define our rule in PSL as follows:

$$w_{sim} : \text{Similar}(T_a, T_b) \wedge \text{BullyingContent}(T_a) \Rightarrow \text{BullyingContent}(T_b)$$

Together a predicate and its arguments form a logical atom; unlike in Boolean logic PSL atoms can assume soft truth values in $[0, 1]$, allowing us to express the uncertainty inherent in annotations. When supplied with data a PSL model defines a unique HL-MRF where atoms represent either observed ($\boldsymbol{x}$) or unobserved random variables ($\boldsymbol{y}$) in the probabilistic model. As MAP inference in a HL-MRF can be formulated as a convex problem, we can tractably and efficiently infer the values to $\boldsymbol{y}$ or $\tilde{\boldsymbol{y}}$ with the PSL software[1], using the alternating direction method of multipliers ADMM [2]. Next we demonstrate how PSL can be used to template linguistic models for cyber-bullying detection.

2.2 Linguistic Patterns of Cyber-Bullying

A question in developing linguistic models of cyber-bullying detection is how exactly to model the words which occur in text. Here we propose two approaches. The first is an *N-Grams* model, which learns the correlation between each n-gram and the bullying content of a message. We compare

[1] `http://psl.linqs.org`

this to the *Seed Phrases* model, which correlates phrases to bullying, and differentiates between subjects of attacks to detect harmful messages. As each of these models has potential weaknesses, we combine them into the *Seeds++* model, which contains all rules from both models to benefit from their respective strengths.

Learning from social media data poses its own unique challenges. Messages are typically shorter than many other natural language documents, and tweets are severely limited to be within 140 characters. Furthermore, the difficulty of extracting linguistic signal from short tweets is compounded by their proclivity to contain misspellings and slang.

Model 1: N-Grams The full *N-Grams* model consists of the rules in Table 1 and Table 2. To address class imbalance we introduce a prior in all models, $\neg$BullyTweet(T), which captures that the majority of messages do not contain bullying content. For each word we instantiate a weighted rule correlating the presence of this word within a tweet to whether or not it is a bullying tweet. By training these weights we learn which words indicate bullying content. Here we consider words to be unigrams and bigrams.

Model 2: N-Grams++ The *N-Grams++* model contains the rules in Table 1 through Table 3. To overcome the sparsity of the feature vectors in the N-Grams model we propose two advanced features: sentiment and a document embedding similarity. Sentiment is assessed at the document level and provides some signal even from otherwise information deficient tweets. As bullying messages can be highly charged, we model the valence of a tweet with SentiTweet(T), where "Senti" is either Negative, Positive, or Neutral. Additionally, we learn distributed representations of each tweet, allowing us to abate some issues of sparsity. By mapping tweets to an embedding space, using any common text embedding methods, we can encode the relationship that documents which are close to each other in that space should have similar labels. To do so we introduce the predicate Similar(T_i, T_j) whose truth value is the cosine similarity between the embeddings of T_i and T_j respectively, scaled to be in [0,1]. By modeling similarity we can explicitly express dependencies between our target variables, thus benefiting from collective inference.

The disadvantage of the *N-Grams* and *N-Grams++* models is that the weight relating each word to bullying must be tuned with training data, which is commonly sparse with some words appearing in only a few documents. This sparsity can make learning from a small corpus difficult, and can hamper generalization to unseen data. A model which exploits domain knowledge to reduce the number of free parameters may be better suited to this task.

w_{nb} : $\neg$BullyTweet(T)

Table 1: *Class Imbalance Prior*

w_i : HasWord$(T, w_i) \Rightarrow$ BullyTweet(T)

Table 2: *N-Grams*

w_p : SentiTweet$(T) \Rightarrow$ BullyTweet(T)
w_c : BullyTweet$(T_i) \wedge$ Similar(T_i, T_j) $\Rightarrow$ BullyTweet(T)

Table 3: *Sentiment and Document Similarity*

Model 3: Seeds++ To this end we propose *Seeds++*, consisting of the rules in Table 1, Table 3, and Table 4, which relates certain phrases to bullying. Van Hee et al. [15], found seven different categories of bullying messages. In our data we found evidence of three: name calling, threatening, and sexual remarks, as well as a fourth category we refer to as silencing. Each category contains a small set of phrases, for example the insult category contains the phrase, "fat", an example threat phrase is "hurt you", a sexual attack might contain the word "slut" and the silencing category contains the phrase "shut up".

Furthermore, we differentiate between attacks made at individuals and third parties with two rules which express if a message contains a direct second or indirect third person reference. For example, "you" is a second person reference, while "they" is a third person reference. The rule set for *Seeds* is specified in Table 4, though this model also contains the rules from Table 1.

Model 4: Combined Finally, we combine these rules into a complete model *Combined*, which consists of all of the rules in Table 1 through

w_n : Insult$(T) \Rightarrow$ BullyTweet(T)
w_t : Threat$(T) \Rightarrow$ BullyTweet(T)
w_x : Sexual$(T) \Rightarrow$ BullyTweet(T)
w_s : Silencing$(T) \Rightarrow$ BullyTweet(T)
w_{m2} : Direct$(T) \wedge$ Insult(T) $\Rightarrow$ BullyTweet(T)
w_{m3} : InDirect$(T) \wedge$ Insult(T) $\Rightarrow \neg$BullyTweet(T)

Table 4: *Seed Phrases*

Table 4. This model allows us to benefit from the strengths of all rules. All models are evaluated empirically in Section 3.

2.3 Uncertain Annotations

As it is difficult and costly to acquire high-quality annotations of textual data, we explore the possible benefits of learning from low certainty annotations. There are two possible forms of uncertainty in this setting, disagreement between annotators and the uncertainty of individual annotators. Here we have three annotators where each can label a tweet with a 0 (not bully), 1 (maybe bully) and 2 (bully).

Let a be the normalized value assigned by the three annotators, that is $\frac{1}{6}\sum_{i=1}^{3} a_i$, where a_i is the label of the i-th annotator, and $\tilde{y}$ be the final label used in training. We explore three methods for determining $\tilde{y}$. In the *Discrete* method we discard all labels without high inter-annotator agreement. That is, we round all labels such that if $a \geq \frac{2}{3}$, $\tilde{y} = 1$ and $a \leq \frac{1}{3}$, $\tilde{y} = 0$ and all $\frac{2}{3} > a > \frac{1}{3}$ are discarded. We also introduce an alternate method, *Soft*, where $\tilde{y} = a$, which allows us to use all of the information provided by the annotation. In the *Hybrid* method, where there is high inter-annotator agreement we treat the label as discrete, and set $\tilde{y}$ exactly as in the discrete method. Yet when all annotators are uncertain, or when all three disagree, such that $\frac{2}{3} > a > \frac{1}{3}$, we set $\tilde{y} = a$.

3 Empirical Evaluation

We investigate two questions in this section: which linguistic model can best learn from limited labeled data and which labeling method is most useful in model training. We analyze the performance of five models: *N-Grams*, and *N-Grams++* augmented with collective rules and sentiment features, a domain inspired model *Seeds++*, and *Combined* which integrates strengths from all models. Additionally we compare to a Support Vector Machine (SVM) [4] [2].To inspect the effect of uncertainty, we compare three methods of handling inconsistent labels: *Discrete*, *Soft* and *Hybrid*.

3.1 Data

Here we explore cyber-bullying on Twitter, a popular social media platform. We focus on young children and adolescents as they represent a particularly at-risk group [14] and collect tweets by or referencing users under the age of 18. A total of 1798 tweets were annotated by three students, who were asked to mark each message according to how they felt reading it, with 0, 1, and 2 indicating no, maybe and definitely bullying respectively. The Fliess inter-annotator agreement was .066.

The tweets were cleaned according to standard natural language processing practices. Stop words were removed, as were numbers, and non-English words. Words with repeat characters were trimmed, for example "haaappy" became "happy". Only those words which appeared in at least 30 documents were retained. Sentiment was assigned with the open-source python tool VADER [5].

To calculate the similarity between two tweets we compute the cosine similarity according to a trained Doc2Vec model [8]. There is a trade-off in document embedding models, where a small domain-specific corpus may not have enough content to properly learn the embedding space yet a publicly available corpus may not fully capture the nuances of a specific domain, such as exchanges among adolescents. For this reason we train a Doc2Vec model on our corpus using Gensim [12], but seed it with pre-trained word vectors[3]. To seed the model we used the openly available Glove [10] Word2Vecs, which were trained on Twitter, and are thus appropriate for this domain.

All models are trained using 5-fold cross validation on 90% of the data. In all folds we maintain a distribution of 30% bully tweets. The reported results are on the final held-out test set of 10% of the data. Here we compare five models, an SVM, and four PSL models: the *N-Grams*, *N-Grams++*, *Seeds++* and *Combined*. When detecting harmful incidents high recall is desirable, especially in situations where predictions are passed to humans for final inspection. However, we also value precision as it could be costly and dangerous to mislabel incidents of aggression. Hence, the F-Measure gives a balanced perspective of the trade-off between precision and recall and is used often in this task, and is more informative than accuracy when there is class imbalance in the data. To evaluate we round all soft predictions to 0 or 1.

[2]We used the implementation in the python package scikit-learn

[3]To initialize the Doc2Vec model with Word2Vecs we use: `https://github.com/jhlau/doc2vec`

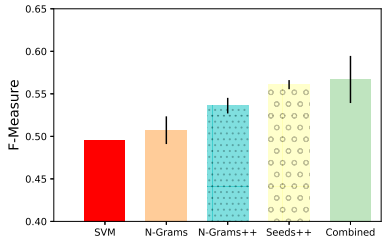


Figure 1: Adding collective rules improves the N-Grams model, while seed phrases are useful overall.

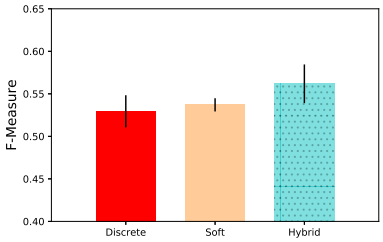


Figure 2: Uncertainty is helpful, with the hybrid labeling method being the most effective

3.2 Results

The PSL models all achieved a higher F-Measure than the SVM baseline. We see that the *Seeds++* model obtained better F-Measure than any of the *N-Grams* models. This is striking, as only a few seed words and phrases (43 total across all categories) were sufficient to obtain these results. Combining all models obtained the best overall F-Measure, with a statistically significantly higher precision recall and accuracy as well, suggesting that the models can adapt to different aspects of the data.

We also see that utilizing uncertainty in any form is beneficial. However a hybrid approach yields the best results. As annotations are often inconsistent, and annotators may not have high confidence in situations in which they have no personal knowledge, it is interesting to see that even uncertain information can be useful.

4 Related Work

There is a body of work on developing textual features for detecting cyber-bullying with linguistic classifiers [3, 17, 13, 15, 16]. Using TF-IDF features and contextual features Yin et al. [17] predict bullying with high accuracy. Also using context, Chen et al. [3] develop a novel framework which incorporates unique style features and structure.

Similar to our work Raisi and Huang [11] use a small seed vocabulary to indicate bullying events. They leverage participant roles to expand the set of candidate bullying terms, and better predict bullying. Their approach corroborates the idea that seed indicators can be successful in this task. Also in a similar vein to our work, Reynolds et al. [13] compare a rule based approach to a Support Vector Machine, and find that the rule-based model obtains higher accuracy. Finally, our work has been highly motivated by Van Hee et al. [15] who go beyond binary classification to label events as belonging to one of several textual categories pertaining to bullying conversations.

We build on existing work by building rich textual features to detect cyber-bullying. Unlike existing approaches we exploit a collective setting to leverage document similarity. Like [11] we employ a small set of seed words, which we develop from the categories of [15] for a domain inspired model.

5 Conclusion

Detecting cyber-bullying is a critical task in today's inter-connected world, where the virtual manifestations of relationships have strong bearings on the emotional health of all parties. Traditional supervised machine learning approaches, reliant on large amounts of labeled data may suffer in this domain, where high quality annotations are expensive, and feature vectors are particularly sparse. To address these issues we propose a linguistic model which uses domain knowledge to detect harmful messages, obtaining better F-Measure than N-Grams variants and a SVM baseline. We investigate the utility of uncertain annotations, discovering that incorporating soft labels can improve performance.

Acknowledgements

This work is supported by the National Science Foundation under Grant No. 1703281.

References

[1] Stephen H. Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *Journal of Machine Learning Research (JMLR)*, 2017. To appear.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 2011.

[3] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. Detecting offensive language in social media to protect adolescent online safety. SOCIALCOM-PASSAT, 2012.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20, 1995.

[5] Clayton J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *ICWSM*, 2014.

[6] Jaana Juvonen and Elisheva F. Gross. Extending the school grounds? – Bullying experiences in cyberspace. *Journal of School Health*, 2008.

[7] Catarina Katzer, Detlef Fetchenhauer, and Frank Belschak. Cyberbullying: Who Are the Victims? *Journal of Media Psychology*, 2009.

[8] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, 2014.

[9] Qing Li. New bottle but old wine: A research of cyberbullying in schools. *Computers in Human Behavior*, 2007.

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[11] Elaheh Raisi and Bert Huang. Cyberbullying detection with weakly supervised machine learning. *ASONAM*, 2017.

[12] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010.

[13] Kelly Reynolds, April Kontostathis, and Lynne Edwards. Using machine learning to detect cyberbullying. ICMLA, 2011.

[14] Robert Slonje, Peter K Smith, and Ann FriséN. The nature of cyberbullying, and strategies for prevention. *Computers in human behavior*, 2013.

[15] Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Veronique Hoste. Detection and fine-grained classification of cyberbullying events. In *Proceedings of Recent Advances in Natural Language Processing, Proceedings*, 2015.

[16] Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. Learning from bullying traces in social media. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT, 2012.

[17] Dawei Yin, Brian D. Davison, Zhenzhen Xue, Liangjie Hong, April Kontostathis, and Lynne Edwards. Detection of Harassment on Web 2.0. In *Content Analysis in the Web 2.0 (CAW2.0)*, 2009.

Hyperparameter Selection under Localized Label Noise via Corrupt Validation

David I. Inouye[*], **Pradeep Ravikumar**
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{dinouye, pradeep}@cs.cmu.edu

Pradipto Das, **Ankur Datta**
Rakuten Institute of Technology
Rakuten USA
Boston, MA 02110
{pradipto.das, ankur.datta}@rakuten.com

Abstract

Existing research on label noise often focuses on simple uniform or class-conditional noise. However, in many real-world settings, label noise is often somewhat systematic rather than completely random. Thus, we first propose a novel label noise model called Localized Label Noise (LLN) that corrupts the labels in small local regions and is significantly more general than either uniform or class-conditional label noise. LLN is based on a k-nearest neighbors corruption algorithm that corrupts all neighbors to the same wrong label and reduces to a class-conditional label noise if $k = 1$. Given this more powerful model of label noise, we propose an empirical hyperparameter selection method under LLN that selects better hyperparameters than traditional selection strategies, such as cross validation, by synthetically corrupting the training labels while leaving the test labels unmodified. This method can provide an approximate and more robust validation for hyperparameter selection. We design several label corruption experiments on both synthetic and real-world data to demonstrate that our proposed hyperparameter selection yields better estimates than standard methods.

1 Introduction

Most previous work on label noise has focused on uniform [1–4] or class-conditional label noise [4–10].[1] Uniform label noise assumes that the probability of corruption is independent of both the true label while class-conditional label noise allows the corruption probability to depend on the true label. However, in many real-world settings, the label noise is not uniformly random but rather *locally systematic*. For example, in an e-commerce setting, suppose a merchant sets the default category for their products to be "Computers" but fails (for lack of effort or by accident) to change the category to "Media" for all their DVDs; this error would cause *all* of their DVDs to be labeled incorrectly—a local but systematic label noise. Shen et al. [16] suggest that even when merchants manually select the category for every item, they make approximately 15% error largely because of category ambiguity. Or as another example, hand-crafted rules may be used to label data in large batches so as to reduce labeling effort; however, many simple hand-crafted rules may introduce systematic errors in the labels. As a concrete example, suppose that any product with "ring" in the title is classified as jewelry; while this rule may provide reasonable accuracy, hardware items like "sealant ring" would all be misclassified into jewelry. In e-commerce, this label corruption problem can cause significant revenue loss because users are likely to become confused and navigate away from the website [16].

[*] A majority of this work was done during a summer internship at Rakuten Institute of Technology, Boston.

[1] Some recent work has considered cases of feature-dependent label noise [8, 11–13] or even adversarial label noise [14, 15] but is either primarily theoretical [8, 12, 14] or model-specific [8, 11, 13–15], where our proposed label noise model is empirically-driven and model-agnostic as will be described in future sections.

Presented as workshop paper at the 31st Conference on Neural Information Processing Systems (NIPS 2017).

Figure 1: Quadruple moon toy dataset before corruption (left) and after localized label corruption (right), which yields systematic noise in local regions.

In light of this mismatch between uniform label noise and locally systematic label noise, we propose a corruption algorithm based on k-nearest neighbors that generates locally systematic label noise. Fig. 1 shows a toy dataset with true clean labels on the left and a corrupted version of the dataset on the right. Note that small localized regions of the data are corrupted together, e.g. five nearby points in one class are all corrupted to another class. This simple corruption algorithm implies a powerful generative model for localized label noise that we develop in the following sections. The harder question then is that, given such localized label corruption phenomenon, how might we train models to handle this label noise? Frenay and Verleysen [4] present three main approaches to handling label noise: (1) robust methods that are naturally robust to label noise but do not explicitly model label noise, (2) data cleansing methods that attempt to identify and remove corrupt labels before classification, and (3) noise-tolerant methods that jointly model the label noise and classification model. Our proposed method most closely follows approach (1) but can be applied to *any* classification model that has a complexity hyperparameter such as k in KNN, *tree depth* in decision tree, or *regularization parameters* in SVMs. In contrast to our proposed model-agnostic method, methods summarized in [4, Sec. V] are tied to certain loss functions (e.g. 0-1 loss), models (e.g. linear models) or algorithms (e.g. ensemble methods). From another viewpoint, we propose a method for estimating *relative* generalization error for hyperparameter selection that is less biased than traditional methods such as cross validation. In fact, Frenay and Verleysen [4] suggest that choosing hyperparameters under label noise is still an open problem. We make an attempt at resolving this important gap in the literature.

We summarize our two primary contributions as follows: (1) A novel model for systematic localized label noise including an empirical algorithm and probabilistic generative model, and (2) a novel label-noise-aware validation method that can be applied to make *any* classification method more robust to label noise. We believe these two contributions are complementary but can be used independently of each other and open up new research directions in both areas.

2 Localized Label Noise

We first present our localized corruption algorithm based on k-nearest neighbors. We then describe the probabilistic generative label noise model *implied* by the algorithm. Throughout this paper, we assume that we there exist n i.i.d. samples of a d-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^d$ and corresponding true clean but *hidden* label $y \in \{1, 2, \dots, J\}$; $X \in \mathbb{R}^{n \times d}$ and $\boldsymbol{y} \in \{1, 2, \dots, J\}^n$ denote all n samples. We assume that the training algorithm only observes a *corrupted* version of the label denoted $\boldsymbol{y}_c \equiv c(\boldsymbol{y})$ rather than the true label $\boldsymbol{y}$.

Algorithm We describe a simple base corruption algorithm to build intuition, and then we will extend it to the full algorithm. Let us denote the total corruption rate or amount as π, e.g. $\pi = 0.2$ corresponds to 20% of the labels being corrupted. First, we initialize corrupt labels to the clean labels $\boldsymbol{y}_c = \boldsymbol{y}$ and corruption sets $\mathcal{C}^{(j)} = \emptyset$. Then, for all class values $j = 1, \cdots, J$ and while $|\mathcal{C}^{(j)}|/n < \pi$:

1. Sample data point $\tilde{\boldsymbol{x}}$ uniformly from the set $X^{(j)} \equiv \{\boldsymbol{x}_i \colon y_i = j\}$
2. Compute the k-size neighborhood of $\tilde{\boldsymbol{x}}$ in the set $X^{(j)}$ denoted $\mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})$, which includes $\tilde{\boldsymbol{x}}$
3. Sample corrupted label $\tilde{y}$ uniformly from all labels except the j-th label
4. Corrupt all labels in neighborhood $\mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})$: $[\boldsymbol{y}_c]_{\mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})} \leftarrow \tilde{y}$
5. Update corrupted set: $\mathcal{C}^{(j)} \leftarrow \mathcal{C}^{(j)} \cup \mathcal{N}^{(j)}(\tilde{\boldsymbol{x}})$

Note that the user must specify k, which is not necessarily intuitive to estimate. Thus, we reparameterize this process by normalizing by the number of observations, i.e. $\mu = k/n$, so that the user only needs to specify an approximate percentage of observations that tend to be corrupted together. For example, a data vendor may estimate their overall corruption rate as $\pi = 15\%$ with local regions of size $\mu = 0.5\%$ may be corrupted together. This parameterization avoids having to manually scale

the algorithm based on the number of observations. Additionally, we can allow μ, a proportion parameter, to follow the Beta distribution with standard deviation σ. Thus our algorithm only requires three relatively intuitive parameters: total corruption percentage π (e.g. 10%), mean local corruption percentage μ (e.g. 1%), and standard deviation of local corruption percentage σ (e.g. 0.5%).[2]

Generative Model Building on the intuitions from the simple algorithm above, we now derive the implied population-level probabilistic generative model.[3] We assume the distribution of $\boldsymbol{x}$ given y is fixed but unknown and is denoted by θ. We also define the following epsilon ball given a point and a probability to cover: $\mathcal{B}(\tilde{\boldsymbol{x}}, \delta) = \mathcal{B}_\epsilon(\tilde{\boldsymbol{x}}) = \{\boldsymbol{x} \colon \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| \leq \epsilon\}$ such that $\mathbb{P}(\boldsymbol{x} \in \mathcal{B}_\epsilon(\tilde{\boldsymbol{x}})) = \delta$. Essentially, this is the unit ball in the feature space centered around $\tilde{\boldsymbol{x}}$ whose probability is exactly δ. With this notation, we define the localized label noise model via a generative graphical model:

1. For $j = 1, \ldots, J$ and for $\ell = 1, 2, \ldots$
   (a) Sample $\tilde{\boldsymbol{x}}_{j\ell} \in \mathbb{R}^d$ from the conditional data distribution given $y = j$
   (b) Sample labels $\tilde{y}_{j\ell}$ from a uniform distribution with all labels *except* j
   (c) Sample $\hat{\delta}$ from a Beta distribution parameterized by mean μ and standard deviation σ
   (d) Set $\delta_{j\ell} = \min(\hat{\delta}, \pi - \mathbb{P}(\boldsymbol{x} \in \mathcal{C}^{(j)} \mid y))$ to ensure the corruption does not exceed π
   (e) Update corruption set by new corrupt ball: $\mathcal{C}^{(j)} \to \mathcal{C}^{(j)} \cup \mathcal{B}(\tilde{\boldsymbol{x}}_{j\ell}, \delta_{j\ell})$
   (f) If $\mathbb{P}(\boldsymbol{x} \in \mathcal{C}^{(j)}) = \pi$, break inner loop, set the number of corruption points $m_j(\pi) \leftarrow \ell$
2. For $i = 1, \ldots, n$
   (a) Sample $y \in \{1, \cdots, J\}$ from prior class distribution
   (b) Sample $\boldsymbol{x} \in \mathbb{R}^d$ given the conditional distribution θ_y
   (c) Set y_c as below, where $m \equiv m_y(\pi)$ is the number of corruption points

$$y_c = \begin{cases} \tilde{y}_{y1} & , \boldsymbol{x} \in \mathcal{B}(\tilde{\boldsymbol{x}}_{y1}, \delta_{y1}) \\ \vdots & , \quad \vdots \\ \tilde{y}_{ym} & , \boldsymbol{x} \in \mathcal{B}(\tilde{\boldsymbol{x}}_{ym}, \delta_{ym}) \text{ and } \boldsymbol{x} \notin \cup_{a=1}^{m-1} \mathcal{B}(\tilde{\boldsymbol{x}}_{ya}, \delta_{ya}) \\ y & , \text{otherwise (i.e. uncorrupted)} \end{cases} \tag{1}$$

In Fig. 2, we compare our generative graphical model with the class-conditional and instance-dependent label noise models.[4] Note that while the label corruption for each instance is independent in previous models, our label noise model induces global dependencies between labels.

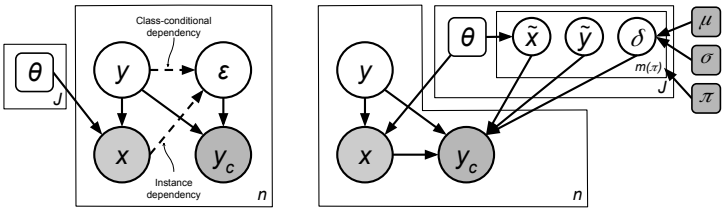


Figure 2: The graphical models of class-conditional label noise (left) and localized label noise (LLN) (right) show that LLN is a significantly more complex label noise model that shares global parameters across samples instead of being independent for each sample. π denotes the total percentage of corruption. μ and σ are the mean and standard deviation of a Beta distribution. Note how δ essentially becomes the percentage to corrupt together.

3 Corrupt Validation for Hyperparameter Selection

We will now consider hyperparameter evaluation methods and propose our method based on synthetically corrupting the training labels while leaving the test labels unmodified. Let $c_1(y \mid \boldsymbol{x}) \colon \{1, \cdots, J\} \times \mathbb{R}^d \to \{1, \cdots, J\}$ and $c_2(\cdot)$ be label corruption functions. In a real world scenario, $c_1(\cdot)$ is the original *unknown* (or hidden) corruption and $c_2(\cdot)$ is some sort of *known*

[2]Note that μ is the *percentage* that is corrupted simultaneously, not the volume in the feature space; thus, the local region will have larger volume in a low density area than a local region in a high density area.

[3]This generative model is primarily to build intuitions about the noise algorithm because we do not attempt to estimate the generative model; however, noise model estimation would be an interesting research direction.

[4]For clarity, we hide the implicit dependency of $\delta_{j\ell}$ on $\tilde{\boldsymbol{x}}_{ja}$ and δ_{ja} for $a < \ell$ via the auxiliary set $\mathcal{C}^{(j)}$.

synthetic corruption—in our case localized label corruption defined previously. We present two oracle validation methods that require oracle access to the true hidden labels, and two corresponding empirical validation methods that can actually be estimated from the observed data. We describe these validation methods below and give a visual summary in Fig. 3.

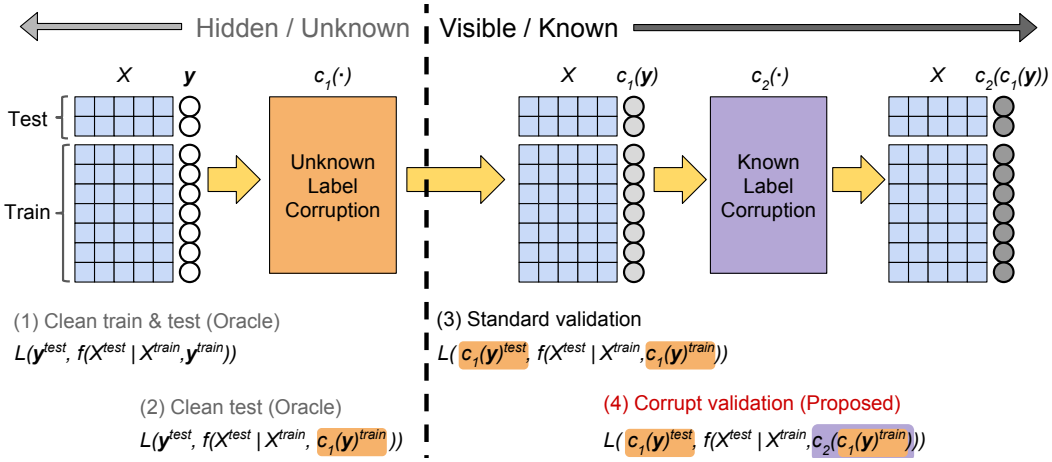


Figure 3: (1) is an oracle (access to hidden clean labels $\boldsymbol{y}$) method that assumes clean labels for both train and test. (2) is an oracle method that assumes clean labels only for testing and is thus the best hyperparameter estimator given only corrupted training labels. (3) is the traditional held-out validation that naively ignores label corruption. (4) is our proposed corrupt validation method that we suggest tracks the trends of method (2) more closely than (3), and thus selects better hyperparameters assuming the algorithm only has access to corrupt labels. $f(a \mid b)$ corresponds to a classifier prediction of a given training data b and L is a loss function.

(1) Clean train & test (Oracle, train: $\boldsymbol{y}$, test: $\boldsymbol{y}$) Validation method if full oracle access to true hidden labels is allowed for both for train and test. This is the classic academic setting where we assume there is no label corruption. **(2) Clean test (Oracle, train: $c_1(\boldsymbol{y})$, test: $\boldsymbol{y}$)** Validation method that only requires oracle access to the true hidden labels for the testing or evaluation phase. This method is the best that any validation model could do if the model can *only train on corrupt labels*. The difference between (1) and (2) is the performance penalty for having label noise. **(3) Standard validation (train: $c_1(\boldsymbol{y})$, test: $c_1(\boldsymbol{y})$)** Traditional validation method that naively ignores the label corruption. The underlying assumption is that this method mimics method (1). However, as we will show, approximating method (1) is actually not the goal because the model must be trained on corrupted labels rather than on the true (but hidden) labels. **(4) Corrupt validation (train: $c_2(c_1(\boldsymbol{y}))$, test: $c_1(\boldsymbol{y})$)** Corrupt validation method where the training labels are synthetically corrupted before training but the labels are left unmodified when evaluating the model. This validation attempts to mimic method (2), and can be seen as a *relative* proxy validation for (2).[5]

4 Experiments

We empirically evaluate our proposed corrupt validation on two datasets: a synthetic toy dataset with four moon-shaped classes from the introduction ($d = 2, n = 1000, J = 4$, see Fig. 1) and a real-world sample of automatically-curated[6] product catalog titles from a leading e-commerce company ($d = 82k, n = 1.2M, J = 29$, see Fig. 1). In these experiments, we set c_1 and c_2 to be the same localized label noise model using the simple empirical algorithm with $\pi = 20\%, \mu = 2\%, \sigma = 1\%$.[7] We implemented the methods and experiments within the scikit-learn Python framework [17]. For the toy dataset, we computed nested cross validation with three replications whereas we used a simple nested train/test validation for the larger dataset of product catalogs. For space reasons, we

[5] While this provides a useful relative model selection criteria, this method does not necessarily provide useful *absolute* performance estimates—estimating (2) correctly without true labels is still a significant open question.

[6] The curation has been done using manually crafted rules that yield high precision.

[7] An important open question is the sensitivity of the method to misspecification of c_2 because c_1 is usually hidden or unknown.

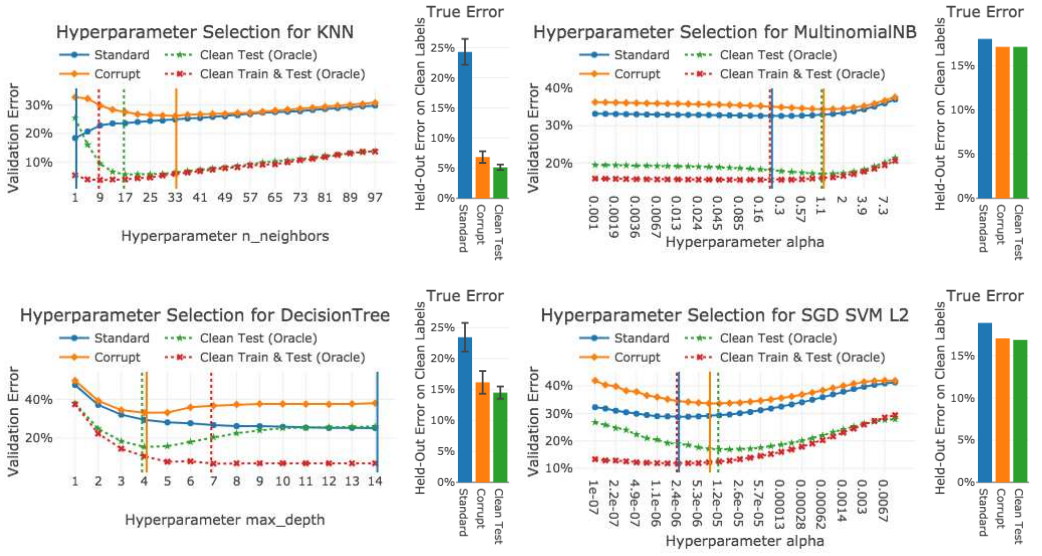


Figure 4: Average hyperparameter selection curves for various datasets and classification algorithms (line graphs, vertical lines are at minimum) and the true held-out test performance on clean labels of the best selected models (bar charts). (Left) Toy quadruple moons dataset. (Right) Product catalog listings from a leading e-commerce company. Our proposed corrupt validation (orange bar) performs better in terms of true held-out error than standard validation (blue bar)—where the gap is larger for non-parametric models like KNN and decision trees. The oracle clean test validation (green bar) is the best hyperparameter selection method if the final model can only be trained on corrupt labels. Note that the validation curves are only used to select the hyperparameter with lowest error; thus, only the *relative* error w.r.t. other hyperparameter values for each method is important. The hyperparameters of the models for the toy dataset are self explanatory. Hyperparameter α for MultinomialNB is the smoothing factor and α for SGD SVM is the L_2 regularization constant.

only give results for KNN and decision trees for the toy dataset, and multinomial naive Bayes and L_2-regularized SVM using SGD for the product catalog dataset as can be seen in Fig. 4.

Across all the experiments on both datasets, the corrupt validation follows the general trends of the oracle clean test validation method (method (2) from previous section) and thus selects hyperparameters that give better *true* generalization performance on the clean data (the bar charts in Fig. 4). While the traditional validation approach does track with the oracle validation on clean test and train data, this is suboptimal because the model can only be trained on corrupt labels—thus it is relative performance estimates are overly optimistic. Intuitively, the traditional method begins to fit the noisy labels when the model has high complexity whereas corrupt validation avoids overfitting the noisy labels. Also, note that the non-parametric models KNN and decision tree are more likely to overfit the label noise than the multinomial naive Bayes or linear SVM, which may be one reason linear models are often used in the real-world situations. However, our corrupt validation method enables the use of even non-parametric models in the presence of significant localized label noise.

5 Conclusion

We proposed a novel localized label noise model that is significantly stronger but often more realistic than the most common uniform or class-conditional models. We then developed a hyperparameter selection methodology that can be applied to any classification model to make it more robust to localized noise. We think this opens up many new and interesting research directions both in terms of label noise models and in terms of model validation under label noise. For example, how sensitive is our hyperparameter selection method to label noise misspecification? What are other label noise models could be used in the corrupt validation scheme that might better reflect real label noise? Can this validation methodology be used to identify the most noisy labels? In general, we believe this work is an important first step in understanding and handling more complex label noise in practice.

Acknowledgments

A majority of this work was completed by D.I. during a summer internship at Rakuten Institute of Technology, Boston. P.R. and D.I. acknowledge the support of NSF via IIS-1149803.

References

[1] Borut Sluban and Nada Lavra. Relating ensemble diversity and performance : A study in class noise detection. *Neurocomputing*, 160:120–131, 2015.

[2] Fabien Rico, Fabrice Muhlenbach, Djamel A Zighed, and Stéphane Lallich. Comparison of two topological approaches for dealing with noisy labeling. *Neurocomputing*, 160:3–17, 2015.

[3] Yiming Qian, Minglun Gong, and Li Cheng. Stocs: An efficient self-tuning multiclass classification approach. pages 291–306. Springer, Cham, 2015.

[4] Benoit Frenay and Michel Verleysen. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

[5] Jakramate Bootkrajang and Ata Kab. Label-noise robust logistic regression and its applications. In *ECML PKDD*, pages 143–158, 2012.

[6] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *NIPS*, 2013.

[7] S Liu, D Maljovec, B Wang, P Bremer, and V Pascucci. Visualizing high-dimensional data : Advances in the past decade. *Eurographics Conference on Visualization (EuroVis)*, 2015.

[8] Aritra Ghosh, Naresh Manwani, and P S Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[9] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694v2*, 2017.

[10] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: a loss correction approach. In *CVPR*, 2017.

[11] Raj S Chhikara and Jim McKeon. Linear discriminant analysis with misallocation in training samples. *Journal of the American Statistical Association*, 79(388):899–906, 1984.

[12] Aditya Krishna Menon, Brendan Van Rooyen, and Nagarajan Natarajan. Learning from binary labels with instance-dependent corruption. *arXiv preprint arXiv:1605.00751v2*, 2016.

[13] Jakramate Bootkrajang. A generalised label noise model for classification in the presence of annotation errors. *Neurocomputing*, 192:61–71, jun 2016.

[14] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.

[15] Nenad Tomasev and Krisztian Buza. Hubness-aware knn classification of high-dimensional data in presence of label noise. *Neurocomputing*, 160:157–172, 2015.

[16] Dan Shen, Jean-David Ruvini, and Badrul Sarwar. Large-scale item categorization for e-commerce. *Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, (492):595, 2012.

[17] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Structured Prediction with Adversarial Constraint Learning

Hongyu Ren
Peking University
rhy@pku.edu.cn

Russell Stewart
Stanford University
stewartr@cs.stanford.edu

Jiaming Song
Stanford University
tsong@cs.stanford.edu

Volodymyr Kuleshov
Stanford University
kuleshov@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

Constraint learning is a recently proposed form of weak supervision which attempts to reduce the labeling burden by having users specify general properties that hold over the output space (e.g. physical laws). However, specifying constraints can be difficult and may require extensive domain expertise. In this paper, we introduce an adversarial constraint learning framework in which invariants are automatically extracted from data. In this framework, users only need to provide a black-box simulator that generates valid system outputs; at training time, we constrain the model to produce outputs that cannot be distinguished from simulated samples by a learned discriminator. Further providing our framework with a small number of labeled examples gives rise to a new semi-supervised structured prediction method; we evaluate this method on multiple tasks — object tracking and pose estimation, and we find that our framework achieves high accuracy with only a small amount of labels, and no labels at all in some cases.

1 Introduction

Large labeled datasets are key component in many machine learning applications [1–3], but collecting them can be expensive. Constraint learning is a recently proposed form of weak supervision which aims to reduce cost of collecting labels by supervising algorithms through general properties that hold over the output space [4, 5]. Examples of such properties include logical rules [6–8], physical laws [5], or anatomical properties of the human body. Unlike labels, which only apply to their corresponding inputs, properties used in a constraint learning approach are specified once for the entire dataset, providing an opportunity for more cost-effective supervision [9, 5].

However, describing the high level invariants of a dataset may also require a non-trivial amount of effort. First, designing explicit constraints requires strong domain expertise. Second, in the case of high dimensional outputs, encoding the constraints using simple formulas is hard. For example, it is difficult to constrain a pedestrian detector with formulas that describe the shape of walking person. Third, constraints may change across tasks; designing new constraints for new tasks may not scale in many practical applications.

In this paper, we propose an *implicit* approach to constraint learning, in which invariants are automatically learned from a small set of representative output samples (see Figure 1). These samples do *not* need to be tied to corresponding inputs (as in traditional supervised and semi-supervised learning) and may come from a black-box simulator that abstracts away physics-based formulas, examples of outputs collected by humans or from standard datasets used in supervised learning.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

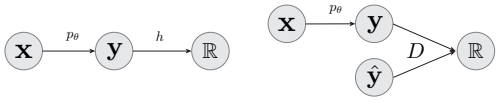


Figure 1: Constraint learning allows us to learn a probabilistic model $p_\theta(\mathbf{y}|\mathbf{x})$ without direct labels by specifying properties h that holds over the output space. In prior work (left), h is defined as a formula describing known invariants. In this paper (right), we propose to instead learn h through a discriminator network D that discriminates $\mathbf{y}$ (provided by $p_\theta(\mathbf{y}|\mathbf{x})$) from $\hat{\mathbf{y}}$ (provided by an additional source unrelated to $\mathbf{x}$, such as a simulator).

Inspired by recent advances in generative models, we capture the distribution of outputs using an approach based on adversarial learning [10]. Specifically, we train two distinct learners: a primary model for the task at hand and an auxiliary classification algorithm called a discriminator. During training, we constrain the main model such that its outputs cannot be distinguished by the discriminator from true output samples, thus forcing it to capture the structure of the output space. This approach forms a novel adversarial framework for performing weak supervision with learned constraints. Our framework turns into semi-supervised learning, when given some labeled data. Experimental results demonstrate that this method performs well on a variety of structured prediction problems, outperforming natural baselines with very few labeled inputs.

2 Background

2.1 Structured Prediction

In this paper, we focus on structured prediction problems, in which the outputs $\mathbf{y} \in \mathcal{Y}$ can be a complex object such as a vector, a tree, or a graph [11]. We capture the distribution of $\mathbf{y}$ using a conditional probabilistic model $p_\theta(\mathbf{y}|\mathbf{x})$ parameterized by $\theta \in \Theta$. A model $p_\theta(\mathbf{y}|\mathbf{x})$ maps each input $\mathbf{x} \in \mathcal{X}$ to the corresponding output distribution $p(\mathbf{y}) \in \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ denotes all the probability distributions over $\mathcal{Y}$. For example, we may take $p_\theta(\mathbf{y}|\mathbf{x})$ to be a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_\theta(\mathbf{x}), \boldsymbol{\Sigma}_\theta(\mathbf{x}))$ with mean $\boldsymbol{\mu}_\theta(\mathbf{x})$ and variance $\boldsymbol{\Sigma}_\theta(\mathbf{x})$.

A standard approach to learning $p_\theta(\mathbf{y}|\mathbf{x})$ (or p_θ) is to solve an optimization problem of the form

$$\theta^* = \arg\min_{\theta \in \Theta} \sum_{i=1}^{n} l(p_\theta(\mathbf{y}|\mathbf{x}_i), \mathbf{y}_i) + R(p_\theta) \tag{1}$$

over a labeled dataset $\mathcal{D}_L = \{(\mathbf{x}_1, \mathbf{y}_1), \cdots, (\mathbf{x}_n, \mathbf{y}_n)\}$. A typical supervised learning objective is comprised of a loss function $l : \mathcal{P}(\mathcal{Y}) \times \mathcal{Y} \to \mathbb{R}$ and a regularization term $R : \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$ that encourages non-degenerate solutions or solutions that incorporate prior knowledge [5].

2.2 Constraint-Based Learning

Let $\mathcal{D}_U = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$ be an unlabeled dataset of inputs, without their corresponding label. Formally, constraints can be specified via a function $h : \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$, which penalizes conditional probabilistic models $p_\theta(\mathbf{y}|\mathbf{x})$ that are not consistent with known high-level structure of the output space. Learning from constraints proceeds by optimizing the following objective.

$$\hat{\theta}^* = \arg\min_{\theta \in \Theta} \sum_{i=1}^{m} h(p_\theta(\mathbf{y}|\mathbf{x}_i)) + R(p_\theta) \tag{2}$$

By solving this optimization problem, we look for a probabilistic model parameterized by $\hat{\theta}^*$ that is likely a priori (through the $R(p_\theta)$ term), and satisfies known constraints when applied to the unlabeled dataset $\mathcal{D}_U$, such as physical laws. Note that the constraint h is data-dependent, although it does not require explicit labels. For example, in free fall object tracking, we could ask that the predictions on $\mathcal{D}_U$ over time form a parabola [5] and h measures how the output distribution from p_θ deviates from the equations. The regularization term is used to avoid overly complex and/or degenerate solutions, and may include $L1$, $L2$, and entropy regularization.

2.3 Adversarial Training and Implicit Probabilistic Models

Recent learning methods based on adversarial training and implicit probabilistic models play a key role in our approach [12]. Implicit probabilistic models are defined as the result of a stochastic sampling procedure, rather than through an explicitly defined likelihood function. Prominent examples are generative adversarial networks (GAN), where samples $G(\mathbf{z})$ are obtained by transforming some Gaussian noise $\mathbf{z} \sim \mathcal{N}(0, I)$ through a neural network G, called the generator.

In this work, we are interested in placing constraints on a probability distribution over the *output space* Y. We define this distribution implicitly by the following sampling procedure,

$$\mathbf{x} \sim p_d(\mathbf{x}) \quad , \quad \mathbf{y} \sim p_\theta(\mathbf{y}|\mathbf{x}) \tag{3}$$

where $p_d(\mathbf{x})$ is the data distribution over the input space $\mathcal{X}$, and $p_\theta(\mathbf{y}|\mathbf{x})$ is a conditional distribution of outputs given inputs. The above procedure corresponds to sampling from the marginal distribution over $\mathcal{Y}$, $p_\theta(\mathbf{y}) = \int p_\theta(\mathbf{y}|\mathbf{x})p_d(\mathbf{x})d\mathbf{x}$. However, evaluating the marginal likelihood $p_\theta(\mathbf{y})$ exactly is typically intractable due to the integration over $p_d(\mathbf{x})$.

Given label samples $\mathcal{D}_S = \{\mathbf{y}_1, \cdots, \mathbf{y}_k\}$, implicit generative models can be trained using likelihood-free methods that define a distance metric between distributions alternative to KL divergence. GANs [10] are trained using an approximation of the Jensen-Shannon divergence (JSD) with the following minimax objective, which can be optimized by stochastic gradient descent.

$$\min_G \max_D \mathbb{E}_{\mathbf{y}\sim p_s(\mathbf{y})}[\log D(\mathbf{y})] + \mathbb{E}_{\mathbf{y}\sim p_\theta(\mathbf{y}|\mathbf{x}),\mathbf{x}\sim p_d(\mathbf{x})}[\log(1 - D(\mathbf{y}))]$$

Other metrics can also be used as an objective for training implicit probabilistic models, such as maximum mean discrepancy (MMD) [13] or Earth Mover's distance (EMD) [14, 15].

3 Adversarial Constraint Learning

The process of describing high level constraints can be time-consuming and may require significant domain expertise. In the sciences, discovering general invariants is often a data-driven approach, for example, physical laws are often discovered by validating hypotheses with experimental results. Motivated by this idea, we propose to learn constraints from a set of representative output samples.

Learning Constraints from Data. Suppose that we are given a small number of labels $\mathcal{D}_S$ (not necessarily associated with input data), or a black-box mechanism/simulator for generating such labels, i.e., to sample from the empirical label distribution $p_s(\mathbf{y})$. We formulate the task of learning a constraint loss h from these label samples using the framework of generative adversarial learning [10].

Our ultimate goal is to learn a conditional probability distribution $p_\theta(\mathbf{y}|\mathbf{x})$ that assigns high probability to correct outputs $\mathbf{y}$. To enforce this goal, we define an auxiliary classifier D_ϕ (parametrized by ϕ) called a discriminator which scores outputs in the label space, and tries to assign higher scores to a small set of given representative output labels $\mathcal{D}_S$, while assigning lower scores to samples of $p_\theta(\mathbf{y}|\mathbf{x})$. Thus, the discriminator learns to effectively extract latent constraints that hold over the output space and are implicitly encoded in the output samples $\mathcal{D}_S$. The goal of the $p_\theta(\mathbf{y}|\mathbf{x})$ is to produce outputs that score higher under the discriminator, meeting the constraints.

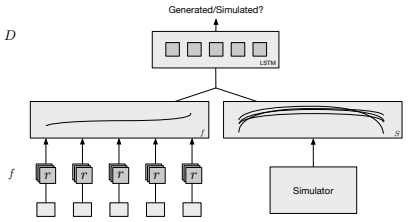
We train D_ϕ and $p_\theta(\mathbf{y}|\mathbf{x})$ jointly through adversarial training, optimizing the Wasserstein GAN objective [14, 15]

$$\min_\theta \max_\phi \mathcal{L}^{\mathcal{A}} = \mathbb{E}_{\mathbf{y}\sim p_s(\mathbf{y})}[D_\phi(\mathbf{y})] - \mathbb{E}_{\mathbf{y}\sim p_\theta(\mathbf{y}|\mathbf{x}),\mathbf{x}\sim p_d(\mathbf{x})}[D_\phi(\mathbf{y})] \tag{4}$$

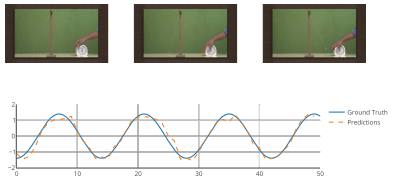
where $p_s(\mathbf{y})$ denotes the distribution of samples in $\mathcal{D}_S$ and $p_d(\mathbf{x})$ denotes the distribution of the input data. At the optimal solution to the objectives in Eq. 4, the discriminator cannot distinguish between the given set of labels and those predicted by the model, suggesting that the latter satisfy the set of constraints identified by the discriminator. Figure 2(a) shows an overview of the adversarial constraint learning framework in the context of an object tracking task.

When given a set of labeled examples, we may formulate our objective as the sum of a constraint learning term (over both labeled and unlabeled data) and a standard regression loss term (over the labeled data), resulting in a semi-supervised framework,

$$\mathcal{L}^{\mathcal{SS}} = \mathcal{L}^{\mathcal{A}} + \alpha\, \mathbb{E}_{\mathbf{x}_i,\mathbf{y}_i\sim p_l}[l(p_\theta(\mathbf{y}|\mathbf{x}_i), \mathbf{y}_i)] \tag{5}$$



(a) Our architecture trains f_θ (or in this example, r_θ) by asking it to generate trajectories T_f that cannot be discriminated from sample trajectories T_S. Training D eliminates the needs to hand-engineer constraints.



(b) Top: frames from video used in the pendulum experiment. Bottom: the network is trained to detect angles that cannot be distinguished from the simulated dynamics, encouraging it to track the metal ball over time.

Figure 2: Architecture and results of the pendulum detection experiment.

where $\mathcal{L}^{\mathcal{A}}$ is the adversarial constraint learning objective defined in Eq. 4, p_l is the distribution for the labeled dataset, and α is a hyperparameter that balances between fitting to the general (implicit) output distribution (first term) and fitting to the explicit labeled dataset (second term).

Constraint Learning by Matching Distributions. Our approach can also be interpreted as matching the marginal distribution over predicted labels to the label samples.

Assuming we can obtain samples from $p(\mathbf{y})$, another way of formulating the constraint h is to let

$$\mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[h(f_\theta(\mathbf{x}))] = \rho(\mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[p_\theta(\mathbf{y}|\mathbf{x})], p(\mathbf{y})) \tag{6}$$

where ρ is some divergence/metric that can be approximately computed through samples from $p(\mathbf{y})$ (such as KL divergence, MMD, JSD, or EMD). Minimizing ρ ensures that $p_\theta(\mathbf{y}|\mathbf{x})$ provides a reasonable $\mathbf{y}$ that lies in the true manifold of labels.

4 Experimental Results

We evaluate our framework on two structured prediction problems. First, we train a network to track the angle of a pendulum in a video using supervision provided by a physics-based simulator. Next, we extend the output space to higher dimensions and perform human pose estimation.

In both experiments, we consider $p_\theta(\mathbf{y}|\mathbf{x})$ to be a Dirac-delta distribution $\delta(\mathbf{y} - f_\theta(\mathbf{x}))$, thus we refer to the conditional probabilistic model as the mapping $f_\theta(\mathbf{x}) : \mathcal{X} \to \mathcal{Y}$, implemented as a neural network parametrized by θ. Please refer to the appendix for detailed network architectures and training details.

4.1 Pendulum Tracking

For this task, we aim to extract the angle of the pendulum on images from a YouTube video [16], i.e., learn a regression mapping $r_\theta : \mathbb{R}^{h\times w\times 3} \to \mathbb{R}$. Since the outputs of r_θ over continuous frames form sine waves and are thus constrained, we concatenate continuous outputs of r_θ and form a high dimensional trajectory $f_\theta([\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]) = [r_\theta(\mathbf{x}_1), r_\theta(\mathbf{x}_2), \cdots, r_\theta(\mathbf{x}_n)]$. We also design a simulator that produces sine waves with frequency based on observations.

We manually label the horizontal position of the ball of the pendulum in each frame in the test set, and measure the correlation of the predicted position with the ground truth label in pixels. As shown in Figure 2, we achieve a correlation of 96.3% while using explicit formulas to supervise training yields 96%. Our model is generally robust against the accuracy of the simulator.

Overall, the real world pendulum experiment shows that our adversarial framework makes it possible to extract object information from real images using only a simulator of physics that the object obeys.

Figure 3: Pose estimation results when 25% of the training data are labeled. The regression network takes in single image and outputs the location of 6 joints (in green).

4.2 Pose Estimation

In this experiment, we benchmark the proposed model on pose estimation, which has a larger output space. We use multi-modal action database (MAD) [17], and aim to detect left/right hip/knee/foot on images. We attempts to learn a regression network $r_\theta : \mathbb{R}^{h\times w\times 3} \rightarrow \mathbb{R}^k$, where k denotes the number of joints we detect. As before, r_θ is applied to several frames to produce a trajectory. We train the network based on a sequence of images, and output a sequence of joint locations, which should be indistinguishable from the sample data. Similar to the pendulum experiment, r_θ is applied to each frame independently, *no knowledge of the neighbor frames* is used in this process.

In this experiment, the output samples are labels $\{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$, but we assume we have no knowledge of the corresponding input vectors. The results and notations are shown in Table 1 listed in appendix. When only trained adversarially ("0%+adv"; i.e., optimizing just $\mathcal{L}^{\mathcal{A}}$), the network is able to find the correct "shape" of the joints for each frame, but the predictions are biased with a minor shift $(\Delta x, \Delta y)$. It shows that mere adversarial training is insufficient for this task. To mitigate the problem, we provide a small amount of labeled training data and consider the semi-supervised objective $\mathcal{L}^{\mathcal{SS}}$. The label loss helps adjust f_θ to output the precise locations. Given just 25% of the available labeled data, f_θ converges to detecting the joints with high accuracy, as shown in Figure 3. "50%+adv" achieves same performance as "100%" (fully supervised on all training data) on the detection of feet.

We also evaluate the following baselines. First, we test the result of pure guessing, using a randomly picked label from the simulator for each test data as its prediction. Furthermore, we run ablative experiments "$t\%$", where only $t\%$ of the labeled data is used for supervised learning. In this case, some data points are neither trained nor tested. "$t\%$+adv" generally shows much better results compared to "$t\%$". Lastly, we test "$t\%$+rand", where we randomly assign labels from the simulator to unlabeled data points, and then use supervised learning. Although this random label assignment could be incorrect, it could still provide some signal. However, the results demonstrate that if the remaining data are used in this random manner, the detection accuracy rarely increases. This emphasizes the importance of our adversarial training loss.

This experiment shows that our model is robust when the output space is large; hand-crafting formula-based constraints is tedious and error-prone. Our model avoids the problematic complexity by extracting constraints implicitly from the output samples, which are capable of describing the feature of output space. In cases where limited input-output pairs are given, our model can be trained in a semi-supervised way by minimizing the label loss and the adversarial loss simultaneously.

5 Conclusion

We propose a new framework for semi-supervised structured prediction using adversarial constraint learning. Instead of using hand-designed constraints, which are difficult to obtain and application-specific, we instead learn the constraints from data using an implicit generative model with adversarial loss. Experimental results on structured prediction tasks show that our method is robust against high dimensional outputs. As future work, we plan to explore adversarial constraint learning applied to scenarios where the output space is characterized by multiple constraints (the output distribution is factored).

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[3] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649, IEEE, 2013.

[4] I. Shcherbatyi and B. Andres, "Convexification of learning from constraints," in *German Conference on Pattern Recognition*, pp. 79–90, Springer, 2016.

[5] R. Stewart and S. Ermon, "Label-free supervision of neural networks with physics and domain knowledge.," in *AAAI*, pp. 2576–2582, 2017.

[6] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1, pp. 107–136, 2006.

[7] M.-W. Chang, L. Ratinov, and D. Roth, "Guiding semi-supervision with constraint-driven learning," in *ACL*, pp. 280–287, 2007.

[8] A. Choi, G. Van den Broeck, and A. Darwiche, "Tractable learning for structured probability spaces: A case study in learning preference distributions," in *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[11] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[12] S. Mohamed and B. Lakshminarayanan, "Learning in implicit generative models," *arXiv preprint arXiv:1610.03483*, 2016.

[13] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1718–1727, 2015.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[16] KClassScienceChannel, "Time period of a pendulum depends on its length." `https://www.youtube.com/watch?v=02w9lSii_Hs`, 2013.

[17] D. Huang, S. Yao, Y. Wang, and F. De La Torre, "Sequential max-margin event detectors," in *European conference on computer vision*, pp. 410–424, Springer, 2014.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[20] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2878–2890, 2013.

A Network Layout

We use the gradient penalty Wasserstein GAN [14] throughout the experiments since it reduces training time and is more stable in general. In both experiments, the discriminator D_ϕ is a recurrent neural network (RNN) with Long Short-Term Memory (LSTM) units. The input of D_ϕ is first fed through 3 FC/ReLU layers with 64 hidden dimensions and terminating in n LSTM inputs of dimension 128, where n is the group size, and equals 5 in both experiments. After processing the entire sequence, we pass the final LSTM hidden state through a single FC layer to score the input sequence. For the pendulum tracking, we use a convolutional neural network (CNN) for the regression network r_θ, with 2 Conv/ReLU/Pool modules followed by a FC layer with 1 output. In the pose estimation experiment, we adopt a VGG-based network, instead of pooling, we follow the advice of [18] and use strided convolution layers to downsample. After the extraction of 512 feature maps, we use two FC layers to output a 12-dimensional vector.

B Training Details

B.1 Pendulum Tracking

We sample the 17 second video at 10 frames per second, resulting in a total of 170 images. We hold out 34 images for evaluation. We manually observe that the pendulum completes one full oscillation approximately every 12 frames. Based on this observation, we write a simulator of these dynamics with a simple harmonic oscillator having a fixed amplitude and random sample period of 10 to 14 frames. r_θ takes in one image in 56×56 size and outputs a scalar, representing the angle of the pendulum in the image. D_ϕ is trained to distinguish between the output of r_θ across 5 continuous images and a random trajectory sampled from the simulator. We train the network with the Adam optimizer for $5,000$ iterations at a learning rate of 10^{-4} [19].

B.2 Pose Estimation

CMU multi-modal action database (MAD) [17] contains 40 videos of 20 subjects (2 for each subject) performing a sequence of 35 actions in each video. We edit the 40 videos, extract the frames when the subjects perform the "Jump and Side-Kick" action. The processed dataset contains 620 valid frames (40 groups). The 40 groups of motion data are divided into 32 groups and 8 groups for training and testing respectively. Each group contains 14 to 17 images. In our experiment, we train on randomly selected intervals of 5 contiguous frames. We use PCK@0.1 [20] for evaluation. The prediction is considered correct if and only if it lies within $\alpha \max(h, w)$ pixels from the correct location, where h and w denote the height and width of the tightest bounding box that covers the whole body, and we use $\alpha = 0.1$, which is a fairly strict criterion.

Images are resized to 64×64 pixels before r_θ is applied. r_θ takes a single image as input and outputs a 12 dimensional vector, representing the location of 6 joints. We concatenate the outputs of the regression network for each group and pass it to the discriminator. The discriminator is LSTM-based and tries to tell the generated locations and sample joint locations apart. We use the Adam optimizer with a learning rate of 10^{-4}. The network is trained for $20,000$ iterations. In each iteration, D_ϕ is updated 5 times and r_θ once. We split the dataset randomly and repeat training and testing 50 times. The results are averaged over these 50 trials.

C Pose Estimation Results

PCK@0.1(%)	Left Hip	Left Knee	Left Foot	Right Hip	Right Knee	Right Foot
RSS	0.517	0.414	0.300	0.520	0.412	0.299
0%+rand	0.743	0.620	0.493	0.750	0.604	0.442
0%+adv	0.846	0.578	0.414	0.824	0.636	0.514
12.5%	0.820	0.794	0.717	0.813	0.729	0.625
12.5%+rand	0.789	0.623	0.498	0.819	0.598	0.464
12.5%+adv	0.857	0.831	0.783	0.939	0.823	0.668
25%	0.766	0.869	0.768	0.769	0.885	0.737
25%+rand	0.852	0.804	0.560	0.864	0.763	0.510
25%+adv	0.923	0.842	0.829	0.914	0.850	0.802
37.5%	0.912	0.884	0.813	0.913	0.897	0.796
37.5%+rand	0.896	0.714	0.591	0.899	0.743	0.579
37.5%+adv	0.944	0.916	0.858	0.951	0.898	0.867
50%	0.943	0.903	0.809	0.958	0.904	0.773
50%+rand	0.841	0.725	0.606	0.847	0.815	0.733
50%+adv	0.965	0.895	0.861	0.968	0.922	0.872
100%	0.994	0.950	0.876	0.994	0.977	0.858

Table 1: PCK@0.1 results on MAD. "Random simulator sample" (RSS) makes a prediction using a random label from the simulator (baseline). "$t\%$" means that we train on $t\%$ of the labeled data (standard supervised learning). "$t\%$+rand" means that we additionally randomly assign labels from the simulator to the remaining $(1 - t\%)$ of the training data (baseline). "$t\%$+adv" means that we use $t\%$ of the labeled training data (supervised loss), with the additional adversarial loss (our approach). Our approach consistently outperforms the baselines.

Training Malicious Domain Name Classifiers with Real, Heuristically Labeled Data

Bin Yu
CTO Office, Infoblox
Santa Clara, California
biny@infoblox.com

Daniel L. Gray
Institute of Technology
University of Washington, Tacoma
dangray@uw.edu

Martine De Cock*
Institute of Technology
University of Washington, Tacoma
mdecock@uw.edu

Anderson C. A. Nascimento
Institute of Technology
University of Washington, Tacoma
andclay@uw.edu

Abstract

Domain generation algorithms (DGAs) have become commonplace in malware that seek to establish command and control communication between an infected machine and the botmaster. DGAs dynamically generate large volumes of malicious domain names, only a few of which are registered by the botmaster and subsequently resolved when the malware on the infected machine tries to access them. Deep neural networks that can classify domain names as benign or malicious are of great interest in the fight against DGAs, because these networks can learn features themselves instead of having to rely on human engineered features. This is of particular importance in the cybersecurity domain, as malware constantly evolves. Keeping deep neural networks current for real-time detection of DGAs requires training them with massive up-to-date data. Since obtaining large amounts of clean ground truth labeled data is infeasible, an attractive alternative is the use of noise-not-free yet practical data collected based on the heuristic that most DGA domain names do not resolve. In this paper we compare a recurrent neural network trained on a small data set of ground truth data vs one trained on a data set of 50 million domain names obtained from real traffic via heuristic labeling.

*Guest professor at Ghent University, Dept. of Applied Mathematics, Computer Science and Statistics

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

EZLearn: Exploiting Organic Supervision in Large-Scale Data Annotation

Maxim Grechkin
Paul G. Allen School of Computer Science
University of Washington
Seattle, WA

Hoifung Poon
Microsoft Research
Redmond, WA

Bill Howe
Information School
University of Washington
Seattle, WA

Abstract

Many real-world applications require large-scale data annotation, such as identifying tissue origins based on gene expression profiles and classifying images into semantic categories. Annotation classes are often numerous and subject to changes over time, and annotating examples has become the major bottleneck for supervised learning methods. In science and other high-value domains, large repositories of data samples are often available, together with two sources of *organic supervision*: a lexicon for the annotation classes, and text descriptions that accompany some data samples. Distant supervision has emerged as a promising paradigm for exploiting such indirect supervision by automatically annotating examples where the text description contains a class mention in the lexicon. However, due to linguistic variations and ambiguities, such training data is inherently noisy, which limits the accuracy in this approach. In this paper, we introduce an auxiliary natural language processing system for the text modality, and incorporate co-training to reduce noise and augment signal in distant supervision. Without any manually labeled data, our *EZLearn* system learned to accurately annotate data samples in functional genomics and scientific figure comprehension, even substantially outperforming state-of-the-art supervised methods trained on tens of thousands of annotated examples.

Introduction

The confluence of technological advances and the open data movement [20] has led to an explosion of publicly available datasets, heralding an era of data-driven hypothesis generation and discovery in high-value applications [24]. A prime example is *open science*, which promotes open access to scientific discourse and data to facilitate large-scale data reuse and scientific collaboration [7]. In addition to enabling reproducibility, this trend has the potential to accelerate scientific discovery, reduce the cost of research, and facilitate automation [25, 16].

However, progress is hindered by the lack of consistent and high-quality annotations. For example, tissues from neurons to blood share the same genome, but vary in gene expression, which is crucial to understanding cell differentiation and cancer [10, 9]. The NCBI Gene Expression Omnibus (GEO) [3] contains over two million sample gene expression profiles, yet only a fraction of them have explicit tissue annotation. As a result, only 20% of the datasets have ever been reused, and tissue-specific expression studies are still being done at small scale [24]. Similarly, figures in scientific papers convey rich information, but there is no principled way to search them by semantics [14].

Annotating data samples with standardized classes is the canonical multi-class classification problem, but standard supervised approaches are difficult to apply. Hiring experts to annotate examples for thousands of classes such as tissue types is unsustainable. Crowd-sourcing is generally not applicable, as annotation requires expertise that most crowd workers do not possess. Moreover, the annotation standard is often revised over time, incurring additional cost for labeling new examples.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

While labeled data is expensive and difficult to create at scale, unlabeled data is usually in abundant supply. Many methods have been proposed to exploit it, but they typically still require labeled examples to initiate the process [1, 18, 6]. Even zero-shot learning, where the name implies learning with no labeled examples for *some* classes, still requires labeled examples for related classes [22, 26].

In this paper, we propose *EZLearn*, which makes annotation learning easy by exploiting two sources of *organic supervision*. First, the annotation classes generally come with a lexicon for standardized references (e.g., "liver", "kidney", "acute myeloid leukemia cell" for tissue types). While labeling individual data samples is expensive and time-consuming, it takes little effort for a domain expert to provide a few example terms for each class. In fact, in the sciences and other high-value applications, such a lexicon is often available as part of an existing domain ontology. For example, the Brenda Tissue Ontology specifies 4931 human tissue types, each with a list of standard names [8]. We call such indirect supervision "organic" to emphasize that it is readily available as an integral part of a given domain. Second, data samples are often accompanied by a text description, some of which directly or indirectly mention the relevant classes (e.g., the caption of a figure, or the description entered by a lab technician for a gene expression sample). Together with the lexicon, these descriptions present an opportunity for exploiting distant supervision by generating noisy labeled examples at scale [19].

In practice, however, there are serious challenges to enact this learning process. Descriptions are created for general human consumption, not as high-quality machine-readable annotations. They are provided voluntarily by data owners and lack consistency of any kind. Ambiguity, typos, abbreviations, and non-standard references abound [15, 25]. Additionally, annotation standard evolves over time, some terms become obsolete but were used in older samples. As a result, while there are potentially many data samples whose description contains class information, only a fraction of them can be identified using distant supervision, and noises are introduced due to reference ambiguity. This problem is particularly acute for domains with a large number of classes and/or frequent update.

To best exploit indirect supervision using all instances, *EZLearn* introduces an auxiliary text classifier for handling complex linguistic phenomena in descriptions. This auxiliary classifier first uses the lexicon to find exact matches to teach the main classifier. In turn, the main classifier helps the auxiliary classifier improve by annotating additional examples where class mentions are non-standard or ambiguous. This co-supervision continues until neither classifier can improve any further. Effectively, *EZLearn* represents the first attempt in combining distant supervision and co-training, using text as the auxiliary modality for learning. Figure 1 shows the architecture.

To investigate the effectiveness and generality of *EZLearn*, we applied it to two important applications in functional genomics and scientific figure comprehension, which differ substantially in domain characteristics such as sample input dimension and description length. In functional genomics, there are thousands of well-established classes. In scientific figure comprehension, prior work only considers three coarse classes, and we expand them to twenty-four finer-grained ones. In both scenarios, *EZLearn* successfully learned an accurate classifier with zero manually labeled examples.

EZLearn

Let $X = \{x_i : i\}$ be the set of data samples and C be the set of classes. Automating annotation amounts to learning a multi-class classifier $f : X \rightarrow C$. For example, x_i may be a gene expression profile, whereas C is the set of tissue types. Additionally, t_i denotes the text description that accompanies x_i. Sometimes, the description is not available, in which case t_i is the empty string. By default, there are no available labeled examples (x, y^*) where $y^* \in C$ is the true class for annotating $x \in X$. Instead, *EZLearn* assumes that a lexicon L_c is available with a set of example terms for referencing $c \in C$. Note that we do not assume that L_c is complete, nor that such terms are unambiguous. Rather, we simply require that L_c is non-empty for any c of interest.

To handle linguistic variations and ambiguities, *EZLearn* introduces an auxiliary classifier $f_T : T \rightarrow C$, where $T = \{t_i : i\}$ is the set of text descriptions that accompany the data samples. f_T is initialized using the initial labeled set D^0, which contains all (x_i, c) where t_i contains a class reference in lexicon L_c. At iteration k, we first train a new main classifier f^k using D^{k-1}. We then apply f^k to X and create a new labeled set D^k_T, which contains all (t_i, c) where $f^k(x_i) = c$. We then train a new text classifier f^k_T using D^k_T, and create the new labeled set D^k with all (x_i, c) where $f^k_T(t_i) = c$. This process continues until convergence, which is guaranteed given conditional independence of the two views [1]. Empirically, it happens quickly. Algorithm 1 shows the *EZLearn* algorithm.

Method	# Labeled	# All	AUPRC	Prec@0.5	Use expression	Use text	Use lexicon	Use EM
URSA	14510	0	0.40	0.52	yes	no	no	no
Co-EM	14510	116895	0.51	0.61	yes	yes	no	yes
Dist. Sup.	0	116895	0.59	0.63	yes	yes	yes	no
EZLearn	0	116895	**0.67**	**0.83**	yes	yes	yes	yes

Table 1: Comparison of test results between *EZLearn* and state-of-the-art supervised, semi-supervised, and distantly supervised methods on the Comprehensive Map of Human Gene Expression. We reported the area under the precision-recall curve (AUPRC) and precision at 0.5 recall.

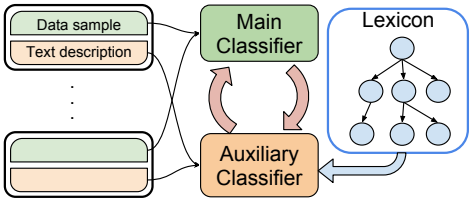


Figure 1: *EZLearn* architecture: an auxiliary text classifier is introduced to bootstrap from the lexicon (often available from an ontology) and co-teach the main classifier until convergence.

Algorithm 1 *EZLearn*

Input: Data samples X, text descriptions T, annotation classes C, and lexicon L_c containing example references for each class $c \in C$.
Output: Trained classifiers $f : X \rightarrow C$ (main) and $f_T : T \rightarrow C$ (auxiliary).
Initialize: Generate the initial training data D^0 by adding all (x_i, c) where $x_i \in X$ and its text description $t_i \in T$ mentions a term in L_c.
for $k = 1 : N_{iter}$ **do**
  $f \leftarrow \texttt{Train}_{\texttt{main}}(D^{k-1})$; $D^k_T \leftarrow f(X)$
  $f_T \leftarrow \texttt{Train}_{\texttt{aux}}(D^k_T)$; $D^k \leftarrow f_T(T)$
end for

In both the initialization step and later iterations, a labeled set might contain more than one class for a sample, which is not a problem for the learning algorithm and is useful when there is uncertainty about the correct class. We can use any classifier for $\texttt{Train}_{\texttt{main}}$ and $\texttt{Train}_{\texttt{aux}}$. Features for the main classifier are domain-specific and can be what any reasonable supervised approach might use. For the text classifier, we use standard n-gram features, which are effective in both applications we experimented on. It is possible to tailor them for specific domains. Generally, a classifier will output a score for each class, rather than predicting a single class. The score reflects the confidence in predicting the given class. *EZLearn* generates the labeled set by adding all (sample,class) pairs for which the score crosses a threshold, which is a hyperparameter. We chose 0.3 in preliminary experiments, which allows up to 3 classes to be assigned to a sample.

Application: Functional Genomics

Annotation task The goal is to annotate tissue types based one gene expressions. The input is a gene expression profile (a 20,000-dimension vector with a numeric value signifying the expression level for each gene). The output is a tissue type. We used the BRENDA Tissue Ontology [8], which contains 4931 human tissue types. For gene expression data, we used the Gene Expression Omnibus [5], a popular repository run by the National Center for Biotechnology Information. We focused on the most common data-generation platform (Affymetrix U133 Plus 2.0), and obtained a dataset of 116895 human samples. Each sample was processed using UPC to minimize batch effects and normalize expression values to [0,1] [23]. Text descriptions were obtained from GEOmetadb [31].

Main classifier We implemented $\texttt{Train}_{\texttt{main}}$ using deep denoising auto-encoder (DAE) with three LeakyReLU layers to convert the gene expression profile to a 128-dimensional vector [30], followed by multinomial logistic regression, trained in Keras [2], using L2 regularization with weight $1e-4$ and RMSProp optimizer [27] with default parameters.

Auxiliary classifier We implemented $\texttt{Train}_{\texttt{aux}}$ using the fastText classifier with their recommended parameters (25 epochs and starting learning rate of 1.0) [13]. The auxiliary classifier is initialized by simply predicting the most specific class in BRENDA with one of its standard terms appearing in the description. It is possible to have multiple matching classes, in which case all were added to the labeled set for training a new main classifier. In principle, we can continue the alternating training steps until convergence, when neither classifier's predictions change significantly. In practice, convergence usually comes quickly [21], and we simply ran all experiments with five iterations.

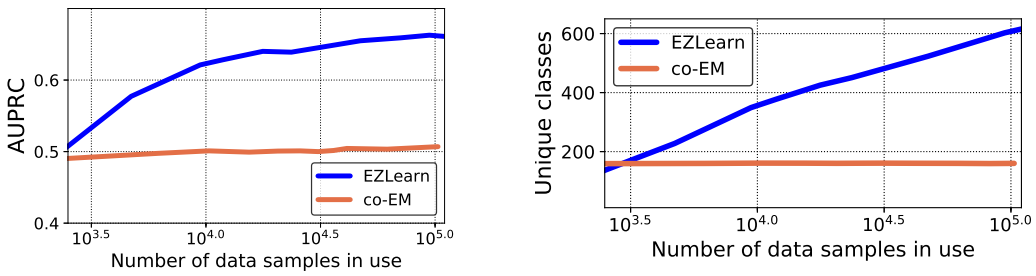


Figure 2: (a) Comparison of test accuracy with varying amount of unlabeled data.(b) Comparison of number of unique classes in high-confidence predictions with varying amount of unlabeled data.

Systems We compared *EZLearn* with URSA [15], the state-of-the-art supervised method that is trained on a large labeled dataset of 14,510 examples and used a sophisticated Bayesian method to refine SVM classification based on the ontology. We also compared it with co-training [1] and its variant co-EM [21], two representative methods for leveraging unlabeled data that also use an auxiliary view to support the main classification. Unlike *EZLearn*, they use labeled data to train their initial classifiers. After the first iteration, high-confidence predictions on the unlabeled data are added to the labeled examples. In co-training, once a unlabeled sample is added to the labeled set, it is not reconsidered again, whereas in co-EM, all of them are re-annotated in each iteration. We found that co-training and co-EM performed essentially the same, and so only report the co-EM results.

Evaluation We evaluated the classification results using *ontology-based precision and recall*. For each singleton class, predicted or gold, we expand it to include its ancestors other than the root (representing everything). We can then measure precision and recall in the standard way. Namely, precision is the proportion of correct predicted classes among all predicted classes, and recall is the proportion of correct predicted classes among gold classes, with ancestors included in all cases. This closely resembles the approach by [29], except that we are using the "micro" version (i.e., the predictions for all samples are first combined before measuring precision and recall), which is more appropriate in our applications. If the system predicts an irrelevant class in a different branch under the root, the overlap between the predicted and gold set is empty and the penalty is severe. If the predicted class is an ancestor (more general) or a descendent (more specific), there is overlap and the penalty is less severe, with overly general or specific predictions penalized more than close neighbors. We tested on the Comprehensive Map of Human Gene Expression (CMHGP), the largest expression dataset with manual tissue annotations [28]. CMHPG used tissue types from the Experimental Factor Ontology (EFO) [17], which can be mapped to the BRENDA Tissue Ontology. To make the comparison fair, 7,209 CMHGP samples that were in the supervised training set for URSA were excluded from the test set. The final test set contains 15,129 samples of 628 tissue types.

Results We report both the area under the precision-recall curve (AUPRC) and the precision at 0.5 recall. Table 1 shows the main classification results. All results were averaged over fifteen runs (except URSA). Remarkably, without using any labeled data, *EZLearn* outperformed the state-of-the-art supervised method by a wide margin, improving AUPRC by an absolute 27 points over URSA, and over 30 points in precision at 0.5 recall. Compared to distant supervision, the use of EM led to further significant gains of 8 points in AUPRC and 20 points in precision at 0.5 recall. Compared to co-EM, *EZLearn* improves AUPRC by 16 points and precision at 0.5 recall by 22 points. To investigate why *EZLearn* attained such a clear advantage even against co-EM, we compared their performance using varying amount of unlabeled data (averaged over fifteen runs). Figure 2(a) shows the results. Note that the x-axis (number of unlabeled examples in use) is in log-scale. Co-EM barely improves with more unlabeled data, whereas *EZLearn* improves substantially from 2% to 100% of unlabeled data. To understand why this is the case, we further compare the number of unique classes predicted by the two methods. See Figure 2(b). Co-EM is confined to the classes in its labeled data and its use of unlabeled data is limited to the extent of improving predictions for those classes. In contrast, by using organic supervision to generate noisy examples, *EZLearn* can expand the classes in its purview with more unlabeled data, while improving predictive accuracy for individual classes.

Application: Scientific Figure Comprehension

Figures communicate key results and provide visual explanations of complex concepts. However, while text understanding has been intensely studied, figures have received much less attention in the past. A notable exception is the Viziometrics project [14], which annotated a large number of examples for classifying scientific figures into semantic classes. Due to the considerable cost of labeling examples, they only used five coarse classes: `Plot`, `Diagram`, `Image`, `Table` and `Equation`. We exclude the last two as they do not represent true figures. In practice, figure-comprehension projects can be much more useful if they include larger set of more fine-grained classes. To explore this direction, we devised an ontology where `Plot`, `Diagram`, and `Image` are further refined into a total of twenty-four classes, such as `Boxplot`, `MRI` and `PieChart`. However, to cover these new classes, the supervised-learning approach adopted by Viziometrics will require annotating an even larger number of examples. *EZLearn*, on the other hand, does not require manually labeled data, and can be applied directly to learning the fine-grained classifier.

Annotation task The goal is to annotate figures with semantic types in the predefined ontology. The input is the image of a figure with varying size. The output is the semantic type. We obtained the data from the Viziometrics project [14] through its open API. For simplicity, we focused on single-pane figures, yielding 1,174,456 images along with free-text captions. As in the gene expression case, captions might be empty or missing.

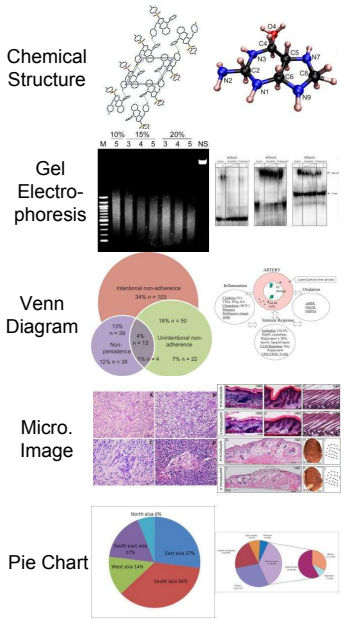


Figure 3: Example annotations by *EZLearn*, all chosen randomly among figures with no class information in their captions.

Systems Each figure image was first resized and converted to a 2048-dimensional real-valued vector using a convolutional neural network [11] trained on ImageNet [4]. We follow [12] and use the ResNet-50 model with pre-trained weights provided by Keras [2]. We used the same classifiers and hyperparameters as in the functional genomics application. We used a lexicon that simply comprises of the names of the new classes, and compared *EZLearn* with the Viziometrics classifier. We also compared with a lexicon-informed baseline that annotates a figure with the most specific class whose name is mentioned in the caption (or root otherwise).

Evaluation We followed the functional genomics application and evaluated on ontology-based precision and recall. Since the new classes are direct refinement of the old ones, we can also evaluate the Viziometrics classifier using this metric. To the best of our knowledge, there is no prior dataset or evaluation for figure annotation with fine-grained semantic classes. Therefore, we manually annotated an independent test set of 500 examples.

Results *EZLearn* substantially outperformed both the lexicon-informed baseline and the Viziometrics classifier, scoring 79% in AUPRC compared to 44% (lexicon baseline) and 53% (Viziometric), and 88% precision at 0.5 recall compared to 31% (lexicon baseline) and 43% (Viziometric). The state-of-the-art Viziometrics classifier was trained on 3271 labeled examples, and attained an accuracy of 92% on the coarse classes. So the gain attained by *EZLearn* reflects its ability to extract a large amount of fine-grained semantic information missing in the coarse classes. Figure 3 shows example figure annotations by *EZLearn*, all chosen from figures with no class mention in their captions.

Discussion

We propose *EZLearn* for large-scale data annotation, which exploits two readily available sources of organic supervision: a lexicon containing standard class references and text descriptions provided by data owners. By introducing an auxiliary text classifier to co-teach the main classifier, *EZLearn* leverages co-training to reduce noise and amplify signal in distant supervision. *EZLearn* is well suited to the sciences and other high-value domains that contain a large number of classes and/or undergo frequent update. Experiments in functional genomics and scientific figure comprehension show that *EZLearn* is broadly applicable, robust to noise, and capable of learning accurate classifier without any manually labeled data, even outperforming state-of-the-art supervised systems by a wide margin. Future directions include: incorporate word embedding and other known semantic similarity; leverage hierarchical relations among annotation classes; apply *EZLearn* to other domains.

References

[1] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[2] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[3] Emily Clough and Tanya Barrett. The gene expression omnibus database. *Methods Mol Biol*, 1418, 2016.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR 2009*, pages 248–255. IEEE, 2009.

[5] Ron Edgar, Michael Domrachev, and Alex E Lash. Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic acids research*, 30(1):207–210, 2002.

[6] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE TPAMI*, 28(4):594–611, 2006.

[7] Sascha Friesike, Bastian Widenmayer, Oliver Gassmann, and Thomas Schildhauer. Opening science: towards an agenda of open science in academia and industry. *J. of Tech. Transfer*, Aug 2015.

[8] Marion Gremse, Antje Chang, Ida Schomburg, Andreas Grote, Maurice Scheer, Christian Ebeling, and Dietmar Schomburg. The BRENDA Tissue Ontology (BTO): the first all-integrating ontology of all organisms for enzyme sources. *Nucleic Acids Research*, 2011.

[9] Maria Gutierrez-Arcelus, Halit Ongen, Tuuli Lappalainen, Stephen B Montgomery, Alfonso Buil, Alisa Yurovsky, Julien Bryois, Ismael Padioleau, Luciana Romano, Alexandra Planchon, et al. Tissue-specific effects of genetic and epigenetic variation on gene regulation and splicing. *PLoS Genet*, 11(1):e1004958, 2015.

[10] Douglas Hanahan and Robert A Weinberg. Hallmarks of cancer: the next generation. *cell*, 144(5):646–674, 2011.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[12] Bill Howe, Po-shen Lee, Maxim Grechkin, Sean T Yang, and Jevin D West. Deep mapping of the visual literature. In *WWW Companion*, pages 1273–1277, 2017.

[13] Armand Joulin, Edouard Grave, and Piotr Bojanowski Tomas Mikolov. Bag of tricks for efficient text classification. *EACL 2017*, page 427, 2017.

[14] Po-shen Lee, Jevin D West, and Bill Howe. Viziometrics: Analyzing visual information in the scientific literature. *IEEE Transactions on Big Data*, 2017.

[15] Young-suk Lee, Arjun Krishnan, Qian Zhu, and Olga G. Troyanskaya. Ontology-aware classification of tissue and cell-type signals in gene expression profiles across platforms and technologies. *Bioinformatics*, 29(23):3036–3044, 2013.

[16] Maxwell W. Libbrecht and William Stafford Noble. Machine learning applications in genetics and genomics. *Nat. Rev. Genetics*, 2015.

[17] James Malone, Ele Holloway, Tomasz Adamusiak, Misha Kapushesky, Jie Zheng, Nikolay Kolesnikov, Anna Zhukova, Alvis Brazma, and Helen Parkinson. Modeling sample variables with an experimental factor ontology. *Bioinformatics*, 2010.

[18] David McClosky and Eugene Charniak. Self-training for biomedical parsing. In *ACL*, 2008.

[19] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL*, pages 1003–1011, 2009.

[20] Jennifer C Molloy. The open knowledge foundation: open data means better science. *PLoS Biol*, 9(12):e1001195, 2011.

[21] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[22] Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009.

[23] Stephen R. Piccolo, Michelle R. Withers, Owen E. Francis, Andrea H. Bild, and W. Evan Johnson. Multiplatform single-sample estimates of transcriptional activation. *PNAS*, 2013.

[24] Heather A. Piwowar and Todd J. Vision. Data reuse and the open data citation advantage. *PeerJ*, 1:e175, October 2013.

[25] Johan Rung and Alvis Brazma. Reuse of public genome-wide gene expression data. *Nature Reviews Genetics*, 2013.

[26] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013.

[27] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural nets for ML*, 2012.

[28] Aurora Torrente, Margus Lukk, Vincent Xue, Helen Parkinson, Johan Rung, and Alvis Brazma. Identification of cancer related genes using a comprehensive map of human gene expression. *PLOS ONE*, 11(6):1–20, 06 2016.

[29] Karin Verspoor, Judith Cohn, Susan Mniszewski, and Cliff Joslyn. A categorization approach to automated ontological function annotation. *Protein Science*, 15(6):1544–1549, 2006.

[30] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103. ACM, 2008.

[31] Yuelin Zhu, Sean Davis, Robert Stephens, Paul S. Meltzer, and Yidong Chen. GEOmetadb: powerful alternative search engine for the gene expression omnibus. *Bioinformatics*, 2008.

A Semantic Loss Function for Deep Learning Under Weak Supervision*

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang & Guy Van den Broeck
Computer Science Department
University of California, Los Angeles

Abstract

This paper develops a novel methodology for using symbolic knowledge in deep learning. We define a semantic loss function that bridges between neural output vectors and logical constraints. This loss function captures how close the neural network is to satisfying the constraints on its output. An experimental evaluation shows that our semantic loss function effectively guides the learner to achieve (near-)state-of-the-art results on semi-supervised multi-class classification. Moreover, it significantly increases the ability of the neural network to predict structured objects under weak supervision, such as rankings and shortest paths.

1 Introduction

The widespread success of representation learning raises the question of which AI tasks are amenable to deep learning, which require classical model-based symbolic reasoning, and whether we can benefit from an integration of both. In recent years, significant effort has gone towards various ways of using representation learning to solve tasks that were previously tackled by symbolic methods. Such efforts include neural computers, Turing machines, and differentiable programming (e.g., [15, 31, 32, 37]), relational embeddings, deep learning for graph data, and neural theorem proving (e.g., [1, 12, 25, 26]), and many more. Other work has sought to augment deep learning with symbolic knowledge through logical or arithmetic constraints (e.g., [9–11, 16, 19, 21, 24, 28, 33, 35, 36]).

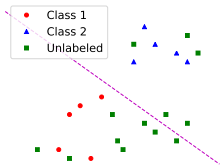
We consider learning tasks where symbolic knowledge is provided to connect the different outputs of a neural network. This knowledge takes the form of a constraint (or sentence) in Boolean logic. It can be as simple as an exactly-one constraint for one-hot output encodings, or as complex as a structured output prediction constraint for intricate combinatorial objects such as rankings, subgraphs, and paths. Our goal is to augment neural network with the ability to learn how to make predictions subject to these constraints, and use the symbolic knowledge to improve its performance.

Most neuro-symbolic approaches aim to simulate or learn symbolic reasoning in an end-to-end deep neural network, or capture symbolic knowledge in a vector-space embedding. This choice is partly motivated by the need for smooth *differentiable* models; adding symbolic reasoning code (e.g., SAT solvers) to a deep learning pipeline destroys this property. Unfortunately, while making reasoning differentiable, the precise logical meaning of the knowledge is often lost. In this paper, we take a distinctly different approach. We first define a differentiable *semantic loss* function that captures how well the outputs of a neural network match a given constraint. This function precisely captures the *meaning* of the constraint, and is independent of its *syntax*.

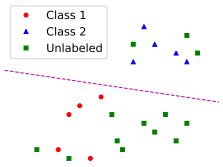
Next, we show how this semantic loss gives *significant practical improvements* in semi-supervised classification. The semantic loss defined over the exactly-one constraint in this setting permits us to obtain a learning signal from vast amounts of unlabeled data. The key idea is that the semantic loss

*The long version of this paper [39] (available at `https://arxiv.org/abs/1711.11157`) derives the semantic loss function from first principles, and discusses more experimental details and related work.

NIPS 2017 Workshop on Learning with Limited Labeled Data: Weak Supervision and Beyond.



(a) Trained without semantic loss



(b) Trained with semantic loss

Figure 1: Binary classification toy example: a linear classifier without and with semantic loss.

helps us improve how consistently we are able to classify the unlabeled data. This simple addition to the loss function of standard deep learning architectures yields (near-)state-of-the-art performance in semi-supervised classification on MNIST, FASHION and CIFAR-10 datasets. Our final set of experiments study the benefits of the semantic loss function for complex structured output learning tasks, such as preference learning and path prediction in a graph [2, 4, 8, 15] to show how it can be generalized onto more difficult problems. By capturing the structure of the output space with logical constraints, and minimizing the semantic loss for this constraint during learning, we are able to learn networks that are much more likely to correctly predict structured objects.

2 Semantic Loss and its Application in Semi-Supervised Classification

The goal of a semantic loss function is to bridge the gap between the continuous world of neural networks, and the symbolic world of propositional logic. The semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$ is a function of a sentence α in propositional logic, defined over variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, and a vector of probabilities p for the same variables $\mathbf{X}$. Element p_i denotes the predicted probability of variable X_i, and corresponds to a single output of the neural net. For example, the semantic loss between an exactly-one constraint α and a neural net output vector p captures how close the prediction p is to having exactly one output set to true (1), and all false (0), regardless of which output is correct.

We desire our semantic loss $\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p})$ to be proportional to the negative log-probability of α being satisfied when sampling values according to p. More formally,

$$\mathrm{L}^{\mathrm{s}}(\alpha, \mathsf{p}) \propto -\log \sum_{\mathbf{x} \models \alpha} \prod_{i : \mathbf{x} \models X_i} \mathsf{p}_i \prod_{i : \mathbf{x} \models \neg X_i} (1 - \mathsf{p}_i). \tag{1}$$

Here, $\mathbf{x} \models \alpha$ means that assignment $\mathbf{x}$ to variables $\mathbf{X}$ satisfies sentence α, and $\mathbf{x} \models X_i$ means that X is set to true in world $\mathbf{x}$. Intuitively, this is the self-information of obtaining an assignment that satisfies the constraint. This equation is derived from first principles in the long version of this paper.

Illustrative Example To illustrate the benefit of semantic loss in the semi-supervised setting, we begin our discussion with a small toy example. Consider a binary classification task as depicted in Figure 1. Ignoring the unlabeled examples, a simple linear classifier learns to distinguish the two classes by separating the labeled examples in Figure 1a. However, the unlabeled examples are also informative, as they must carry some properties that give them a particular label. This is the crux of semantic loss: a model must confidently assign a consistent class even to unlabeled data. Encouraging the model to do so results in a more accurate decision boundary, as illustrated in Figure 1b. Next, we further explore this idea and apply it to real-world image classification tasks.

2.1 Algorithm

Our proposed method intends to be generally applicable and compatible with any feedforward neural network. The semantic loss is simply another regularization term that can directly be plugged into an existing loss function. More specifically, for some weight w, the new overall loss becomes

$$\text{existing loss} + w \cdot \text{semantic loss}.$$

When the constraint over the output space is simple (for example, there is a small number of solutions $\mathbf{x} \models \alpha$), the semantic loss can be directly computed with Equation 1. Concretely, for the

Table 1: MNIST. Previously reported test accuracies followed by semantic loss results (± stddev)

Accuracy % with # of used labels	100	1000	ALL
AtlasRBF [29]	91.9 (± 0.95)	96.32 (± 0.12)	98.69
Deep Generative [17]	96.67(± 0.14)	97.60(± 0.02)	99.04
Virtual Adversarial [22]	97.67	98.64	99.36
Ladder Net [30]	**98.94** (±0.37)	**99.16** (±0.08)	**99.43** (± 0.02)
Baseline: MLP, Gaussian Noise	78.46 (±1.94)	94.26 (±0.31)	99.34 (±0.08)
Baseline: Self-Training	72.55 (±4.21)	87.43 (±3.07)	99.34 (±0.08)
MLP with Semantic Loss	98.38 (±0.51)	98.78 (±0.17)	99.36 (±0.02)

exactly-one constraint used in n-class classification, the semantic loss reduces to

$$\mathrm{L}^{\mathrm{s}}(\text{exactly-one}, \mathsf{p}) \propto -\log \sum_{i=0}^{n-1} \mathsf{p}_i \prod_{j=0, j\neq i}^{n-1} (1 - \mathsf{p}_j),$$

where he values p_i denote the probability of class i as predicted by the neural net. The semantic loss for the exactly-one constraint is efficient and causes no noticeable overhead in our experiments.

In general, for arbitrary constraints α, the semantic loss is not efficient to compute using Equation 1, and more advanced automated reasoning is required. Section 3 discusses this issue in more detail.

2.2 Experimental Evaluation

We evaluate semantic loss in the semi-supervised setting by comparing it with several competitive models. We add semantic loss to the same base models used in ladder nets [30], which currently achieve state-of-the-art results on semi-supervised MNIST and CIFAR-10 [18]. Specifically, the MNIST base model is a fully-connected multilayer perceptron (MLP), with layers of size 784-1000-500-250-250-250-10. On CIFAR-10, it is a 10-layer convolutional neural network (CNN) with 3-by-3 padded filters. After every 3 layers, features are subject to a 2-by-2 max-pool layer with strides of 2. We refer to the long version of this paper for further details.

For all semi-supervised experiments, we use the standard 10,000 held-out test examples provided in the original datasets and randomly pick 10,000 training examples as validation set. For values of N that depend on the experiment, we retain N randomly chosen labeled examples from the training set, and remove labels from the rest. We balance classes in the labeled samples to ensure no particular class is over-represented.

MNIST The permutation invariant MNIST classification task is commonly used as a test-bed for general semi-supervised learning algorithms. We run experiments for 10 epochs, with a batch size of 10 labeled and 10 unlabeled examples. Experiments are repeated 10 times with different random seeds. Table 1 compares semantic loss to two baselines and state-of-the-art results from the literature. The first baseline is a purely supervised MLP, which makes no use of unlabeled data. The second is the classic self-training method for semi-supervised learning, which operates as follows. After every 1000 iterations, the unlabeled examples that are predicted by the MLP to have more than 95% probability of belonging to a single class, are assigned a psuedo-label and become labeled data.

When given 100 labeled examples ($N = 100$), MLP with semantic loss gains around 20% improvement over the purely supervised baseline. The improvement is even larger (25%) compared to self-training. Considering *the only change is an additional loss term*, this result is very encouraging. Compared to the state of the art, ladder nets slightly outperform semantic loss by 0.5% accuracy. This difference may be an artifact of the excessive tuning of architectures, hyper-parameters and learning rates that the MNIST dataset has been subject to.

FASHION The FASHION [38] dataset consists of Zalando's article images, aiming to serve as a more challenging drop-in replacement for MNIST. Arguably, it has not been overused and requires more advanced techniques to achieve good performance. As in the previous experiment, we run our method for 10 epochs, whereas ladder nets need 100 epochs to converge. Again, experiments are repeated 10 times and Table 2 reports the classification accuracy and its standard deviation (except for $N =$ all where it is close to 0 and omitted for space).

Table 2: FASHION. Test accuracy comparison between MLP with semantic loss and ladder nets.

Accuracy % with # of used labels	100	500	1000	ALL
Ladder Net [30]	81.46 ($\pm$0.64)	85.18 ($\pm$0.27)	86.48 ($\pm$ 0.15)	**90.46**
Baseline: MLP, Gaussian Noise	69.45 ($\pm$2.03)	78.12 ($\pm$1.41)	80.94 ($\pm$0.84)	89.87
MLP with Semantic Loss	**86.74** ($\pm$0.71)	**89.49** ($\pm$0.24)	**89.67** ($\pm$0.09)	89.81

Table 3: CIFAR. Test accuracy comparison between CNN with semantic loss and ladder nets.

Accuracy % with # of used labels	4000	ALL
CNN Baseline in Ladder Net	76.67 ($\pm$ 0.61)	90.73
Ladder Net [30]	79.60 ($\pm$0.47)	
Baseline: CNN, Whitening, Cropping	77.13	**90.96**
CNN with Semantic Loss	**81.79**	90.92

Experiments show that utilizing semantic loss results in a very large 17% improvement over the baseline when only 100 labels are provided. Moreover, our method compares favorably to ladder nets, except when the setting degrades to be fully supervised. Note that our method already nearly reaches its maximum accuracy with 500 labeled examples, which is only 1% of the training dataset.

CIFAR-10 To show the general applicability of semantic loss, we evaluate it on CIFAR-10. This dataset consisting of 32-by-32 RGB images in 10 classes. A simple MLP would not have enough representation power to capture the huge variance across objects within the same class. To cope with this spike in difficulty, we switch our underlying model to a 10-layer CNN as described earlier. We use a batch size of 100 samples of which half are unlabeled. Experiments are run for 100 epochs. However, due to our limited computational resources, we report on a single trial. Note that we make slight modifications to the underlying model used in ladder nets to reproduce similar baseline performance. Please refer to the long version of this paper for the details of this experimental setup.

As shown in Table 3, our method compares favorably to ladder nets. However, due to the slight difference in performance between the supervised base models, a direct comparison would be methodologically flawed. Instead, we compare the net improvements over baselines. In terms of this measure, our method scores a gain of 4.66% whereas ladder nets gain 2.93%.

2.3 Discussion

Overall, the experiments so far have demonstrated the competitiveness and general applicability of our proposed method on semi-supervised learning tasks. It surpassed the previous state of the art (i.e. ladder nets) on FASHION and CIFAR-10, while being close on MNIST. Considering the simplicity of our method, such results are encouraging. Indeed, a key advantage of semantic loss is that it only requires a simple additional loss term. Without changing the network architecture itself, we incur almost no computational overhead. Conversely, this property makes our method sensitive to the underlying model's performance. Without the underlying predictive power of a strong supervised learning model, we do not expect to see the same benefits we observe here. Recently, we became aware that [22] extended their work to CIFAR-10 and achieved state-of-the-art results [23], surpassing our performance by 5%. In future work, we plan to investigate whether applying semantic loss on their architecture would yield an even stronger performance. The objective of semantic loss to increase the confidence of predictions on unlabeled data is in common with information-theoretic approaches to semi-supervised learning [13, 14, 22]. A key difference between semantic loss and information-theoretic losses is that semantic loss generalizes to arbitrary output constraints [39].

3 Learning with Complex Constraints

While much of current machine learning research is focused on problems such as multi-class classification, there remain a multitude of difficult problems involving highly constrained output domains. Because semantic loss is defined by a Boolean formula, it can be used on any output domain that can be fully described in this manner. Here, we develop a framework for tractable semantic loss on highly complex constraints, and evaluate it on some difficult examples.

Table 4: Grid shortest path test accuracy.

Test accuracy %	Coherent	Incoherent
5-layer MLP	5.62	**85.91**
With semantic loss	**28.51**	83.14

Table 5: Peference prediction test accuracy.

Test accuracy %	Coherent	Incoherent
3-layer MLP	0.01	**75.74**
With semantic loss	**13.59**	72.43

3.1 A Tractable Semantic Loss

Our goal here is to develop a method for computing both the semantic loss and its gradient in a tractable manner. Examining Equation 1, we see that the right-hand side is a well-known automated reasoning task called weighted model counting (WMC) [3, 34]. A key property of WMC is that its partial derivatives can be computed in terms of other, slightly modified WMCs. Furthermore, we know of circuit languages that compute weighted model counts, and that are amenable to back-propagation [5]. We use the language and circuit compilation techniques described in [6] to build a Boolean circuit representing our semantic loss. Due to certain properties of this circuit form, we can use it to compute both the values and the gradients of the semantic loss in time linear in the size of the circuit [7]. Once we have constructed this function, we can add it to our standard loss function as described in Section 2.1.

3.2 Experimental Evaluation: Grids & Preference Learning

Our ambition when evaluating semantic loss' performance on complex constraints is not to achieve state-of-the-art performance on any particular problem, but rather to highlight its effect. To this end, we evaluate our method on problems with a difficult output space, where the model could no longer fit directly from data, and purposefully use simple MLPs for evaluation.

We begin with the problem of finding the shortest path in a graph. Specifically, we use a 4-by-4 grid $G = (V, E)$, and randomly remove some edges for each example to increase difficulty. Formally, our input is a binary vector of length $|V| + |E|$, with the first $|V|$ variables indicating sources and destinations, and the next $|E|$ which edges are removed. Similarly, the label for a given example is a binary vector of length $|E|$ indicating which edges are in the shortest path. Finally, we require through our semantic loss that the output form a valid simple path between the desired source and destination. This is achieved by using the appropriate logical constraint, as specified in [27].

We also examine the problem of predicting a complete order of user preferences. That is, for a given set of user features, we would like to predict how the user would rank their preference over a fixed set of items. We encode a preference ordering over n items as a flattened binary matrix $\{X_{ij}\}$, where for each $i, j \in \{1, \ldots, n\}$, it denotes that item i is at position j [4]. Clearly most outputs do not represent a total ordering, so we use semantic loss to enforce this. We predict rankings over 4 items, with data taken from PREFLIB's sushi dataset [20].

Tables 4 and 5 report two different accuracies that illustrate the effect of semantic loss: "Coherent" indicates the percentage of examples for which the classifier gets the entire configuration right, while "Incoherent" measures the percentage of individually correct binary labels, which as a whole may not constitute a valid path/ranking at all. In the case of incoherent accuracy, semantic loss has little effect, and in fact slightly reduces the accuracy as it combats the standard sigmoid cross entropy. In regard to coherent accuracy, however, the semantic loss has a very large effect in guiding the network to jointly output structured objects, rather than optimizing each binary output individually. Remarkably, without semantic loss, the network is only able to output a valid preference ordering on 0.01% of the test examples.

4 Conclusions

Both reasoning and semi-supervised learning are often identified as key challenges for deep learning going forward. In this paper, we developed a principled way of combining automated reasoning for propositional logic with existing deep learning architectures. Moreover, we showed that our semantic loss function provides significant benefits during semi-supervised classification, as well as deep structured prediction for highly complex output spaces.

Acknowledgements

This research was conducted while Zilu Zhang was a visiting student from Peking University. The authors thank Arthur Choi and Yujia Shen for helpful discussions. This work is partially supported by NSF grants #IIS-1657613, #IIS-1633857 and DARPA XAI grant #N66001-17-2-4032.

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[2] M.-W. Chang, D. Goldwasser, D. Roth, and Y. Tu. Unsupervised constraint driven learning for transliteration discovery. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 299–307. Association for Computational Linguistics, 2009.

[3] M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6):772 – 799, 2008.

[4] A. Choi, G. Van den Broeck, and A. Darwiche. Tractable learning for structured probability spaces: A case study in learning preference distributions. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[5] A. Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, 50(3):280–305, May 2003.

[6] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 819, 2011.

[7] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[8] H. Daumé, J. Langford, and D. Marcu. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

[9] T. Demeester, T. Rocktäschel, and S. Riedel. Lifted rule injection for relation embeddings. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1389–1399, 2016.

[10] M. Diligenti, M. Gori, and C. Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

[11] I. Donadello, L. Serafini, and A. d. Garcez. Logic tensor networks for semantic image interpretation. 2017.

[12] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[13] A. Erkan and Y. Altun. Semi-supervised learning via generalized maximum entropy. In *Thirteenth International Conference on Artificial Intelligence and Statistics*, volume PMLR, pages 209–216, 2010.

[14] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[15] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[16] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. In *ACL*, 2016.

[17] D. P. Kingma, S. Mohamed, D. Jimenez Rezende, and M. Welling. Semi-supervised learning with deep generative models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc., 2014.

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[19] P. Márquez-Neila, M. Salzmann, and P. Fua. Imposing hard constraints on deep networks: Promises and limitations. *arXiv preprint arXiv:1706.02025*, 2017.

[20] N. Mattei and T. Walsh. Preflib: A library of preference data HTTP://PREFLIB.ORG. In *Proceedings of ADT*, 2013.

[21] P. Minervini, T. Demeester, T. Rocktäschel, and S. Riedel. Adversarial sets for regularising neural link predictors. *arXiv preprint arXiv:1707.07596*, 2017.

[22] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. In *ICLR*, 2016.

[23] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *ArXiv e-prints*, 2017.

[24] J. Naradowsky and S. Riedel. Modeling exclusion with a differentiable factor graph constraint. 2017.

[25] A. Neelakantan, B. Roth, and A. McCallum. Compositional vector space models for knowledge base inference. 2015.

[26] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

[27] M. Nishino, N. Yasuda, S. Minato, and M. Nagata. Compiling graph substructures into sentential decision diagrams. In *Proceedings of the Thirty-First Conference on Artificial Intelligence (AAAI)*, 2017.

[28] D. Pathak, P. Krahenbuhl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1796–1804, 2015.

[29] N. Pitelis, C. Russell, and L. Agapito. Semi-supervised learning using an unsupervised atlas. In *Proceedings of ECML-PKDD*, pages 565–580. Springer, 2014.

[30] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.

[31] S. Reed and N. De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

[32] S. Riedel, M. Bosnjak, and T. Rocktäschel. Programming with a differentiable forth interpreter. *CoRR, abs/1605.06640*, 2016.

[33] T. Rocktäschel, S. Singh, and S. Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *HLT-NAACL*, pages 1119–1129, 2015.

[34] T. Sang, P. Beame, and H. A. Kautz. Performing bayesian inference by weighted model counting. In *AAAI*, volume 5, pages 475–481, 2005.

[35] R. Stewart and S. Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, pages 2576–2582, 2017.

[36] M. Wang, Y. Tang, J. Wang, and J. Deng. Premise selection for theorem proving by deep graph embedding. *arXiv preprint arXiv:1709.09994*, 2017.

[37] J. Weston, S. Chopra, and A. Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[38] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[39] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. *CoRR*, abs/1711.11157, Nov. 2017.

Multikernel Gaussian Processes for patient stratification from imaging biomarkers with heterogeneous patterns

Liane S. Canas[1]∗**, Benjamin Yvernault**[1]**, Carole H. Sudre**[1,5]**, M Jorge Cardoso**[1]**,**
John Thornton[2]**, Frederik Barkhof** [1,3]**, Sebastien Ourselin**[1]**, Simon Mead**[4]**, Marc Modat**[1]
[1]Translational Imaging Group, Centre for Medical Image Computing, University College of London, UK
[2]Lysholm Department of Neuroradiology, National Hospital for Neurology and Neurosurgery, London, UK
[3]Institute of Neurology, University College London, UK
[4]MRC Prion Unit, Department of Neurodegenerative Disease UCL Institute of Neurology, London, UK
[5]Dementia Research Centre, UCL Institute of Neurology, London, UK
∗liane.canas.15@ucl.ac.uk

Abstract

Prion diseases are a group of progressive neurodegenerative conditions, which cause cognitive impairment and neurological deficits. The approaches used to study other types of neurodegenerative diseases, such as Alzheimer's disease, are not appropriate to capture the progression of the human form of Prion disease. This is largely due to the heterogeneity of the phenotypes associated with Prion disease. This heterogeneity, combined with the rarity of the disease and thus the limited amount of available data, hampers the ability of state of art models to stratify patients in disease stages accurately.

In this paper, we focus on the development of a novel framework based on Gaussian Process (GP) to stratify subjects with the inherited human form of Prion disease. Our framework tackles the small number of training data as well as the presence of heterogeneity among subjects' biomarkers. This is achieved by firstly using a subject-specific multi-modal feature extraction scheme in order to normalise the data across subjects. Secondly an additive GP classifier is used to correlate subjects with disease severity. Through a simulated dataset we highlight the rationale and added value of our technique before applying it to real data.

1 Introduction

Neurodegenerative diseases (NDD) are characterised by a progressive alteration of the physiology and morphology of the brain, which leads to irreversible functional impairments. In order to characterise the progression of such diseases, researchers rely on biomarkers, extracted for example from wet samples or imaging. Biomarkers are then used, through modelling, for the purpose of diagnosis, prognosis or stratification of subjects. Different approaches have been proposed in the literature to model disease progression, including event-based models [5], mixed effect models [6] or continuous Bayesian framework [4]. Gaussian processes (GP) have also been used [9; 7] as they have the advantage of modelling the interaction between multiple biomarkers [14].

All proposed approaches for disease modelling rely on the hypothesis that biomarkers follow a similar pattern of change across patients. For example, in typical Alzheimer's disease the hippocampus is affected early in the disease, followed by other areas of the brain in a systematic order [5; 14].

The human form of Prion disease, Creutzfeldt-Jakob disease (CJD), a rare NDD, does not seem to follow a singular pattern of disease progression. The clinical diagnosis of CJD is challenging while

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

the patient is alive, due to the heterogeneity of observed phenotypes, particularly in the early stages of the disease. To this date, definite diagnosis of CJD is only possible *post mortem* (brain autopsy). Nonetheless, several approaches have been investigated to improve the sensitivity and specificity of CJD diagnosis during life. MRI images show signal abnormalities such as hyperintensities in FLAIR and DWI images [2; 8]. Being able to diagnose CJD at the early stages of the disease would enable the patients to be involved in clinical trials, which is currently challenging as patients can die in less that 12 months from diagnosis [1]. Researcher groups are focusing on the extremely rare cases of Inherited human form of Prion disease (IPD) as it is possible to follow subjects years before they develop any clinical symptoms. IPD occurs as a result of one of more than 30 mutations in the prion protein gene (*PRNP*), hence conferring highly heterogeneous phenotypes to the disease. Variablity has also been reported among affected individuals within families carrying the same mutation but this is yet to be explained [15; 11].

In this paper, we focus on the development of a novel framework based on GP to stratify subjects with IPD. Our framework tackles the insufficient number of training data and the presence of heterogeneity among subjects' biomarkers. We designed a subject specific biomarkers extraction scheme and model the interaction between biomarkers from different MRI pulse-sequences.

To validate our framework we apply it to synthetic data before applying it to a dataset of IPD.

2 Methods

A non-parametric kernel-based model $\mathcal{M} : y = f(X) + \varepsilon,\ f \sim \mathcal{GP}(\mu_f; \boldsymbol{K}),\ \varepsilon \sim \mathcal{N}(\mu_\varepsilon; \sigma)$ was used to infer the subjects status, y, given a set of imaging biomarkers $\mathbf{X} \in \mathcal{X}$, where $\mathcal{X}$ is the features space.The selected biomarkers include volumetric features extracted from structural Magnetic Resonance (MR) images, intensity-based features from FLAIR scans and mean diffusivity (MD) measures from diffusion weighted imaging (DWI). Our approach follows four main sections: (i) features extraction followed by (ii) features selection and normalisation; (iii) model training — estimation of the hyperparameters associated to the covariance kernel functions defined for the modalities under consideration —, and (iv) stratification of subjects based on the stages of the disease.

2.1 Features extraction, selection and normalisation

2.1.1 Features Extraction

Structural features consist of regional volumetric measurements and are extracted from T1-weighted MRI scans (T1w) using the Geodesical Information Flows [3] algorithm. We regress the impact of confounding effects, such as age and the total intracranial volume, by comparison with a healthy population.

To mimic clinical practice, we extract abnormal signal intensities in the grey matter (GM) from FLAIR images using the Bayesian Model Selection algorithm [12] approach. In order to robustly characterise the signal intensities of each brain region and across all subjects we compute the Mahalanobis distance between the regional GM median and the overall healthy white matter intensity distribution.

The mean diffusivity (MD) is computed from the diffusion tensors estimated from diffusion weighted imaging (DWI). We use the median MD per brain region as biomarker.

2.1.2 Features selection and normalisation

For most NDD, group-wise approaches are used to select the relevant features. This can be done as the features are consistent and homogeneous across subjects and throughout the disease stages. Such methods however cannot be applied to IPD due to the phenotype heterogeneity and the lack of consistent pattern of disease progression. A common approach could be to use Automatic Relevance Determination (ARD) as it aims at optimising different weights to automatically extract relevant features. However, such approach requires a number of sample considerably larger than the number of features, which setup is unlikely to occur with real data of patients with Prion disease.

To overcome this issue, we here hypothese that the disease does not follow a geometrical pattern but instead that imaging biomarkers can become abnormal in any location in the brain. The quantity of abnormality rather than its location is thus to quantify the progression of the disease. To quantify this

abnormality, we extract z-scores against a normative database of healthy controls for all extracted imaging biomarkers. The z-score values are then ranked per modality (structural, FLAIR, diffusion) and only the highest values for each modality are considered for subsequent learning and inference stages.

2.2 Model learning

We assume that the feature relationship across modalities can be modelled as a multi-task paradigm, a contribution of independent functions that explain the biomarkers progression. We implement an Additive GP to perform the stratification of the subjects.

Considering a GP with $\mu_f = 0$ and covariance kernel function *K*, we determine the pattern of the inductive generalisation of the features. We implement a multiclass classification GP based on individualised likelihood factors computed for the target classes defined by $y_i \in \{1, ..., 7\}$ for the subject i. These classes correspond to the seven stages of the disease: 0 – healthy control, 1 – asymptomatic subjects, 2– subjects at clinical onset , 3 to 6 – symptomatic subjects divided in 4 groups according to their clinical scores (MRC Scale) [13]. Due to the number of classes under consideration, f_i is a vector $\mathbf{f}_i = [f_i^1, ..., f_i^7]^T$. The multi-class classification is performed by means of a multinominal probit likelihood, as defined by equation 1, where the auxiliary variable u_i is distributed as $p(u_i) = \mathcal{N}(u_i|0, 1)$ and $\Phi(x)$ denotes the cumulative density function of the standard normal distribution.

$$p(y_i|\mathbf{f}_i) = \mathrm{E}_{p(u_i)} \left\{ \prod_{j=1, j\neq y_i}^{7} \mathbf{\Phi}(u_i + f_i^{y_i} - f_i^j) \right\} \tag{1}$$

Inference is achieved via a nested expectation propagation (EP), which does not require numerical quadratures or sampling to estimate the predictive probabilities, as detailed in [10].

Note that our model also accounts for the individualised pattern of each genetic mutation of IPD. In order to reduce the bias introduced by the high number of genetic mutations, we group the subjects in three clusters according to the rate of disease progression associated with each mutation. This information is included in our model using a fourth kernel matrix, K_c. The matrix K_I, which encodes the imaging biomarker, is obtained by the addition of the three kernel matrices computed individually using the information extracted from the three MR images. The final matrix K considered in the GP model results from the Hadamard product, $K_c \odot K_I$.

3 Experiments

3.1 Synthetic Dataset

To demonstrate the efficacy of the proposed approach as well as its rationale, we create a synthetic dataset. The dataset is derived from three functions designed to simulate the three MRI modalities used in this study. For each one of them, referred to as modality A, B and C in figure 1, we generate 10 different biomarkers.

Biomarkers from the same modality are set to follow a common evolution over time but all deviate differently from the main function. We alter the three main functions to also simulate the three different rate of disease progression previously mentioned. For each individual i we obtain the biomarkers $\tau \in [1, 10]$ corresponding to class y_i as a function $\mathbf{f}_i(\tau) = (f^i_{M_A}(\tau) + \epsilon_i, f^i_{M_B}(\tau) + \epsilon_i, f^i_{M_C}(\tau) + \epsilon_i), \epsilon_i \sim \mathcal{N}(0; 0.25)$. f_{M_A}, f_{M_B} and f_{M_C} correspond respectively to a monotonically increasing sigmoid function, a second order polynomial function and a monotonically decreasing sigmoid function. The number of subjects per class is uniformly distributed. In order to simulate the spatial heterogeneity of features among subjects, we selected randomly between one to three biomarkers τ to deviate from the controls samples. The estimation of y_i, j, $\hat{y}_{i,j}$ requires to find the best hyperparameters for our regression task and for each kernel. The hyperparameters θ are estimated via the maximisation of the marginal likelihood of the model, $p(y|X, \theta)$; i.e., the marginalisation over the kernel parameters is performed by *maximum a posteriori* algorithm (MAP). A 10-fold cross-validation scheme is used.

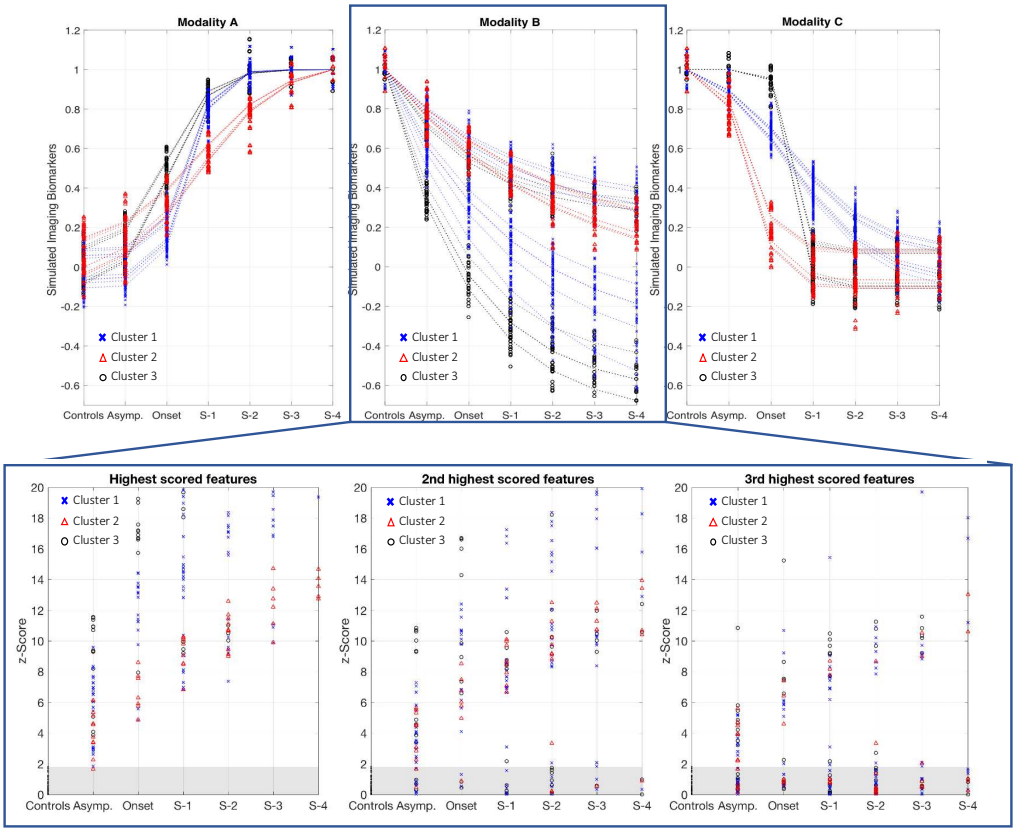


Figure 1: Synthetic dataset. Upper row: Synthetic data generated to simulate the three different modalities. The three different colour define the clusters of subjects according to the rate of progression of IPD. Lower row: Example of the data extracted from modality B after being normalised and ranked by the most significant features for each subject.

The model was used to estimate the probability of a subject to belong to each class. The same experiment is repeated with several sample size $N \in \{100, 500, 1000, 2500\}$ and different GP schemes for comparison. We divided our synthetic samples in two sub-samples: a training set with 75% of subjects, and a testing set with the remaining 25%. To ensure stability and accuracy of the presented results with the synthetic data we use bootstrapping.

Results. Table 1 reports the results of the proposed approach on synthetic data as well as the results obtained using an additive GP using all available features and an additive GP with an ARD scheme for feature selection. The mean percentages of misclassified subjects per class indicate that the proposed method is able to better capture the granularity of the disease stages than the standard approaches. Our method is only outperformed by the GP-ARD when the sample size is very large, as GP-ARD is able to find correlations between subjects even using heterogeneous biomarkers. Note that in this case, the number of samples is considerably higher (2500) than the number of features (30).

3.2 Inherited human form of Prion disease dataset

Using real patient data, we perform the stratification of IPD subjects based on their clinical diagnosis, using both imaging and genetic data. The dataset includes the baseline scans of control, symptomatic and asymptomatic subjects, as well as scans of subjects who have shown clinical symptoms within a year after their scan. The whole sample consists of 25 controls, 29 asymptomatic, 5 close to clinical onset and 30 symptomatic subjects, yielding unbalanced classes. We randomly divided our sample in two sub-samples, training and testing.

Table 1: Percentage of misclassified subjects per class. *N* defines the sample size (number of subjects). The first models corresponds to an additive GP; second model is an additive GP, as implemented before, with ARD scheme for feature selection; finally, third corresponds to the approach proposed in this paper. The bold values correspond to the best approach in each test.

N	Model	Healthy Controls	Asymp. Subjects	Clinical Onset	MRC S. (20-16)	MRC S. (15-11)	MRC S. (10-6)	MRC S. (5-0)	Mean Percentage
	GP	1.38	4.63	5.00	9.63	10.13	13.75	**4.75**	7.04
100	*GP-ARD*	2.25	6.50	6.13	9.75	10.63	12.88	5.38	7.65
	Proposed	**1.00**	**1.13**	**1.38**	**5.13**	**8.50**	**7.88**	5.50	**4.36**
	GP	**0.40**	1.36	2.00	8.00	11.44	10.96	7.12	5.90
500	*GP-ARD*	0.72	2.16	3.28	7.04	10.08	10.08	7.44	5.83
	Proposed	0.56	**0.80**	**1.20**	**3**.28	**6.72**	**9.52**	**5.60**	**3.95**
	GP	**0.40**	1.60	2.67	6.93	10.27	13.33	**6.67**	5.98
1000	*GP-ARD*	0.93	4.40	6.27	8.80	10.67	10.27	6.93	6.90
	Proposed	**0.40**	**1.20**	**1.60**	**3.33**	**6.80**	**9.73**	6.80	**4.27**
	GP	**0.08**	**0.08**	0.48	3.76	11.04	10.48	4.96	4.41
2500	*GP-ARD*	0.24	0.24	0.48	1.92	**5.20**	**6.96**	4.48	**2.79**
	Proposed	**0.08**	0.16	**0.08**	**1.84**	6.24	8.16	**4.40**	2.99

Table 2: Percentage of misclassified subjects per class using clinical data. *N* defines the sample size (number of subjects).

N	Model	Healthy Controls	Asymp. Subjects	Clinical Onset	MRC S. (20-16)	MRC S. (15-11)	MRC S. (10-6)	Mean Percentage
	GP	14.58	21.53	**4.66**	11.41	**2.18**	**0.20**	7.79
89	*GP-ARD*	14.48	16.17	5.36	11.31	2.68	**0.20**	7.17
	Proposed	**12.30**	**14.88**	4.96	**9.23**	2.28	**0.20**	**6.26**

Results. Table 2 reports the results of our proposed approach when applied to IPD patients. The proposed method reduces the misclassification rate per class, when compared with the other methods tested. The classification rate for class of symptomatic subjects was not evaluated due to the absence of individual in IPD dataset with MRC Scale lower than 5. Note that the results reported in Table 2 are deterministic and they do not account for the fuzziness of the classes estimation. The results also support the hypothesis that for rare diseases, like IPD, due to the reduced number of subjects the problem is ill-posed. In this specific case, the results achieved by the GP-ARD are not as accurate as the results obtained by an approach in which the number of features considered are reduced in order to avoid confounding effects raised by the inconsistency of features across subjects.

4 Discussion

We proposed a non-parametric Bayesian approach, which aimed to predict the disease stage of IPD subjects. The approach was designed to (i) account for the lack of biomarker geometrical consistency across subjects, as well as (ii) small sample size. The results obtained with our framework suggest that the model is better able to address efficiently the two aforementioned constraints by implementing a subject-specific biomarkers extraction. In fact the model obtained better results when compared with the current frameworks used in context of NDD, both on synthetic and clinical data. Note also that by using the GP as classification model, the current framework can be easily extended to predict the time to clinical onset of asymptomatic subjects. We thus plan to explore the advantages of GP to model the evolution of the biomarkers over time.

We aim to extend our framework to account for the longitudinal information available from these subjects. This will allow not only the stratification of subjects based on the extracted biomarkers, but also the subjects prognosis in a given time frame. Similarly, other neurodegenerative diseases with consistent biomarkers patterns across subjects can benefit from our approach.

Acknowledgements. This work is supported by the EPSRC-funded UCL Centre for Doctoral Training in Medical Imaging (EP/L016478/1) and the Department of Health's NIHR-funded Biomedical Research Centre at University College London Hospitals.

References

[1] Barry M. Bradford, Pedro Piccardo, James W. Ironside, and Neil A. Mabbott. Human prion diseases and the risk of their transmission during anatomical dissection. *Clinical Anatomy*, 27(6):821–832, 2014. doi: 10.1002/ca.22403.

[2] Federico Caobelli, Milena Cobelli, Claudio Pizzocaro, Marco Pavia, Silvia Magnaldi, and Ugo Paolo Guerra. The role of neuroimaging in evaluating patients affected by Creutzfeldt-Jakob disease: A systematic review of the literature. *Journal of neuroimaging : official journal of the American Society of Neuroimaging*, 6:1–12, 2014. doi: 10.1111/jon.12098.

[3] M. Jorge Cardoso, Marc Modat, Robin Wolz, Andrew Melbourne, David Cash, Daniel Rueckert, and Sebastien Ourselin. Geodesic information flows: Spatially-variant graphs and their application to segmentation and fusion. *IEEE Transactions on Medical Imaging*, 34(9):1976–1988, 2015. doi: 10.1109/TMI.2015.2418298.

[4] Edward Challis, Peter Hurley, Laura Serra, Marco Bozzali, Seb Oliver, and Mara Cercignani. Gaussian process classification of Alzheimer's disease and mild cognitive impairment from resting-state fMRI. *NeuroImage*, 112:232–243, 2015. doi: 10.1016/j.neuroimage.2015.02.037.

[5] Hubert M. Fonteijn, Matt J. Clarkson, and Marc Modat. An event-based disease progression model and its application to familial Alzheimer's disease. In *Information Processing in Medical Imaging*, pages 1–12, Kloster Irsee, Germany, 2011.

[6] Tian Ge, Thomas E. Nichols, Debashis Ghosh, Elizabeth C. Mormino, Jordan W. Smoller, and Mert R. Sabuncu. A kernel machine method for detecting effects of interaction between multidimensional variable sets: An imaging genetics application. *NeuroImage*, 109:505–514, 2015. doi: 10.1016/j.neuroimage.2015.01.029.

[7] Jung Won Hyun, Yimei Li, Chao Huang, Martin Styner, Weili Lin, and Hongtu Zhu. STGP: Spatio-temporal Gaussian process models for longitudinal neuroimaging data. *NeuroImage*, 134:550–562, 2016. doi: 10.1016/j.neuroimage.2016.04.023.

[8] Marc Manix, Piyush Kalakoti, Miriam Henry, Jai Thakur, Richard Menger, Bharat Guthikonda, and Anil Nanda. Creutzfeldt-Jakob disease: updated diagnostic criteria, treatment algorithm, and the utility of brain biopsy. *Neurosurgical Focus*, 39:1–11, 2015. doi: 10.3171/2015.8.FOCUS15328.

[9] Carl Rasmussen and Christopher Williams. *Gaussian processes for machine learning.*, volume 14. 2004. doi: 10.1142/S0129065704001899.

[10] Jaakko Riihim. Nested expectation propagation for Gaussian process classification with a multinomial probit likelihood. *Journal of Machine Learning Research*, 14:75–109, 2013.

[11] Andreas Schroter, Inga Zerr, Karten Henkel, Henriette J. Tschampa, Michael Finkenstaedt, and Sigrid Poser. Magnetic resonance imaging in the clinical diagnosis of Creutzfeldt-Jakob disease. *Journal of American Medical Association*, 57:1751–1757, 2000. doi: 10.1001/archneur.57.12.1751.

[12] Carole Sudre, M. Jorge Cardoso, Willem Bouvy, Geert Biessels, Josephine Barnes, and Sebastien Ourselin. Bayesian model selection for pathological neuroimaging data applied to white matter lesion segmentation. *IEEE Transactions on Medical Imaging*, 34(c):1–1, 2015. doi: 10.1109/TMI.2015.2419072.

[13] Andrew G B Thompson, Jessica Lowe, Zoe Fox, Ana Lukic, Marie Claire Porter, Liz Ford, Michele Gorham, Gosala S. Gopalakrishnan, Peter Rudge, A. Sarah Walker, John Collinge, and Simon Mead. The medical research council prion disease rating scale: A new outcome measure for prion disease therapeutic trials developed and validated using systematic observational studies. *Brain*, 136(4):1116–1127, 2013. ISSN 14602156. doi: 10.1093/brain/awt048.

[14] Jonathan Young, Marc Modat, Manuel J Cardoso, John Ashburner, and Sebastien Ourselin. An oblique approach to prediction of conversion to Alzheimer ' s disease with multikernel Gaussian processes. In *Machine Learning and Interpretation in Neuroimaging*, volume 1, pages 122–128, 2014. doi: 10.1007/978-3-319-45174-9.

[15] Inga Zerr and Sigrid Poser. Clinical diagnosis and differential diagnosis of CJD and vCJD with special emphasis on laboratory tests. *Acta Pathologica, Microbiologica et Immunologica Scandinavica*, 110:88–98, 2002. doi: 10.1034/j.1600-0463.2002.100111.x.

Relaxed Oracles for Semi-Supervised Clustering

Taewan Kim
The University of Texas at Austin
twankim@utexas.edu

Joydeep Ghosh
The University of Texas at Austin
jghosh@utexas.edu

Abstract

Pairwise "same-cluster" queries are one of the most widely used forms of supervision in semi-supervised clustering. However, it is impractical to ask human oracles to answer every query correctly. In this paper, we study the influence of allowing "not-sure" answers from a weak oracle and propose an effective algorithm to handle such uncertainties in query responses. Two realistic weak oracle models are considered where ambiguity in answering depends on the distance between two points. We show that a small query complexity is adequate for effective clustering with high probability by providing better pairs to the weak oracle. Experimental results on synthetic and real data show the effectiveness of our approach in overcoming supervision uncertainties and yielding high quality clusters. [1]

1 Introduction

Clustering is one of the most popular procedures for extracting meaningful insights from unlabeled data. However, clustering is also very challenging for a wide variety of reasons [14]. Finding the optimal solution of even the simple k-means objective is known to be NP-hard [13, 17, 23, 20]. Second, the quality of a clustering algorithm is difficult to evaluate without context. Semi-supervised clustering is one way to overcome these problems by providing a small amount of additional knowledge related to the task [9, 12, 10, 11, 6, 16, 18, 4, 19, 1].

The semi-supervised active clustering (SSAC) framework proposed by Ashtiani et al. [4] combines both margin property and pairwise constraints in the active query setting. A domain expert can help clustering by answering same-cluster queries, which ask whether two samples belong to the same cluster or not. By using an algorithm with two phases, it was shown that the oracle's clustering can be recovered in polynomial time with high probability. However, their formulation of the same-cluster query has only two choices of answers, *yes* or *no*. This might be impractical as a domain expert can also encounter ambiguous situations which are difficult to respond to in a short time.

Our work is motivated by the following question: "Is it possible to perform a clustering task efficiently even with a non-ideal domain expert?". We answer this question by formulating practical weak oracle models and allowing *not-sure* answers to query responses. Our model assumptions considers two reasonable scenarios that may lead to ambiguity in answering a same-cluster query: (i) distance between two points from different clusters is too small, and (ii) distance between two points within the same cluster is too large. We prove that our improved SSAC algorithm can work well under uncertainties if there exists at least one cluster element close enough to the center.

Experimental results on both synthetic and real data show the effective performance of our approach. In particular, our algorithm successfully deals with uncertainties compared to the previous SSAC algorithm by relaxing an oracle's role and providing better pairs for annotation in an active semi-supervision framework.

[1] This paper focuses on the distance-based weak oracle models with additional experimental results. Proofs for theoretical results are available in the extended version. [15]

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

2 Problem Setting

For the purpose of theoretical analysis, the domain of data is assumed to be the Euclidean space $\mathbb{R}^m$, and each center of a clustering $\mathcal{C}$ is defined as a mean of elements in the corresponding cluster, i.e. $\mu_i = \frac{1}{|C_i|}\sum_{x\in C_i} x, \forall i \in [k]$. Then, an optimal solution of the k-means clustering is a center-based clustering.[2] Also, a γ-margin property ensures the existence of an optimal clustering.

Definition 1 (Center-based clustering)**.** *A clustering $\mathcal{C} = \{C_1, \cdots, C_k\}$ is a center-based clustering of $\mathcal{X} \subset \mathbb{R}^m$ with k clusters, if there exists a set of centers $\mu = \{\mu_1, \cdots, \mu_k\} \subset \mathbb{R}^m$ satisfying the following condition with a distance metric $d(x, y)$:*

$$x \in C_i \Leftrightarrow i = \arg\min_j d(x, \mu_j), \ \ \forall x \in \mathcal{X} \text{ and } i \in [k]$$

Definition 2 (γ-margin property - Clusterability)**.** *Let $\mathcal{C}$ be a center-based clustering of $\mathcal{X}$ with clusters $\mathcal{C} = \{C_1, \cdots, C_k\}$ and corresponding centers $\{\mu_1, \cdots, \mu_k\}$. $\mathcal{C}$ satisfies the γ-margin property if the following condition is true:*

$$\gamma d(x, \mu_i) < d(y, \mu_i), \ \ \forall i \in [k], \forall x \in C_i, \forall y \in \mathcal{X} \setminus C_i$$

Problem Formulation We apply the SSAC algorithm on data $\mathcal{X}$, which is supported by a weak oracle that receives weak same-cluster queries. The true clustering $\mathcal{C}$ satisfies the γ-margin property.

Definition 3 (Weak Same-cluster Query)**.** *A weak same-cluster query asks whether two data points $x_1, x_2 \in \mathcal{X}$ belong to the same cluster and receives one of three responses from an oracle.*

$$Q(x_1, x_2) = \begin{cases} 1 & \text{if } x_1, x_2 \text{ are in the same cluster} \\ 0 & \text{if not-sure} \\ -1 & \text{if } x_1, x_2 \text{ are in different clusters} \end{cases}$$

Definition 4 (Weak Pairwise Cluster-assignment Query)**.** *A weak pairwise cluster-assignment query identifies the cluster index of a given data point x by asking k weak same-cluster queries $Q(x, y_i)$, where $y_i \in C_{\pi(i)}$, $i \in [k]$. One of $k+1$ responses is inferred from an oracle with $\mathcal{C} = \{C_1, \cdots, C_k\}$. $\pi(\cdot)$ is a permutation defined on $[k]$ which is determined during the assignment process accordingly.*

$$Q(x) = \begin{cases} t & \text{if } x \in C_{\pi(t)}, t \in [k] \\ 0 & \text{if not-sure} \end{cases}$$

In our framework, the cluster-assignment process uses k weak same-cluster queries and therefore only depends on pairwise information provided by weak oracles. And we denote the radius of a cluster as $r(C_i) \triangleq \max_{x\in C_i} d(x, \mu_i)$ throughout the paper.

Algorithm 1 SSAC for Weak Oracles

Input: Dataset $\mathcal{X}$, an oracle for weak query Q, target number of clusters k, sampling numbers (η, β), and a parameter $\delta \in (0, 1)$.

$\mathcal{C} = \{\}, \ \mathcal{S}_1 = \mathcal{X}, \ r = \lceil k\eta \rceil$
for $i = 1$ to k **do**
  - Phase 1:
    $Z \sim \text{Uniform}(\mathcal{S}_i, r)$ // Draw r samples from S_i
    for $1 \le t \le k$ **do**
      $Z_t = \{x \in Z : Q(x) = t\}$ // Pairwise cluster-assignment query
    end for
    $p = \arg\max_t |Z_t|$, $\mu'_p \triangleq \frac{1}{|Z_p|}\sum_{x\in Z_p} x$
  - Phase 2:
    $\hat{\mathcal{S}}_i = \text{sorted}(\mathcal{S}_i)$ // Increasing order of $d(x, \mu'_p), \ x \in \mathcal{S}_i$
    $r'_i = \text{BinarySearch}(\hat{\mathcal{S}}_i, Z_p, \mu'_p, \beta)$ // Same-cluster query
    $C'_p = \{x \in \mathcal{S}_i : d(x, \mu'_p) < r'_i\}, \ \mathcal{S}_{i+1} = \mathcal{S}_i \setminus C'_p, \ \mathcal{C} = \mathcal{C} \cup \{C'_p\}$
end for

Output: A clustering $\mathcal{C}$ of the set $\mathcal{X}$

[2] In fact, this will hold for all Bregman divergences [8].

3 SSAC with Distance-Weak Oracles

It is reasonable to expect the accuracy of feedback from domain experts to depend on the inherent ambiguities of the given pairs of samples. The cause of "not-sure" answer for the same-cluster query can be investigated based on the distance between the elements in a feature space. Two reasons for having indefinite answers are considered in this work: (i) points from different clusters are too close, and (ii) points within the same cluster are too far. The first situation happens a lot in the real world. For instance, distinguishing wolves from dogs is not an easy task if a Siberian Husky is considered. The second case is also reasonable, because it might be difficult to compare characteristics of two points within the same cluster if they have quite dissimilar features.

Algorithm 2 Unified-Weak BinarySearch

Input: Sorted dataset $\hat{\mathcal{S}}_i = \{x_1, \cdots, x_{|\hat{\mathcal{S}}_i|}\}$ in increasing order of $d(x_j, \mu'_p)$, an oracle for weak query Q, target cluster p, set of assignment-known points Z_p, empirical mean μ'_p, and a sampling number $\beta \leq |Z_p|$.

- Search($x_j \in \hat{\mathcal{S}}_i$)**:**
  Select the point x_1 and use it for same-cluster queries
  if $Q(x_1, x_j) = 1$ **then** Set left bound index as $j + 1$
  else if $Q(x_1, x_j) = -1$ **then** Set right bound index as $j - 1$
  else
    Sample $\beta - 1$ points from Z_p. $B \subseteq Z_p$, $|B| = \beta - 1$
    Weak same-cluster query $Q(x_j, y)$, for all $y \in B$
    if x_j is in cluster C_p **then** Set left bound index as $j + 1$
    else Set right bound index as $j - 1$
    end if
  end if
- Stop: Found the smallest index j^* such that x_{j^*} is not in C_p

Output: $r'_i = d(x_{j^*}, \mu'_p)$

Remark 1. Algorithm 2 can also handle oracles with a random behavior. $\beta = 1$ is sufficient for distance-weak oracles.

Local Distance-Weak Oracle We define the first weak-oracle model sensitive to distance, a local distance-weak oracle, in a formal way to include two vague situations described before. These confusing cases for local distance-weak oracle are visually depicted in Figure 1 for better explanation.

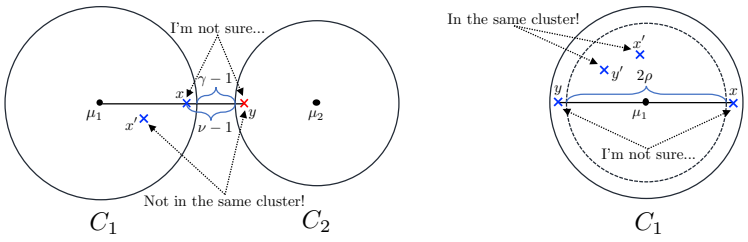


Figure 1: Visual representation of two *not-sure* cases for the local distance-weak oracle. (*Left*) Two points from the different clusters are too close. (*Right*) Two points from the same clusters are too far.

Definition 5 (Local Distance-Weak Oracle)**.** *An oracle having a clustering $\mathcal{C} = \{C_1, \cdots, C_k\}$ for data $\mathcal{X}$ is said to be (ν, ρ) local distance-weak with parameters $\nu \geq 1$ and $\rho \in (0, 1]$, if $Q(x, y) = 0$ for any given two points $x, y \in \mathcal{X}$ satisfying one of the following conditions:*

(a) $d(x, y) < (\nu - 1) \min\{d(x, \mu_i), d(y, \mu_j)\}$, *where* $x \in C_i, y \in C_j, i \neq j$

(b) $d(x, y) > 2\rho r(C_i)$, *where* $x, y \in C_i$

One way to overcome the uncertainty is to provide at least one good point in a query, i.e. *better pairs*. If one of the points x and y for the query $Q(x, y)$ is close enough to the center of a cluster, a local distance-weak oracle does not get confused in answering. This situation is realistic because one

representative data sample of a cluster might be a good baseline when comparing to other elements. Theorem 1 is founded on this intuition, and we show that our modified version of SSAC will succeed if at least one representative sample per cluster is suitable for the weak oracle.

Theorem 1. *If a cluster C_i contains at least one point $x^* \in C_i$ satisfying $d(x^*, \mu_i) < c_{local} \cdot r(C_i)$ for all $i \in [k]$, then combination of Algorithm 1 and 2 outputs the oracle's clustering $\mathcal{C}$ with probability at least $1 - \delta$ by asking weak same-cluster queries to a (ν, ρ) local distance-weak oracle.* $(c_{local} = \min\{2\rho - 1, \gamma - \nu + 1\} - 2\epsilon$, *where* $\epsilon \leq \frac{\gamma - 1}{2})$

Sketch of Proof. We first show the effect of a point close to the center on weak queries. Then the possibility of having a close empirical mean is provided by defining *good sets* and calculating data-driven probability of failure from it. Last, an assignment-known point is identified to remove the uncertainty of same-cluster queries used in the binary search step.

Global Distance-Weak Oracle A global distance-weak oracle fails to answer depending on the distance of each point to its respective cluster center. In this case, both elements x and y should be in the covered range of an oracle if they don't belong to the same cluster.

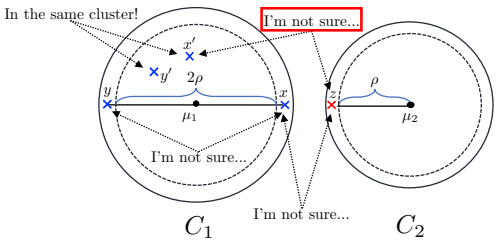


Figure 2: Visual representation of two *not-sure* cases for the global distance-weak oracle. The red box indicates the difference with the local distance-weak oracle.

Definition 6 (Global Distance-Weak Oracle)**.** *An oracle having a clustering $\mathcal{C} = \{C_1, \cdots, C_k\}$ for data $\mathcal{X}$ is said to be ρ global distance-weak with parameter $\rho \in (0, 1]$, if $Q(x, y) = 0$ for any given two points $x, y \in \mathcal{X}$ satisfying one of the following conditions:*

(a) $d(x, \mu_i) > \rho r(C_i)$ *or* $d(y, \mu_j) > \rho r(C_j)$, *where* $x \in C_i, y \in C_j, i \neq j$

(b) $d(x, y) > 2\rho r(C_i)$, *where* $x, y \in C_i$

The problem of a global distance-weak oracle compared to the local distance-weak model is the increased ambiguity in distinguishing elements from different clusters. Nevertheless, once we get a good estimate of the center, better pairs with one good point can be still found to support the oracle in answering same-cluster queries.

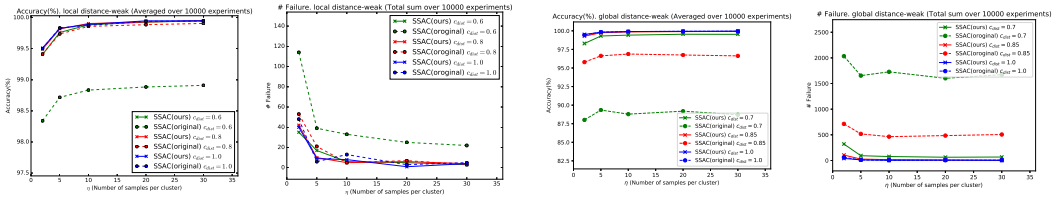
Theorem 2. *If a cluster C_i contains at least one point $x^* \in C_i$ satisfying $d(x^*, \mu_i) < c_{global} \cdot r(C_i)$ for all $i \in [k]$, then combination of Algorithm 1 and 2 outputs the oracle's clustering $\mathcal{C}$ with probability at least $1 - \delta$, by asking weak same-cluster queries to a ρ global distance-weak oracle.* $(c_{global} = 2\rho - 1 - 2\epsilon$, *where* $\epsilon \leq \frac{\gamma - 1}{2})$

4 Experimental Results

Synthetic Data Points of each cluster are generated from isotropic Gaussian distribution. We assume that there exists a ground truth oracle's clustering, and the goal is to recover it where labels are partially provided via weak same-cluster queries. For visual representation, 2-dimensional data points are considered, and other parameters are set to $n = 600$ (number of points), $k = 3$ (number of clusters), and $\sigma_{std} = 2.0$. Data points satisfy γ-margin property with condition $\gamma_{\min} \leq \gamma \leq \gamma_{\max}$. To focus on scenarios with narrow margins, $\gamma_{\min} = 1.0$ and $\gamma_{\max} = 1.1$ are chosen.

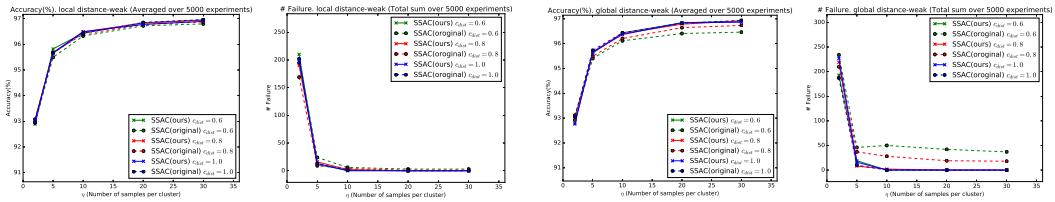
MNIST γ-margin property is difficult to evaluate and satisfy in real world data as a *good* representation or an embedding space is not given. Therefore, we assumed that the oracle has a 2-dimensional embedding space equivalent to the one generated by t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [22]. We used digits 0, 6, and 8 in the subset of MNIST dataset for similarity.[3]

[3]Sample MNIST (2500 points) is from the t-SNE code. `https://lvdmaaten.github.io/tsne/`



(a) $Accuracy$ (%) (b) # $Failure$ (c) $Accuracy$ (%) (d) # $Failure$

Figure 3: Synthetic data. (a),(b): Local distance-weak oracle, $c_{dist} \in \{0.6, 0.8, 1.0\}$. (c),(d): Global distance-weak oracle, $c_{dist} \in \{0.7, 0.85, 1.0\}$. *x-axis*: $\eta \in \{2, 5, 10, 20, 30\}$ (Number of samples)



(a) $Accuracy$ (%) (b) # $Failure$ (c) $Accuracy$ (%) (d) # $Failure$

Figure 4: MNIST. (a),(b): Local distance-weak oracle, $c_{dist} \in \{0.6, 0.8, 1.0\}$. (c),(d): Global distance-weak oracle, $c_{dist} \in \{0.6, 0.8, 1.0\}$. *x-axis*: $\eta \in \{2, 5, 10, 20, 30\}$ (Number of samples)

Evaluation Each round of the evaluation is composed of experiments with different parameter settings on (η, c_{dist}). Parameters for the distance-weak oracles, ρ and ν, are controlled by c_{dist} in the experiments: $\rho = c_{dist}$ and $\nu = \max(1, \gamma) + 2 \cdot (1 - c_{dist})$. β is fixed as 1 since we are only considering distance-weak oracles. η and c_{dist} are varied in each round, and the task is repeated 5000 (MNIST) and 10000 (Synthetic) times. Two evaluation metrics are considered: $Accuracy$ is the ratio of correctly recovered data points averaged over n points, and $\#Failure$ is the total number of failures occurred at cluster-assignments. The best permutation for the cluster labels is investigated based on the distances between estimated centers and true centers for the evaluation. To compare the performance of our improved SSAC, the original one [4] receives random answers, $Q(x, y) = \pm 1$ with probability 0.5, whenever an oracle encounters the case of not-sure. Also, pairs used in the binary search steps are randomly selected from the cluster-known points.

Results An accuracy improves as η increases, and this shows the importance of enough number of samples to succeed in clustering with weak oracles. In fact, even small number of samples are sufficient in practice. Failures of the SSAC algorithm can happen as it is a probabilistic algorithm. When η is really small, the possibility of failure increases as we have only few chances to ask cluster-assignment queries. For example, if $\eta = 2$, only $r = \lceil k\eta \rceil = 6$ points are sampled. Then, if all 6 cluster-assignment queries fail, Phase 1 fails which leads to the recovery of less than k clusters. However, such situations rarely occur if η is large enough.

Results in Figure 3 and 4 show that our improved algorithm (solid lines) outperforms the vanilla SSAC (dashed lines) by allowing not-sure query responses to relax oracles. Especially, results on synthetic data clearly prove the effectiveness of providing better pairs to weak oracles in binary search steps. Our algorithm is robust against the different level of distance weakness. Also, empirical results on MNIST further supports the practicality of our algorithm and weak models.[4]

5 Conclusion and Future Work

This paper presents an approach for utilizing weak oracles in clustering. Specifically, we suggest two realistic types of domain experts who can provide an answer "not-sure" for the same-cluster query. For each model, probabilistic guarantee on discovering the oracle's clustering is provided based on our improved algorithm. In particular, a single element close enough to the cluster center mitigates ambiguous supervision by providing better pairs to an oracle. One interesting future direction is to accommodate embedding learning methods for the real-world clustering tasks.

[4]The source code is available online. `https://github.com/twankim/weaksemi`

References

[1] Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. Approximate clustering with same-cluster queries. *arXiv preprint arXiv:1704.01862*, 2017.

[2] Hassan Ashtiani and Shai Ben-David. Representation learning for clustering: a statistical framework. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 82–91. AUAI Press, 2015.

[3] Hassan Ashtiani and Ali Ghodsi. A dimension-independent generalization bound for kernel supervised principal component analysis. In *Proceedings of The 1st International Workshop on "Feature Extraction: Modern Questions and Challenges", NIPS*, pages 19–29, 2015.

[4] Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. Clustering with same-cluster queries. In *Advances In Neural Information Processing Systems*, pages 3216–3224, 2016.

[5] Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1):49–54, 2012.

[6] Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. In *International Conference on Algorithmic Learning Theory*, pages 316–328. Springer, 2008.

[7] Maria Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM Journal on Computing*, 45(1):102–155, 2016.

[8] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of machine learning research*, 6(Oct):1705–1749, 2005.

[9] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *Proceedings of 19th International Conference on Machine Learning*. Citeseer, 2002.

[10] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 333–344. SIAM, 2004.

[11] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM, 2004.

[12] David Cohn, Rich Caruana, and Andrew McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 4(1):17–32, 2003.

[13] Ian Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 138–149. SIAM, 2005.

[14] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[15] Taewan Kim and Joydeep Ghosh. Semi-supervised active clustering with weak oracles. *arXiv preprint arXiv:1709.03202*, 2017.

[16] Brian Kulis, Sugato Basu, Inderjit Dhillon, and Raymond Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009.

[17] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. The planar k-means problem is np-hard. In *International Workshop on Algorithms and Computation*, pages 274–285. Springer, 2009.

[18] Arya Mazumdar and Barna Saha. Clustering via crowdsourcing. *arXiv preprint arXiv:1604.01839*, 2016.

[19] Arya Mazumdar and Barna Saha. Query complexity of clustering with side information. In *Advances In Neural Information Processing Systems*, 2017.

[20] Lev Reyzin. Data stability in clustering: A closer look. In *International Conference on Algorithmic Learning Theory*, pages 184–198. Springer, 2012.

[21] Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

[22] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

[23] Andrea Vattani. The hardness of k-means clustering in the plane. *Manuscript, accessible at http://cseweb. ucsd. edu/avattani/papers/kmeans_hardness. pdf*, 617, 2009.

Data-Free Knowledge Distillation for Deep Neural Networks

Raphael Gontijo Lopes * rgl3@gatech.edu

Stefano Fenu * sfenu3@gatech.edu

Thad Starner * thad@gatech.edu

* Georgia Institute of Technology

Abstract

Recent advances in model compression have provided procedures for compressing large neural networks to a fraction of their original size while retaining most if not all of their accuracy. However, all of these approaches rely on access to the original training set, which might not always be possible if the network to be compressed was trained on a very large dataset, or on a dataset whose release poses privacy or safety concerns as may be the case for biometrics tasks. We present a method for data-free knowledge distillation, which is able to compress deep neural networks trained on large-scale datasets to a fraction of their size leveraging only some extra metadata to be provided with a pretrained model release. We also explore different kinds of metadata that can be used with our method, and discuss tradeoffs involved in using each of them.

1 INTRODUCTION

It is widely understood that training larger deep neural networks almost uniformly yields better accuracy on a variety of classification tasks than networks with fewer parameters. This has led to it becoming increasingly commonplace to train networks with incredibly large numbers of parameters, which can make deployment or model-sharing impractical or costly.

However, once a model is trained, a big architecture may not be required to represent the function it has learned [8]. Even without radical architectural changes, experimental approaches to model compression have shown 40x decreases in the memory profile of existing networks without significant loss of accuracy [5, 6]. Therefore it is clear that the problem of model compression is of great practical interest, as it allows one to train huge models in data-centers, then compress them for deployment in embedded devices, where computation and memory are limited.

One issue with existing approaches is that they frequently require access to the original training data. As datasets get larger, their release can become prohibitively expensive. Even when a big dataset is released [1], it usually represents only a small subset of a much larger internal dataset used to train many state of the art models. Additionally, many datasets encounter hurdles to release in the form of privacy or security concerns, as in the case of biometric and sensitive data. This can complicate both data-sharing by the original trainer and data-collection by the model compressor.

This paper aims to explore the following question: "Can we leverage metadata about a network to allow us to compress it effectively without access to its original training data?".

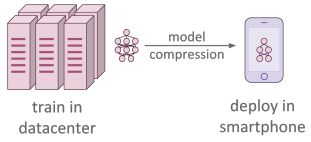


Figure 1: A production pipeline for Deep Learning models: an over-parameterized model is trained to high accuracies using the computation power in a data center, then is compressed for deployment.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

We present a novel neural network compression strategy based on knowledge distillation [7] that leverages summaries of the activations of a network on its training set to compress that network without access to the original data.

2 RELATED WORK

Most compression methods for neural networks fall into three major camps: weight quantization, network pruning, and knowledge distillation. All of these methods work fairly independently of each other, and can be combined in different ways to get 35x-49x reductions in the memory profile of state of the art models [5]

Figure 2: The effect of scaling non-linearities to some temperature: the softened activations provides the student model more information about how the teacher model generalizes.

"Knowledge Distillation" trains a smaller "student" network to copy the actions of a larger "teacher" network. This is typically done either by attempting to train a shallow student network [2, 3] or a thin one [15] to match the outputs of the "teacher" network. A generalized approach, from which we draw heavily, was proposed by Hinton et al. [7]: it relies on modifying the last layer of the teacher network so that, instead of outputting a classification, it outputs activations scaled to some temperature parameters in an attempt to provide more information about how the teacher model generalizes.

The modifications to Hinton's knowledge distillation method which we propose are in part inspired by Intrinsic Replay, introduced by [4] as a way to re-stabilize information a network had already learned after its architecture changed to compensate for new information. In our compression method, instead of relying on the original dataset to guide the optimization of the student network, we attempt to regenerate batches of data based on metadata collected at training time describing the activation of the network. We also depart from [4] in that Intrinsic Replay requires a generative model to recreate the data used to retrain the network, where our method can be applied to any convolutional or fully-connected classifier by simply regenerating samples of the input using the inversion method proposed in [14].

3 METHOD

After training the teacher network on the original dataset, we compute records for the activations of each layer in the network, and save those alongside the model. These can take different forms, and several such forms are discussed in more detail in Section 3.1.

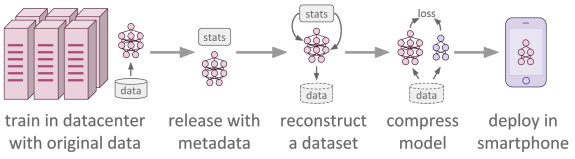


Figure 3: The proposed model compression pipeline: a model is trained in a datacenter and released along with some metadata. Then, another entity uses that metadata to reconstruct a dataset, which is then used to compress the model with Knowledge Distillation. Finally, the model is deployed.

In order to train the student without access to the original data, we attempt to reconstruct the original dataset using only the teacher model and its metadata in the form of precomputed activation records. This is done similarly to [14], attempting to find the set of images whose representation best matches the one given by the network. We pass random gaussian noise as input to the teacher model, then apply gradients to that input noise to minimize the difference between the activation records and those for the noise image. Doing this repeatedly allows us to partially reconstruct the teacher model's view of its original training set.

More formally, given a neural network representation ϕ, and an initial network activation or proxy thereof $\phi_0 = \phi(x_0)$, we want to find the image x^* of width W and height H that:

$$x^* = \underset{x \in R^{HxW}}{\arg\min}\ l(\phi(x), \phi_0)$$

where l is some loss function that compares the image representation $\phi(x)$ to the target one ϕ_0. Unlike the work in [14], we use loss functions that attempt to use precomputed activation statistics for ϕ to find images that maximize the similarity between the neural response of the network for the original dataset and that for the reconstructed dataset instead of trying to maximize some classification probability. These are detailed more clearly in 3.1. Once we have a reconstructed dataset, it can be fed as training input to the student network without any further modification.

3.1 Activation Records

We now present details about the different types of activation records we used as strategies for reconstructing the original dataset. An overview of these can be found in the Appendix (Figure 4).

3.1.1 Top Layer Activation Statistics

The simplest activation records we keep are the means and covariance matrices for each unit of the teacher's classification layer. This is also the layer used in conventional knowledge distillation [7]. We record these statistics according to Equation 1, where L refers to the values in the network right before the final softmax activation, i refers to the i-th unit in that top layer, and T refers to some temperature scaling parameter. In our experiments, we use temperature $T = 8$, just as in [7].

$$\mu_i = \text{Mean}(L_i/T) \quad Ch_i = \text{Chol}(\text{Cov}(L_i/T)) \tag{1}$$

To reconstruct the input, we first sample from these statistics and apply ReLU to it. We then replace the student's topmost non-linearity with ReLU and minimize MSE loss between these two activations by optimizing the input of the network, thus reconstructing an input that recreates the sampled activations.

3.1.2 All Layers Activation Statistics

Unfortunately the method above is underconstrained: there are many different inputs that can lead to the same top-layer activations, which means that our reconstructions aren't able to train the student model to very high accuracies. To better constrain the reconstructions, we store records for all layers instead of just the top-most. The reconstruction procedure is the same as the one above, except that for the hidden layers T is set to 1.

The optimization objective used was the sum of the MSE for each layer, normalized by the number of hidden units in the layer. This normalization is important in order to ensure that the relative importance of each layer is uniform.

However, simply reconstructing with statistics of all layers doesn't preserve inter-layer dynamics of chains of neurons that specialized together to perform some computation. In an attempt to preserve these, we freeze the dropout filters for each batch of reconstructed examples. This way, certain neurons will have their effect zeroed-out in each layer's activations, forcing other neurons to compensate. As can be seen in section 4, this addition was able to make the reconstructed set more visually similar to the original dataset. However, the student accuracies after data-free distillation were slightly worse than without the filters (see Table 1).

3.1.3 Spectral Methods

In order to better capture all of the interactions between layers of the network, we attempt to compress the entirety of the teacher network's activations rather than summarizing it statistically.

Many commonly used signal compression techniques are based around the idea of expanding a signal in some orthonormal basis, under the assumption that most of the information is stored in a small subset of those bases. If we represent the neural network's fully connected layers as a graph, and its activation as a graph signal, we can leverage the formulation of the Graph Fourier transform (F_G) presented in [16] to compute a sparse basis for the activation of the network for a given class.

More formally, if we consider the neural network as the graph G(V, A), where V is a set of vertices corresponding to each neuron in a layer of the network and A is the adjacency matrix corresponding to the weighted connections of those neurons to each other, we can represent the network activation s as a real-valued graph signal [16] on that network, which we can write as a vector $s = [s_0, s_1, ..., s_{N-1}]^T \in R$ where each element s_n is indexed by a vertex v_n of the graph. We

can then compress this graph signal by computing its graph Fourier basis and retaining only a small fraction of its largest spectrum coefficients.

In the framework presented in [16], a graph Fourier basis simply corresponds to the Jordan basis of the graph adjacency matrix A: $A = VJV^{-1}$, where $F = V^{-1}$ is the graph Fourier transform matrix, with the frequency content $\hat{s}$ of s given by $\hat{s} = Fs$.

We can then compress the activation of the network by retaining only some fraction C of the spectrum coefficients $\hat{s}$ with the largest magnitude. Reconstructing the original signal is then done by simply inverting the initial Fourier transform matrix and multiplying by a zero-padded matrix of the spectrum coefficients: $\bar{s} = F_G^{-1}(\hat{s_0}, ..., \hat{s_C - 1}, 0, ..., 0)^T$. Given an initial set of spectrum coefficients $\hat{s}$ and the corresponding graph Fourier transform matrix, we can compute the reconstruction loss for our network as the Euclidean distance between the reconstructed network activation $\bar{s}$ and s_i the activation at a given iteration:

$$l = \sum_i (\bar{s} - s_i)^2$$

The accuracy of the reconstructions is dependent on the number of retained spectrum coefficients, and so is less space efficient than the simpler statistical methods used above, but we find even just retaining 10% of the spectrum coefficients yields a high degree of reconstruction accuracy. It is worth noting that it can be expensive to compute the eigendecomposition of large matrices, so we also consider reconstructing the teacher network activations based on spectra for the smaller graphs given by the connections of pairs of network layers instead of just using those of the entire graph.

4 RESULTS

Two datasets were chosen to examine different qualities of the proposed distillation method: MNIST [11], which was used as a proof of concept that the proposed method works and to provide results that can be directly compared to Hinton et al. [7], and CelebA [13], which was used to show that our method scales to large datasets and models.

For MNIST, we distilled a fully connected model and a convolutional model, to show versatility of the method. They were trained for 10 epochs using Adam with a learning rate of 0.001. Any distillation procedures (including our method) were run for 30 epochs on the reconstructed datasets. Every reconstructed input was first initialized to per-pixel $\sim N(0.15, 0.1)$, and optimized using Adam.

For CelebA, we trained a larger convolutional model [10], to show scalability. We used learning rate 0.0001 and input $\sim N(0.44, 0.29)$

We found that these input initializations, pixel means and variances of the training set, worked well for their respective datasets, and we posit that this kind of information could be provided as model metadata as well. Their computation can be done at training time with a running average of all pixels used for training, similar to what methods like Batch Normalization [9] do.

4.1 MNIST - Fully Connected Models

For the experiments with fully connected models, we used the networks described by Hinton et al. [7]. A network comprising of two hidden layers of 1200 units (HINTON-784-1200-1200-10) was used as the teacher model, and was trained using dropout. Another network of two hidden layers of 800 units (HINTON-784-800-800-10) was used as the student. The total number of parameters was reduced by 50%. For each of them, the temperature parameter used was 8, just like Hinton.

First, we trained both teacher and student models directly on MNIST. Then, we replicate the results from [7] by training the student model using knowledge distillation. Refer to Table 1 for results.

4.2 MNIST - Convolutional Models

For the experiments with convolutional models, we used LENET-5 [12] as the teacher model, and a modified version with half the number of convolutional filters per layer (LENET-5-HALF) as the student model. The total number of parameters was reduced by $\sim 50\%$. Refer to Table 1 for results.

4.3 CelebA - Convolutional Models

In order to bring the experiments closer to the biometrics domain, and to show that our method generalizes to a larger task, we evaluate our approach on a model classifying the most balanced attribute in the large scale facial attributes dataset CELEBA [13] using the larger convolutional model ALEXNET [10]. As before, we use a student model ALEXNET-HALF with half the number of filters per convolutional layer.

As a note, we found that the All Layers optimization objective scales poorly with larger convolutional layers, as the covariance matrix grows at a much higher rate. See Table 1 for other methods.

Model	HINTON-1200	HINTON-800	LENET-5	LENET-5-HALF	ALEXNET	ALEXNET-HALF
Dataset	MNIST	MNIST	MNIST	MNIST	CELEBA	CELEBA
Train	96.95%	95.70%	98.91%	98.65%	80.82%	
Hinton [7]		95.74%		98.91%		
Top Stats		68.75%		77.30%		54.12%
All Stats		76.38%		85.61%		
All Stats+Drop		76.23%				
All Spectral		89.41%		90.28%		77.56%
Layer Spectral		91.24%		92.47%		76.94%

Table 1: Test set accuracies of the models and their respective datasets for each procedure. Note that for each column pair, the model on the left is the teacher, and the model on the right is the student.

5 DISCUSSION

With the increasing popularity of deep learning methods requiring exorbitantly large numbers of parameters, it is useful to consider whether there may be better ways to distribute learned models. We have made the case that there may be metadata worth collecting at, or shortly after, training time that may facilitate the compression and distribution of these models.

However, different choices made about such metadata can have different tradeoffs with regards to the resulting compression accuracy and memory profile. The simple statistical methods presented in equation 1 are easy to compute and require little in the way of additional parameters, but suffer from limited compression accuracy even when reducing the overall memory of a profile by only 50%. Methods more similar to traditional image compression strategies (Figure 4d) require the retention of slightly more metadata in the form of vectors of spectral coefficients, yielding more accurate compression but being significantly more computationally expensive.

Additionally, having run a few initial experiments to assess the quality of the metadata and reconstructions we used, we believe to have motivated further work to better understand how well the former preserves information (and/or privacy) and the latter matches the original data manifold. Regardless, there are countless additional options that could be considered for such metadata, and we hope that this work can help spur some discussion into the development of standard deep neural network formats that may allow for their easier distribution.

6 CONCLUSION

We have presented a method for data-free knowledge distillation. We have shown how many different strategies for activation recording can be used to reconstruct the original dataset, which can then be used to train the student network to varying levels of accuracy. We present tradeoffs related to the use of each of these strategies in section 5. We have shown that these activation records, if appended to the release of a pre-trained model, can facilitate its compression even in scenarios where the original training data is not available.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. arXiv preprint arXiv:1609.08675, 2016.

[2] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Advances in neural information processing systems, pages 2654–2662, 2014.

[3] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 535–541. ACM, 2006.

[4] Timothy J Draelos, Nadine E Miner, Christopher C Lamb, Craig M Vineyard, Kristofor D Carlson, Conrad D James, and James B Aimone. Neurogenesis deep learning. arXiv preprint arXiv:1612.03770, 2016.

[5] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. International Conference on Learning Representations (ICLR), 2016.

[6] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems (NIPS), pages 1135–1143, 2015.

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[8] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2):251–257, 1991.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167, 2015.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012.

[11] Yann LeCun. The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/, 1998.

[12] Yann LeCun et al. Lenet-5, convolutional neural networks.

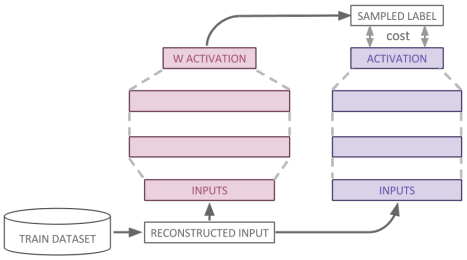
[13] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In Proceedings of International Conference on Computer Vision (ICCV), 2015.

[14] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5188–5196, 2015.

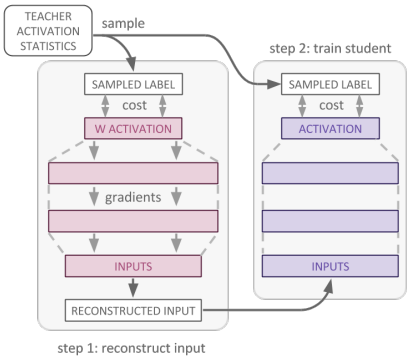
[15] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550, 2014.

[16] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. IEEE transactions on signal processing, 61(7):1644–1656, 2013.

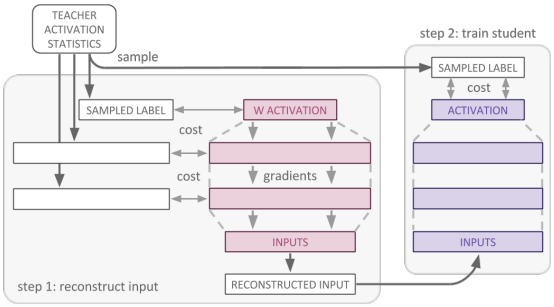
7 APPENDIX



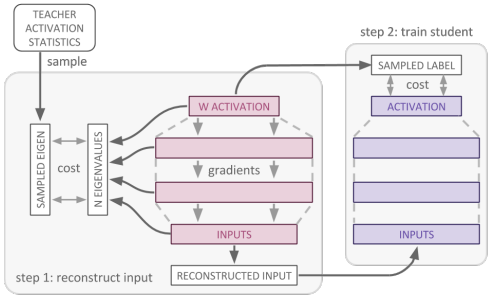
(a) Knowledge distillation



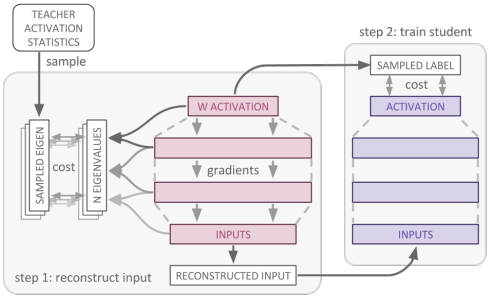
(b) Top Layer Activation Statistics



(c) All layers Activation Statistics



(d) All-Layers Spectral Activation Record



(e) Layer-Pairs Spectral Activation Record

Figure 4: Overview of the different activation records and methods used to reconstruct the original dataset and train the student network. In (a), the student network is trained directly on examples from the original dataset as input, and the teacher's temperature-scaled activations as labels. In (b), we keep activation statistics for the top layer of the teacher network. Then, we sample from those, and optimize the input to the teacher to recreate those activations. That reconstructed input is then used to train the student network. (c) is very similar to (b), but it involves recording statistics, sampling, and recreating activations for all layers of the network. In (d), the optimization objective is to reconstruct the entire activation of the network to correspond to a compressed version of the original network activation. This is intended to better capture inter-layer dynamics and is initially done by expanding the activation into a graph Fourier basis and only retaining a fraction of the spectrum coefficients. In order to compute such an expansion more quickly, we consider applying the same method to each pair of layers separately (e). This is less computationally expensive to compute but requires storing eigenvalues for each pair of layers, which is ultimately less space-efficient.

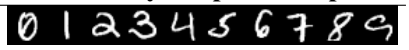
Activation Record	Means	Randomly sampled example
MNIST		
Top Layer Statistics		
All Layers Statistics		
All Layers + Dropout		
Spectral All Layers		
Spectral Layer Pairs		

Table 2: Per-class means and randomly sampled examples of datasets reconstructed using the different activation statistics from HINTON-784-1200-1200-10.

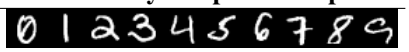
Activation Record	Means	Randomly sampled example
MNIST		
Top Layer Statistics		
All Layers Statistics		
Spectral All Layers		
Spectral Layer Pairs		

Table 3: Per-class means and randomly sampled examples of datasets reconstructed using the different activation statistics from LENET-5.

Simple and Effective Dimensionality Reduction for Word Embeddings

Vikas Raunak
Microsoft India, Hyderabad
viraun@microsoft.com

Abstract

Word embeddings have become the basic building blocks for several natural language processing and information retrieval tasks. Pre-trained word embeddings are used in several downstream applications as well as for constructing representations for sentences, paragraphs and documents. Recently, there has been an emphasis on further improving the pre-trained word vectors through post-processing algorithms. One such area of improvement is the dimensionality reduction of the word embeddings. Reducing the size of word embeddings through dimensionality reduction can improve their utility in memory constrained devices, benefiting several real-world applications. In this work, we present a novel algorithm that effectively combines PCA based dimensionality reduction with a recently proposed post-processing algorithm, to construct word embeddings of lower dimensions. Empirical evaluations on 12 standard word similarity benchmarks show that our algorithm reduces the embedding dimensionality by 50%, while achieving similar or (more often) better performance than the higher dimension embeddings.

1 Introduction

Word embeddings are distributed and dense real-valued representations of words as low dimensional vectors, that geometrically capture the semantic “meaning" of a word, along with several linguistic regularities such as analogy relationships. Such embeddings (e.g. Glove [12], word2vec Skip-Gram [7]) are learned from unlabeled text corpora and have found great use in several natural language processing and information retrieval tasks [9]. Given their widespread utility, recently there has been an emphasis on applying post-processing algorithms on the pre-trained word vectors to further improve their quality. For example, algorithm in [8] tries to inject antonymy and synonymy constraints into vector representations, while [4] tries to refine word vectors by using relational information from semantic lexicons such as WordNet. [3] tries to remove the biases (e.g. gender biases) present in word embeddings and [10] tries to “denoise" word embeddings by strengthening salient information and weakening noise. In particular, the post-processing algorithm in [9] tries to improve word embeddings by projecting the embeddings away from the most dominant directions and considerably improves the embeddings’ performance by making them more discriminative.

Another issue related with word embeddings is their size [6]. For example, loading a word embedding matrix of 2.5M tokens takes up to 6 GB memory (for 300-dimensional vectors, on a 64-bit system). Such large memory requirements impose significant constraints on the practical use of word embeddings, especially on mobile devices where the available memory is often highly restricted. [6] tries to ameliorate this situation by using limited precision representation during word embedding use and training while [1] tries to compress word embeddings using different compression algorithms. Our approach differs from both these works as we directly try to reduce the dimensionality of word embeddings rather than using limited precision representation or compressing individual vector

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

values. In the next section, we explain our algorithm and describe with an example the choices behind its design. The evaluation results are presented subsequently.

2 The Algorithm

In this section, first, the post-processing algorithm from [9] is explained in section 2.1. Our algorithm, along with its motivations is explained in section 2.2. Section 2.3 presents the experimental results on pre-trained Glove [1] and fastText Skip-Gram embeddings [2].

2.1 Post-Processing Word Embeddings

[9] presents a simple post-processing algorithm that renders off-the-shelf word embeddings even stronger, as measured on a number of lexical-level and sentence-level tasks. The algorithm is based on the geometrical observations that the word embeddings (across all representations such as Glove, word2vec etc.) have a large mean vector and most of their energy (after subtracting the mean vector) is located in a subspace of about 8 dimensions. Since, all embeddings share a common mean vector and all embeddings have the same dominating directions, both of which strongly influence the representations **in the same way**, eliminating them renders the embeddings stronger. A formal description of the post-processing algorithm from [9] is presented below:

Algorithm 1 The Post-Processing Algorithm, PPA(X, D)

Input: Word Embedding Matrix X, Threshold Parameter D.

1. Subtract the Mean:
X = X - mean(X).
2. Compute the PCA Components:
u_i = PCA(X), where i = 1,2,...,d.
3. Eliminate the Top D Components: $\forall$ v in X:
$v = v - \sum_{i=1}^{D}(u_i^T \cdot v)u_i$

Output: Post-Processed Word Embedding Matrix X.

end

Figure 1(a) demonstrates the impact of the post-processing algorithm (PPA, with $D = 7$) as observed on Glove embeddings (300-dimensions). It compares the fraction of variance explained by the top 20 principal components of the original and post-processed word vectors respectively (the total sum of explained variances over the 300 principal components is equal to 1.0). In the post-processed word embeddings none of the top principal components are disproportionately dominant in terms of explaining the data, which implies that the post-processed word vectors are not as influenced by the common dominant directions as the original embeddings. This makes the individual word vectors more “discriminative", hence, improving their quality, as validated on several benchmarks in [9].

2.2 Dimensionality Reduction

In this section we explain and present our algorithm that effectively incorporates the post-processing algorithm in the dimensionality reduction procedure. First of all, since the post-processing algorithm demonstrably leads to better word embeddings, it is appropriate that to construct a lower dimensional representation of word embeddings, the dimensionality reduction algorithm (PCA [13]) be applied on the “purified" word embeddings.

Secondly, to motivate our algorithm, consider **Figure 2(b)**. It compares the variance explained by the top 20 principal components for the embeddings constructed by first post-processing the Glove-300D embeddings according to Algorithm 1 (PPA) and then transforming the vectors to 150 dimensions using PCA (labelled as Post-Processing + PCA); against a further post-processed version of the same embeddings (the total sum of explained variances over the 150 principal components is equal to 1.0). We observe that even though PCA has been applied on post-processed embeddings (which

[1] Available at https://github.com/facebookresearch/fastText/.
[2] Available at https://nlp.stanford.edu/projects/glove/.

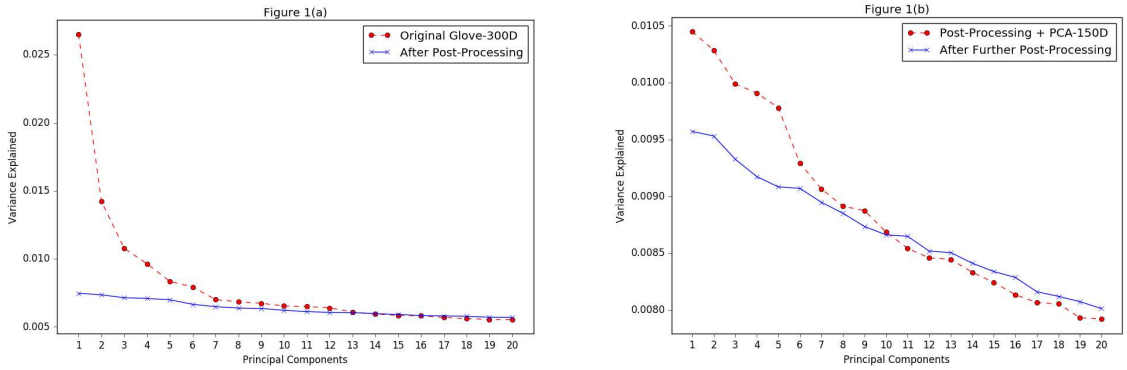


Figure 1: Comparison of (a) the Original and Post-Processed Glove Embeddings (300-Dimensional) in terms of fraction of variance explained by top 20 Principal Components. (b) the Post-Processing + PCA-150D Baseline and Further Post-Processed Glove Embeddings (150-Dimensional) in terms of fraction of variance explained by top 20 Principal Components.

had their dominant directions eliminated), the variance in the resulting embeddings is still explained disproportionately by a few of the top principal components. The re-emergence of this geometrical behaviour implies that further post-processing the lower-dimensional embeddings by projecting the word vectors away from the dominant directions will make the embeddings even stronger.

Finally, from Figures 1(a) and 1(b), it is also evident that the extent to which the top principal components explain the data in the case of the reduced embeddings is not as great as in the case of the original 300 dimensional embeddings. Hence, multiple levels of post-processing at different levels of dimensionality will yield diminishing returns as the influence of common dominant directions decrease on the word embeddings.

Algorithm 2 The Dimensionality Reduction Algorithm

Input: Word Embedding Matrix X, New Dimension N, Threshold Parameter D.

1. Apply the Post-Processing Algorithm:
X = PPA(X, D).
2. Transform X Using PCA:
X = PCA(X).
2. Apply the Post-Processing Algorithm:
X = PPA(X, D).

Output: Word Embedding Matrix of Reduced Dimension N: X.

end

These considerations form the intuition behind our algorithm for constructing lower-dimensional word embeddings, where we apply the post-processing algorithm on either side of a PCA based dimensionality reduction of the word vectors. A formal description of our algorithm is presented as Algorithm 2 (further comments on choosing an appropriate value for parameter D are presented in subsection 2.3.4). Further, in our implementation, subtracting the common mean vector from word embeddings is also done as a pre-processing step before applying PCA on the embedding matrix.

2.3 Evaluation

2.3.1 Word Embeddings

The pre-trained word embeddings (for English only) used for evaluating our algorithms are: Glove embeddings of dimensions 300, 200 and 100, trained on Wikipedia 2014 and Gigaword 5 corpus (400K vocabulary) [12] and fastText embeddings of 300 dimensions trained on Wikipedia using the Skip-Gram model described in [2] (with 2.5M vocabulary).

Table 1: Dataset Statistics and Performance (Spearman's Correlation Coefficient x 100) of Various Algorithms on 300 Dimensional Glove Vectors.

SN	Dataset-Name	WP	Glove-300D	PCA-150D	P+PCA-150D	PCA-150D+P	**Algo-150D**
1	MTurk-771	771	65.01	52.47	**65.59**	63.86	64.58
2	WS-353-SIM	203	66.38	52.69	70.03	70.87	**71.61**
3	MTurk-287	287	63.32	56.56	63.38	**64.62**	63.01
4	VERB-143	144	30.51	28.52	39.04	40.14	**42.24**
5	WS-353-ALL	353	60.54	46.52	66.23	66.85	**67.41**
6	RW-Stanford	2034	41.18	27.46	**43.17**	40.79	42.21
7	MEN-TR-3K	3000	73.75	63.35	75.34	75.37	**75.80**
8	RG-65	65	**76.62**	71.71	73.62	74.27	75.71
9	MC-30	30	70.26	70.03	69.21	72.35	**74.80**
10	SIMLEX-999	999	**37.05**	27.21	36.71	33.81	35.57
11	WS-353-REL	252	57.26	41.82	62.02	60.50	**62.09**
12	YP-130	130	**56.13**	36.72	55.42	50.20	55.91

Table 2: Performance (Spearman's Correlation Coefficient x 100) of Algorithm 2 across different Embedding Types and Dimensions

Serial No.	FastText-300D	Algo-150D	Glove-100D	Algo-50D	Glove-200D	Algo-100D
1	66.89	**67.29**	58.05	**58.85**	62.12	61.99
2	78.12	77.40	60.35	**66.27**	62.91	**68.43**
3	67.93	66.17	61.93	**64.09**	61.99	**63.55**
4	39.73	34.24	30.23	**33.04**	28.45	**36.82**
5	73.69	73.16	52.90	**62.05**	57.42	**65.41**
6	48.66	47.19	36.64	36.64	38.95	**39.80**
7	76.37	76.36	68.09	**70.93**	71.01	**74.44**
8	79.74	**80.95**	69.07	64.56	71.26	**71.53**
9	81.23	**86.41**	62.71	**68.79**	66.56	**69.83**
10	38.03	35.47	29.75	29.13	34.03	**34.19**
11	68.21	**69.96**	49.55	**59.55**	54.48	**61.56**
12	53.33	50.90	45.43	41.95	52.21	49.94

2.3.2 Datasets

We use the standard word similarity benchmarks summarized in [5] for evaluating the word vectors [5]. The datasets have word pairs (WP) that have been assigned similarity rating by humans. While evaluating word vectors, the similarity between the words is calculated by the cosine similarity of their vector representations. Then, Spearman's rank correlation coefficient (Rho) between the ranks produced by using the word vectors and the human rankings is calculated. The reported metric in experiments is Rho x 100. Hence, for better word similarity, the evaluation metric will be higher.

2.3.3 Compared Baselines

To evaluate the performance of our algorithm, we establish some baselines comprising of alternative schemes of combining the post-processing algorithm along with PCA based dimensionality reduction [3]. The baseline algorithms are:

1. **PCA**: Transform the word vectors using PCA.
2. **P+PCA**: Apply the post-processing algorithm and then transform the word vectors using PCA.
3. **PCA+P**: Transform word vectors using PCA and then apply the post-processing algorithm.

[3]Further, in our experiments, generic non-linear dimensionality-reduction/manifold-learning techniques such as kernel PCA, autoencoders performed worse than the baselines, presumably because they do not exploit the unique geometrical property of word embeddings as discussed in subsection 2.2.

These baselines can also be regarded as ablations on our algorithm and can shed light on whether our intuitions in developing the algorithm were correct. In the comparisons ahead, our algorithm is represented as Algo-N (where N is the new dimensionality of the word embeddings). For all evaluations, we use the PCA implementation available in [11].

2.3.4 Experiments and Evaluation Results

First we evaluate our algorithm on the same embeddings against the 3 baselines mentioned above and then, we evaluate our algorithm across word embeddings of different dimensions and types. In all the experiments, the threshold parameter D in the PPA algorithm was set to 7 and the new dimensionality after applying the dimensionality reduction algorithms, N is set to $d/2$.

The appropriate value of parameter D could be inferred from the "Variance Explained" vs "Principle Components" plot as in Figure 1. It could be observed that the top 7 components are disproportionately contributing to the variance. Hence, setting $D = 7$ is appropriate as choosing a lower D will not eliminate the disproportionately dominant directions, while choosing a higher D will eliminate useful discriminative information from the word vectors. Further, we show the results at $N = d/2$ since at this dimension, the reduced embeddings consistently outperform/achieve very close results to the original embeddings. We observed that going below half the dimensions ($N < d/2$) significantly hurts the performance for all of the baselines.

Against Different Baselines: **Table 1** summarizes the results of different baselines on the 12 datasets. As expected from the discussions in Section 2.1, our algorithm achieves the best results on 6 out of 12 datasets when compared across all the columns (the best scores are highlighted in bold). In particular, the 150-dimensional word embeddings constructed using our algorithm performs better than the 300-dimensional embeddings in 7 out of 12 datasets (with an average improvement of 2.74% across the 12 datasets), does significantly better than PCA, PCA+P baselines and beats P+PCA baseline in 8 out of the 12 tasks.

Across Different Embeddings (Both Type & Dimensionality): **Table 2** summarizes the results of applying our algorithm on 300-dimensional fastText embeddings, 100-dimensional Glove embeddings and 200-dimensional Glove embeddings (the better scores are highlighted in bold). In the case of fastText embeddings, the 150-dimensional word vectors constructed using our algorithm gets better performance on 4 out of 12 datasets when compared to the 300-dimensional embeddings. Overall, the 150-dimensional word vectors have a cumulative score of 765.5 against the 771.93 of the 300-dimensional vectors. Hence, overall its performance is quite similar to the 300-dimensional embeddings (with an average performance decline of 0.53% across the 12 datasets). In the case of Glove embeddings of 100 and 200 dimensions, our algorithm leads to significant gains (with average performance improvements of 2.6% and 3% respectively) over the original embeddings and achieves better performance on 8 and 10 datasets respectively.

Another interesting observation from **Table 2** is that the embeddings generated by reducing Glove-200D to 100 dimensions using our algorithm (Algo-100D) significantly outperform the original Glove-100D embeddings (with an average performance improvement of 6% across all the 12 datasets).

Hence, empirical results validate that our algorithm is effective in constructing lower dimensional word embeddings, while maintaining similar or (more often) better performance than the higher dimension embeddings.

3 Conclusions

Our algorithm considerably reduces the size of word embeddings while maintaining or improving upon their utility and has a very simple implementation. This will allow the use of word embeddings on memory-constrained devices by significantly reducing the memory requirements. In future, an interesting area to explore would be the application of compressed and limited precision representations on top of dimensionality reduction to further reduce the size of the word embeddings. We are also working on automating our algorithm, by deriving an algorithm to choose D, N and the levels of post-processing automatically, while optimizing for performance.

References

[1] Martin Andrews. 2016. Compressing Word Embeddings, Springer International Publishing, pages 413–422.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word166 vectors with subword information. Transactions of the Association for Computational Linguistics 5:135–146.

[3] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Advances in Neural Information Processing Systems. pages 4349–4357.

[4] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. arXiv preprint arXiv:1411.4166.

[5] Manaal Faruqui and Chris Dyer. 2014. at wordvectors. org. ACL 2014 page 19.

[6] Shaoshi Ling, Yangqiu Song, and Dan Roth. 2016. Word embeddings with limited memory. In The175 54th Annual Meeting of the Association for Computational Linguistics. page 387.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems. pages 3111–3119.

[8] Nikola Mrkšic, Diarmuid OSéaghdha, Blaise Thomson, Milica Gašic, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counterfitting word vectors to linguistic constraints. In Proceedings of NAACL-HLT. pages 142-148.

[9] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. All-but-the-top: Simple and effective postprocessing for word representations. arXiv preprint arXiv:1702.01417.

[10] Kim Anh Nguyen, Sabine Schulte Walde, and Ngoc Thang Vu. 2016. Neural based noise-filtering from word embeddings. arXiv preprint arXiv: 160.01874.

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, etal. 2011. Scikit-learn: Machine learning in python. Journal of Machine Learning Research 12(Oct):2825–2830.

[12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In EMNLP. Citeseer.

[13] Jonathon Shlens. 2014. A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100.

FlagIt: A System for Minimally Supervised Human Trafficking Indicator Mining

Mayank Kejriwal, Jiayuan Ding, Runqi Shao, Anoop Kumar and Pedro Szekely
Information Sciences Institute, USC Viterbi School of Engineering
4676 Admiralty Way, Ste. 1001
Marina Del Rey, CA 90292
{kejriwal,jiayuand,runqisha,anoopk,pszekely}@isi.edu

Abstract

In this paper, we describe and study the indicator mining problem in the online sex advertising domain. We present an in-development system, FlagIt (Flexible and adaptive generation of Indicators from text), which combines the benefits of both a lightweight expert system and classical semi-supervision (heuristic re-labeling) with recently released state-of-the-art unsupervised text embeddings to tag millions of sentences with indicators that are highly correlated with human trafficking. The FlagIt technology stack is open source. On preliminary evaluations involving five indicators, FlagIt illustrates promising performance compared to several alternatives. The system is being actively developed, refined and integrated into a domain-specific search system used by over 200 law enforcement agencies to combat human trafficking, and is being aggressively extended to mine at least six more indicators with minimal programming effort. FlagIt is a good example of a system that operates in limited label settings, and that requires creative combinations of established machine learning techniques to produce outputs that could be used by real-world non-technical analysts.

1 Introduction

The growth, combined with the ease of sharing information on the Web, has also led to increased illicit activity both on the Open and Dark Web, an egregious example being human trafficking (HT) [7], [14]. The DARPA MEMEX program, which funds research into domain-specific search, has collected hundreds of millions of online sex advertisements, a significant (but unknown) number of which are believed to be sex (and human) *trafficking* instances[1].

Flagging scraped content from webpages in such domains with activity tags or *indicators*, denoted herein as the *indicator mining* problem, has an investigative and socially motivated purpose not just in online human trafficking but several other domains that have also been studied in MEMEX, including patent trolling, counterfeit electronic sales, illegal weapons sales and narcotics[2]. In recent years, some of these 'illicit' Web domains have been intensely studied by social and computer scientists seeking to profile activity on the Dark Web [15], [17]; however, to the best of our knowledge, flagging Web content with tags indicating illicit activity has received considerable less attention.

An indicator is a flag, typically binary, but potentially multi-categorical, that is suggestive of suspicious activity that would warrant investing more resources into investigating the subject of the content being flagged. In the case of the online sex advertisement domain, subjects are not only escorts (usually, the

[1] https://www.darpa.mil/program/memex

[2] The authors have personally encountered a variant of indicator mining in each of these domains, although this paper is limited to human trafficking.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

victims of trafficking) but also perpetrator organizations like massage parlors, procurers, as well as reviewers and clients. While the ideal motivation is to directly flag an online sex trafficking document with a single human trafficking indicator, this is extremely problematic in practice, especially without a proper field level investigation. A more attainable goal in data mining is to instead use the mined indicators to *guide* further investigation. To distinguish trafficked escorts from non-trafficked escorts, investigators seek signals that suggest (whether at present or in the future), for example, that the subject exhibits *movement* between cities or provides *risky* services that were highly correlated with trafficking activity in the past. Another example of a tag studied in this paper, *multi-girl*, flags webpage content that is suggestive of simultaneous advertising of multiple women. Content tagged in this way becomes a fruitful target, both for investigators and non-governmental organizations, for implementing preventive and prosecution measures.

Computationally, this task is challenging for various reasons, the most prominent being the unusual nature of the domain, the lack of training data and multiple forms of bias that emerge in the corpus due to the nature of crawling systems tuned for portals with significant presence of sex ads and reviews. Some of these problems were studied independently in both [4] and [7].

In online sex advertisements, the text content being tagged is typically extracted from the main body of an advertisement scraped from a Web domain that is known (by domain experts building the relevance model for the crawling) to have high prevalence of online sex trafficking activity. An example of such a Web domain is the adult section of `backpage.com`, which contains many ads, both for escorts as well as brick-and-mortar operations like massage parlors that are fronting illicit sex trafficking.

In this paper, we present the architecture (Figure 1) of an end-to-end indicator mining approach called FlagIt (**Fl**exible and **a**daptive **g**eneration of **I**ndicators from **t**ext). To reduce the burdens of manual supervision and potential user bias, FlagIt combines a lightweight expert system with applied research (both recent and classic) in minimally supervised machine learning, including unsupervised text embeddings and semi-supervised heuristic re-labeling [16], to achieve average F-Measure scores slightly below 80%. Also, compared to four baselines, both adaptive and non-adaptive, FlagIt outperforms the best baseline by 2-13% F-Measure on four of the five indicators and is always the best of all adaptive baselines. FlagIt is simple and extensible, and scales to millions of sentences.

Indicator tags discovered by FlagIt are designed to be used both for lead generation and lead investigation. Lead generation arises when investigators search from scratch e.g., an investigator may be interested in investigating the illegal operations of brothels that are fronting as legitimate massage parlors on paper (or by day). Lead investigation is a more focused search, often involving the gathering of evidence, where some information e.g., a phone number, is already available and suspected of involving trafficking activity. Using indicators produced by FlagIt, and the exploratory search capabilities produced as a result of other research in MEMEX, investigators can hone in on crucial evidence without conducting expensive fieldwork or issuing time-consuming subpoenas.

2 Related Work

FlagIt draws on a combination of established techniques in limited-label machine learning, knowledge base construction [10], text embeddings [5], [3] and expert systems [8], as well as recent conceptual advances that have pushed the state-of-the-art [12]. We provide details on relevant sub-topics below. An important contribution of FlagIt is to test the practical *application* and *combination* of established techniques in an unusual domain like human trafficking. To the best of our knowledge, successful data mining results in the HT domain have been very limited, mostly restricted to extensively studied areas such as information extraction [6] and generic document clustering [4]. However, interest in the area has been increasing, a recently published example being [2].

3 Architecture

FlagIt assumes a fixed corpus of webpages. We construct the sentence corpus from Web resources that are known to contain high levels of human trafficking (HT) activity. The sentences are obtained by executing a webpage pre-preprocessing pipeline that was tuned to scale and generalize to a 54 GB HT corpus containing webpages from numerous *Web domains*. First, we use the Readability Text

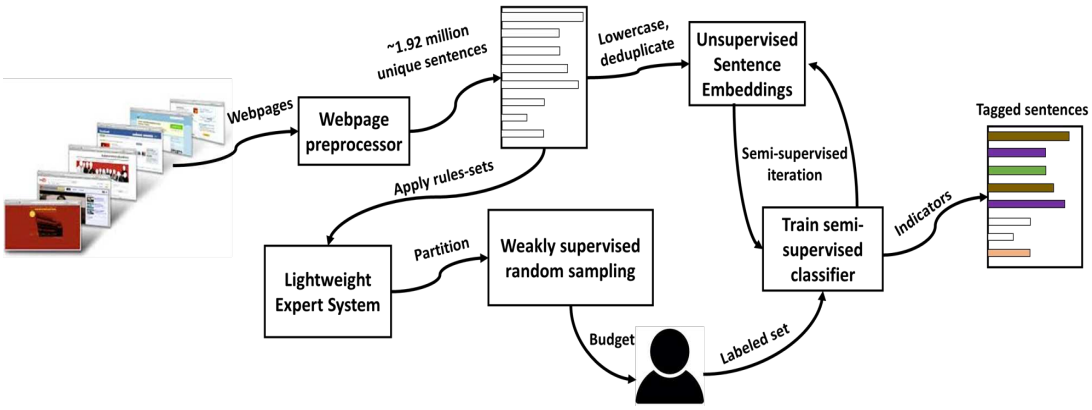


Figure 1: A high-level overview of FlagIt, which accepts a (possibly streaming) domain-specific corpus of crawled webpages as input, extracts sentences from those pages and tags them with a set of indicators. The output of FlagIt is deployed in a GUI (not shown) and can be used to quickly zoom in on suspicious pages and text within the corpus.

Extractor[3] (RTE) to scrape descriptive content from each webpage in the corpus. We tuned RTE for high recall i.e. with high probability, all relevant sentences constituting the main content of the webpage are extracted, but some irrelevant sentences (e.g., 'taylor profile was posted 14 weeks 4 days ago') may also be extracted. Next, the output of RTE is segmented into a list of sentences by using newlines and consecutive occurrences of three whitespace characters as delimiters. All sentences were converted to lowercase and deduplicated before executing the lightweight expert system (LES), which relies on capitalizations and other text features to perform well. Following LES execution, the minimally supervised machine learning modules (e.g., the text embeddings) are executed.

We briefly describe some of the critical aspects of the LES, as it is an important component in FlagIt. At its core, the LES relies on shallow *pattern matching* rules, with each rule exclusively falling into one of four *rule categories* depending on its predictive relationship to the indicator: positive (P), strong positive (Sp), negative (N) and strong negative (Sn). The 'strong' qualifier is meant to express user confidence, providing users a simple but effective way of expressing when they are very sure about a rule, and when they are not. The categories are motivated by extensive research in cognitive science and crowdsourcing showing that asking users for fine-grained confidence outputs (including probabilities) is rife with issues of consistency and labeling context (a notorious example being the order in which outputs are elicited).

We define a pattern matching rule as a finite sequence of *pattern elements*, where a pattern element can either be a *constant* token (a finite sequence of Unicode characters) or a pattern variable V. In keeping with the lightweight nature of the system, four kinds of pattern variables are considered. We describe these variables below, followed by some representative examples.

Glossary Variable. Given a glossary G (a finite set of tokens), a glossary variable V_G can take exactly one token from G as its value. In some cases, glossaries are easily available from the Web (e.g., a list of common English names) but in other cases, have to be specified either by the expert or through entity set expansion and glossary mining [11]. For the indicators considered in this paper, five glossaries are used for both *incall* and *outcall*, two glossaries each for *movement* and *risky*, and four glossaries for *multi-girl*.

Regular Expression (RegEx). Pattern variables can also be encoded as regular expressions. For efficiency (and also noting that domain experts are non-technical), we limit expressivity of regular expressions in FlagIt. The most commonly used RegEx was to check for tokens possibly symbolizing names and geolocations by specifying capitalization and alphabetic constraints on the tokens.

[3]RTE was available and open-source at the time of preprocessing. It has since been taken offline and superseded by a new web scraping API called Mercury: `https://www.readability.com/`

NLP Tags. Despite being lightweight, the expert system can make use of tags, such as POS tags and English dependency labels[4], output by NLP packages like spaCy [1]. As a subsequent example illustrates for *incall*, dependency labels, such as an *adjectival modifier* tag, can be used to great effect without complex overhead.

Named Entities. The LES can also use an expected named entity *type* as a pattern variable specification. The variable is assigned the word, only if it is extracted as a named entity of that type [9].

Example. Given a glossary *place*, containing residence terms and synonyms such as *apartment* and *studio*, and another glossary *priv* containing words such as *private* and *discreet*, a simple but effective Sp rule for *incall* is the sequence $(G_{priv} < -amod, G_{place})$, which would match a pattern such as *private apartment* in the text. The $< -amod$ specifies that the word must have an adjectival modifier dependency tag. This example shows that finite combinations of pattern variables are also possible. In a slight abuse of notation, we use the symbols P, Sp, N and Sn to respectively refer to the collections of pattern matching rules defined by domain experts. A rule R, when applied to a sentence, yields *True* if at least one pattern is successfully matched in the sentence (said to be *covered* by the rule).

Because rules from multiple rule categories can cover a sentence, the notion of a *rules-set* becomes necessary to enforce a *partition* of the corpus. The FlagIt LES supports seven rules-sets, resulting in a 7-split partition: P OR Sp, N OR Sn, Sp AND N, Sn AND P, Sp AND Sn, P AND N, and Null. The last is when no rule (from any of the four categories) yields *True*. OR in a rule such as P OR Sp indicates that only a rule from P or Sp (or both) covers a sentence. AND in a rule such as Sp AND N indicates that the sentence must be covered by a rule from Sp, as well as from N. Where applicable, Sn takes *precedence* over N and Sp over P. For example, if a rules-set such as (P AND Sp) AND Sn fires, it is considered equivalent to the rules-set Sp AND SN firing.

Because of the specified precedence, the rules-sets described above are such that a sentence will be covered by exactly one of them, yielding a partition over a sentence corpus. This partition is used in a weakly supervised setting to obtain labeled data that can then be used in a minimally supervised machine learning setting.

3.1 Minimally Supervised Components

Given a sentence from the corpus of segmented sentences, the indicator mining problem can be modeled as *multi-label*[5] text classification, assuming that representative training data and feature functions are available. Unfortunately, we have no training data available for this task, and enormous *negative* class skew (higher than 90% in many cases) in the task domain preempts random sampling and labeling. Because indicator mining is an emerging problem in data mining, it is also not clear what indicator-specific features will lead to good performance for a given indicator. FlagIt attempts to address these challenges in a simple and lightweight manner.

The version of FlagIt described in this paper implements five indicator classifiers, namely *incall*, *outcall*, *movement*, *risky* and *multi-girl*. *Incall* explicitly indicates that a client must visit an escort (conversely for *outcall*) at a specified location. *Movement* indicates that the escort is visiting the locale (usually for short periods) from a different previous locale, while *risky* indicates that the escort is willing to engage in risky sexual activities (as determined by domain experts). *Multi-girl* (along with *movement*) is highly correlated with organized sex trafficking activity[6], and indicates that multiple girls are being simultaneously advertised.

FlagIt follows a *weakly supervised random sampling* scheme for training set construction. First, we construct a small but diverse training set per indicator by randomly sampling a small number of sentences from each of the seven rules-sets in the partition. In keeping with the minimal supervision goals of this paper, *labeling budget* was restricted to a small number (140), meaning that, in the typical case, 20 sentences (for each rules-set category) were labeled by a user. The only exception is when a rules-set did not cover 20 sentences, in which case we used a form of iterative, re-distributive sampling whereby each sentence covered by that rules-set was labeled, and the rest of the budget

[4] See `https://spacy.io/docs/api/annotation` for a description.

[5] Distinguished herein from *multi-class*, since a sentence could be labeled with multiple indicators.

[6] In an extended version of FlagIt, an explicit indicator for *massage parlor* activity is also considered. This indicator can overlap with, but is technically distinct from, *multi-girl*.

was divided between the remaining rules-sets that had more than 20 sentences[7]. In the worst case, the scheme would be reduced to sampling all 140 sentences from the Null rules-set but this never happened in practice for any of the five indicators in FlagIt.

Two options to considerably mitigate or even eliminate the twin labors of labeling and feature crafting effort are semi-supervised learning and unsupervised text embeddings respectively. While the former has been long studied, the latter is an intense topic of recent research. FlagIt leverages both. First, given that the weakly supervised random sampling scheme described earlier for constructing the evaluation set for each indicator is extremely small compared to the whole corpus, we posit that semi-supervised learning can be used to leverage this body of unlabeled text to further improve the results [16]. We integrate a simple, and classic semi-supervised scheme in FlagIt: first, we train an initial (indicator-specific) classifier on the perfectly labeled training set, and use the classifier to generate a positive label *probability* (for that indicator) for each unlabeled sentence in the overall corpus. Next, we pick some percentage of unlabeled sentences from the 'extreme' ends of the probability distribution and heuristically label these sentences. We re-train the classifier by supplementing the perfectly labeled training set with the heuristically labeled sentences. This process can be iterated a certain constant number of steps, or till some convergence criterion is met. The final classifier is tested on a held-out test set. We found in early experimentation that even a relatively simple scheme of heuristically relabeling 45% (from each end of the probability distribution) unlabeled data, and a single iteration, yields significant improvements over not using any form of semi-supervised learning.

4 Brief Summary of Experimental Results

We evaluated[8] FlagIt on five diverse indicators against a range of competitive baselines, with encouraging results. Compared to four baselines, three adaptive[9] and one non-adaptive[10], FlagIt outperforms the best baseline on the F1-Measure metric by 2-13% on four of the five indicators and is always the best of all adaptive baselines. We also found that, while semi-supervised learning always improves performance in FlagIt (by 1-9%), the choice of the unsupervised text embedding algorithm can be crucial. For example, the recently released fasttext package [5], based on a *bag-of-tricks* approach, proved to be a much more suitable fit for an irregular domain like human trafficking than the widely used paragraph2vec algorithm [3], which performed badly despite extensive parameter tuning.

5 Impact, Deployment and Future Work

FlagIt is already being integrated into one of two end-to-end domain-specific search engines (denoted herein as DS1 and DS2 to maintain anonymity) being developed under the DARPA MEMEX program, with possible integration into the other system (DS2) in the medium term. The key idea is to show webpages highly ranked with respect to a given indicator, either because of highly indicative sentences, or the number of sentences with non-trivial indicator probability. Both DS1 and DS2 are currently being used by over 200 law enforcement agencies in the US to combat human trafficking. The tools are complementary to each other. Indicator mining is an important application of both search and lead generation, especially considering the limited resources of law enforcement. Analysis is showing that indicator co-occurrences show unusual correlations with each other, which raises interesting avenues both for quantitative social science research, and for complex multi-label classification models. Data collection is currently ongoing to quantify such correlations. Also, although only five indicators have been formally evaluated thus far, FlagIt is designed to support between 11-13 (and potentially many more) indicators specified by domain experts at the time of writing. Because of its lightweight, minimally supervised architecture, this extension is possible with only a few hours of effort, divided between specifying rules in spaCy, sampling sentences, and providing manual indicator labels. Another important line is research is supporting FlagIt for explainable machine learning, for which we are investigating two options: highlighting highly indicative sentences flagged by FlagIt in the GUI, or using the approach espoused by [13] with the pattern matching rules in the lightweight expert system serving the role of locally explainable classifiers.

[7] The process is iterative because a rules-set may not now cover the *new* local labeling budget (by definition, greater than 20).

[8] Because of the sampling scheme, classes were relatively balanced during both training and testing.

[9] Resp. based on bag-of-words, limited label supervised fasttext and paragraph2vec [3].

[10] A highly tuned version of the LES component.

References

[1] spacy nlp toolkit. https://spacy.io/docs/usage/resources, 2017. Accessed: 2017-08-12.

[2] Hamidreza Alvari, Paulo Shakarian, and JE Kelly Snyder. Semi-supervised learning for detecting human trafficking. *Security Informatics*, 6(1):1, 2017.

[3] Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015.

[4] Artur Dubrawski, Kyle Miller, Matthew Barnes, Benedikt Boecking, and Emily Kennedy. Leveraging publicly available data to discern patterns of human-trafficking activity. *Journal of Human Trafficking*, 1(1):65–85, 2015.

[5] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[6] Rahul Kapoor, Mayank Kejriwal, and Pedro Szekely. Using contexts and constraints for improved geotagging of human trafficking webpages. In *Proceedings of the Fourth International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*, page 3. ACM, 2017.

[7] Mark Latonero. Human trafficking online: The role of social networking sites and online classifieds. 2011.

[8] Shu-Hsien Liao. Expert system methodologies and applications - a decade review from 1995 to 2004. *Expert systems with applications*, 28(1):93–103, 2005.

[9] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[10] Feng Niu, Ce Zhang, Christopher Ré, and Jude W Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. *VLDS*, 12:25–28, 2012.

[11] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 938–947. Association for Computational Linguistics, 2009.

[12] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, pages 3567–3575, 2016.

[13] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[14] Elizabeth M Wheaton, Edward J Schauer, and Thomas V Galli. Economics of human trafficking. *International Migration*, 48(4):114–141, 2010.

[15] Jennifer Xu and Hsinchun Chen. The topology of dark networks. *Communications of the ACM*, 51(10):58–65, 2008.

[16] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

[17] Ahmed T Zulkarnine, Richard Frank, Bryan Monk, Julianna Mitchell, and Garth Davies. Surfacing collaborated networks in dark web to find illicit and criminal content. In *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on*, pages 109–114. IEEE, 2016.

A DIRT-T Approach to Unsupervised Domain Adaptation

Rui Shu
Stanford University

Hung Bui
Adobe Research

Stefano Ermon
Stanford University

Abstract

Domain adaptation refers to the problem of how to leverage labels in one source domain to boost up learning performance in a new target domain where labels are scarcely available or completely unavailable. In this paper, we address these issues through the lens of the cluster assumption, i.e., decision boundaries should not cross high-density data regions. We propose two novel and related models: (1) the Virtual Adversarial Domain Adaptation (VADA) model, which combines domain adversarial training with a penalty term that punishes the violation of the cluster assumption; (2) the Decision-boundary Iterative Refinement Training with a Teacher (DIRT-T) model, which takes the VADA model as initialization and employs natural gradient steps to further minimize the cluster assumption violation. Extensive empirical results demonstrate that the combination of these two models significantly improve the state-of-the-art performance on several visual domain adaptation benchmarks.

1 Introduction

In many tasks, direct access to vast quantities of labeled data to the task of interest (the target domain) is either costly or otherwise absent, but labels are readily available for related training sets (the source domain). However, the source data distribution is often dissimilar to the target data distribution, and the resulting significant covariate shift is often detrimental to the performance of the source-trained model when applied to the target domain [1]. Solving the covariate shift problem of this nature is commonly referred to as domain adaptation. In this paper, we consider a challenging setting of domain adaptation where 1) we are provided with fully-labeled source samples and completely-unlabeled target samples, and 2) the existence of a classifier in the hypothesis class with low error on both source and target distributions is not guaranteed. Borrowing approximately the terminology from [2], we refer to this setting as unsupervised, *non-conservative* domain adaptation.

To tackle unsupervised domain adaptation, [3] proposed to constrain the classifier to only rely on domain-invariant features. This is achieved by training the classifier to perform well on the source domain while minimizing the divergence between features extracted from the source versus target domains. To achieve divergence minimization, [3] employ domain adversarial training. However, a fundamental weakness of domain adversarial training is that it does not account for the case where good generalization on the source domain hurts target performance in the non-conservative setting.

[4] addressed these issues by replacing domain adversarial training with asymmetric tri-training (ATT), which relies on the assumption that target samples that are labeled by a source-trained classifier with high confidence *are* correctly labeled by the source classifier. In this paper, we consider an orthogonal assumption: the cluster assumption [5], that covariate distribution contains separated data clusters and that data samples in the same cluster share the same class label. This assumption introduces an additional bias where we seek decision boundaries that do not go through high-density regions. Based on this intuition, we propose two novel models: (1) the Virtual Adversarial Domain Adaptation (VADA) model which incorporates an additional virtual adversarial training [6] and

Workshop on Learning with Limited Labeled Data (NIPS 2017), Long Beach, CA, USA.

conditional entropy loss to push the decision boundaries away from the empirical data, and (2) the Decision-boundary Iterative Refinement Training with a Teacher (DIRT-T) model which uses natural gradient to further refine the output of the VADA model while focusing purely on the target domain. We demonstrate that

1. In conservative domain adaptation, where the classifier is trained to perform well on the source domain, VADA can be used to further constrain the hypothesis space by penalizing violations of the clustering assumption, thereby improving domain adversarial training.
2. In non-conservative domain adaptation, where we account for the mismatch between the source and target optimal classifiers, DIRT-T allows us to transition from a good joint (source and target) classifier (VADA) to a better target domain classifier. Interestingly, we demonstrate the advantage of natural gradients in DIRT-T refinement steps.

We report results for domain adaptation in digits classification (MNIST-M, MNIST, SYN DIGITS, SVHN), traffic sign classification (SYN SIGNS, GTSRB), and general object classification (STL-10, CIFAR-10). We show that, in nearly all experiments, VADA improves upon previous methods and that DIRT-T improves upon VADA, setting new state-of-the-art performance across a wide range of visual domain adaptation benchmarks. In adapting MNIST $\rightarrow$ SVHN, a very challenging task, we out-perform ATT by over 20%.

2 Related Work

Given the extensive literature on domain adaptation, we highlight the works most relevant to our paper. [3] proposed to project both source and target distributions into some feature space and encourage distribution matching in the feature space. To better perform non-conservative domain adaptation, [4] proposed to modify tri-training [7] for domain adaptation, leveraging the assumption that highly-confident predictions are correct predictions [8]. Both methods are based on [9]'s theoretical analysis of domain adaptation, which states the following,

Theorem 1 *[9] Let $\mathcal{H}$ be the hypothesis space and let (X_s, ϵ_s) and (X_t, ϵ_t) be the two domains and their corresponding generalization error functions. Then for any $h \in \mathcal{H}$,*

$$\epsilon_t(h) \leq \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) + \epsilon_s(h) + \min_{h' \in \mathcal{H}} \epsilon_t(h') + \epsilon_s(h'), \tag{1}$$

where $d_{\mathcal{H}\Delta\mathcal{H}}$ denotes the $\mathcal{H}\Delta\mathcal{H}$-distance between the domains X_s and X_t,

$$d_{\mathcal{H}\Delta\mathcal{H}} = 2 \sup_{h, h' \in \mathcal{H}} \left| \mathbb{E}_{x \sim \mathcal{D}_s} \left[h(x) \neq h'(x) \right] - \mathbb{E}_{x \sim \mathcal{D}_t} \left[h(x) \neq h'(x) \right] \right|. \tag{2}$$

Intuitively, $d_{\mathcal{H}\Delta\mathcal{H}}$ measures the extent to which small changes to the hypothesis in the source domain can lead to large changes in the target domain. It is evident that $d_{\mathcal{H}\Delta\mathcal{H}}$ relates intimately to the complexity of the hypothesis space and the divergence between the source and target domains. For disjoint domains and infinite-capacity models, $d_{\mathcal{H}\Delta\mathcal{H}}$ is maximal.

3 Constraining via Conditional Entropy Minimization

In this paper, we apply the cluster assumption to domain adaptation. The cluster assumption assumes that the input distribution X contains density clusters and that points in the same cluster come from the same class. This assumption has been extensively studied and applied successfully to a wide range of classification tasks [5, 6, 10, 11, 12, 13, 14]. If the cluster assumption holds, the optimal decision boundaries should occur far away from data-dense regions in the space of $\mathcal{X}$ [5]. Following [10], we achieve this behavior via minimization of the conditional entropy with respect to the target distribution,

$$\mathcal{L}_c(\theta; \mathcal{D}_t) = -\mathbb{E}_{x \sim \mathcal{D}_t} \left[h_\theta(x)^\top \ln h_\theta(x) \right], \tag{3}$$

where $\mathcal{D}_t$ is the target domain data and h maps to the K-simplex. Intuitively, minimizing the conditional entropy forces the classifier to be confident on the unlabeled target data, which occurs if the classifier places its decision boundaries far from data-dense regions. The conditional entropy

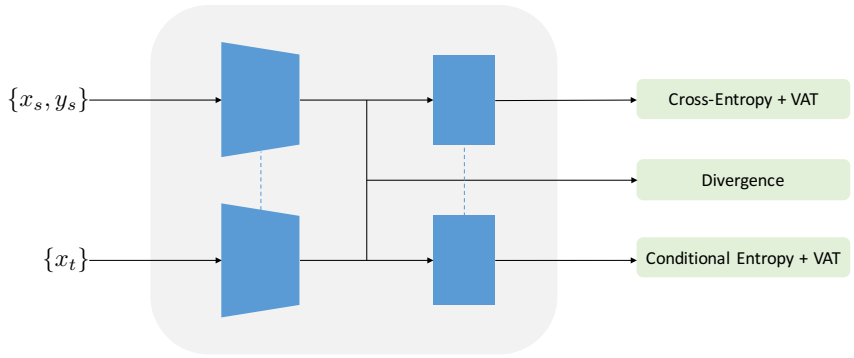


Figure 1: VADA improves upon domain adversarial training by additionally penalizing violations of the cluster assumption.

must be empirically estimated using the available data. However, [10] notes that this approximation breaks down if the classifier h is not locally-Lipschitz. To prevent this, we propose to incorporate the locally-Lipschitz constraint via virtual adversarial training [6] and add to the objective function the additional term

$$\mathcal{L}_v(\theta; \mathcal{D}) = \mathbb{E}_{x \sim \mathcal{D}} \left[\max_{\|r\| \leq \epsilon} \mathrm{D}_{\mathrm{KL}}(h_{\hat{\theta}}(x) \| h_\theta(x + r)) \right], \tag{4}$$

which enforces classifier consistency around the norm-ball neighborhood of each sample x, where $\hat{\theta}$ is a copy of θ. Note that virtual adversarial training can be applied with respect to either the target or source distributions. We can combine the conditional entropy minimization objective and domain adversarial training to yield

$$\min_\theta . \mathcal{L}_y(\theta; \mathcal{D}_s) + \lambda_d \mathcal{L}_d(\theta; \mathcal{D}_s, \mathcal{D}_t) + \lambda_s \mathcal{L}_v(\theta; \mathcal{D}_s) + \lambda_t \left[\mathcal{L}_v(\theta; \mathcal{D}_t) + \mathcal{L}_c(\theta; D_t) \right], \tag{5}$$

a basic combination of cross-entropy $\mathcal{L}_y$, domain adversarial $\mathcal{L}_d$ [3], and semi-supervised $(\mathcal{L}_v, \mathcal{L}_c)$ objectives. We refer to this as the Virtual Adversarial Domain Adaptation (VADA) model.

$\mathcal{H}\Delta\mathcal{H}$-Distance Minimization. VADA aligns well with the theory of domain adaptation provided in Theorem 1. Let the loss,

$$\mathcal{L}_t(\theta) = \mathcal{L}_v(\theta; \mathcal{D}_t) + \mathcal{L}_c(\theta; D_t), \tag{6}$$

be a proxy measure for the degree of violation of the target-side cluster assumption. For a reasonably small choice of λ_t, VADA can penalize models with high $\mathcal{L}_t$ while still enabling decently small source generalization error. This penalization effectively rejects hypotheses which egregiously violate the target-side cluster assumption. By rejecting such hypotheses from the hypothesis space $\mathcal{H}$, VADA reduces $d_{\mathcal{H}\Delta\mathcal{H}}$ and yields a tighter bound on the target generalization error. We verify empirically that VADA achieves significant improvements over existing models on multiple domain adaptation benchmarks (Table 1).

4 Decision-boundary Iterative Refinement Training

In non-conservative domain adaptation, we account for the following inequality,

$$\min_{h \in \mathcal{H}} \epsilon_t(h) < \epsilon_t(h^a) \text{ where } h^a = \arg\min_{h \in \mathcal{H}} \epsilon_s(h) + \epsilon_t(h), \tag{7}$$

where (ϵ_s, ϵ_t) are generalization error functions for the source and target domains. This means that, for a given hypothesis class $\mathcal{H}$, the optimal classifier in the source domain does not coincide with the optimal classifier in the target domain.

We assume that the optimality gap in Eq. (7) results from violation of the cluster assumption. In other words, we suppose that any source-optimal classifier drawn from our hypothesis space *necessarily* violates the cluster assumption in the target domain. Since VADA is still constrained to do well on the source domain (ensured by choosing a small λ_t), it will still violate the target-side cluster assumption to some extent.

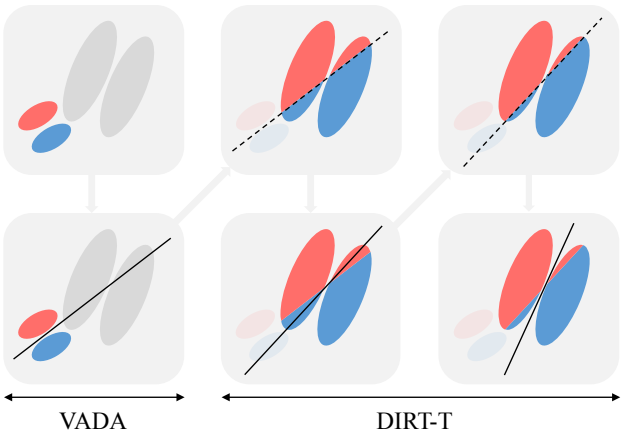


Figure 2: DIRT-T uses VADA as initialization. After removing the source training signal, DIRT-T minimizes cluster assumption violation in the target domain through a series of natural gradient steps.

Under this assumption, the natural solution is to initialize with the VADA model and then further minimize the cluster assumption violation in the target domain. In particular, we first use VADA to learn an initial classifier h_{θ_0}. Next, we incrementally push the classifier's decision boundary away from data-dense regions by minimizing the proxy target-side cluster assumption violation loss $\mathcal{L}_t$ in Eq. (6). We denote this procedure Decision-boundary Iterative Refinement Training (DIRT).

4.1 Decision-boundary Iterative Refinement Training with a Teacher

Stochastic gradient descent minimizes the proxy loss $\mathcal{L}_t$ by selecting gradient steps $\Delta\theta$ according to the following objective,

$$\min_{\Delta\theta} \mathcal{L}_t(\theta + \Delta\theta) \tag{8}$$

$$\text{s.t. } \|\Delta\theta\| \leq \epsilon, \tag{9}$$

which defines the neighborhood in the parameter space. This notion of neighborhood is sensitive to the parameterization of the model; depending on the parameterization, a seemingly small step $\Delta\theta$ may result in a vastly different classifier. This contradicts our intention of incrementally and locally pushing the decision boundary to a local conditional entropy minimum, which requires that the decision boundary of $h_{\theta+\Delta\theta}$ stay close to that of h_θ. It is therefore important to define a neighborhood that is parameterization-invariant. Following [15], we instead select $\Delta\theta$ using the following objective,

$$\min_{\Delta\theta} \mathcal{L}_t(\theta + \Delta\theta)$$
$$\text{s.t. } \mathbb{E}_{x\sim D_t}\left[\mathrm{D}_{\mathrm{KL}}(h_\theta(x)\|h_{\theta+\Delta\theta}(x))\right] \leq \epsilon. \tag{10}$$

Each optimization step now solves for a gradient step $\Delta\theta$ that minimizes the conditional entropy, subject to the constraint that the Kullback-Leibler divergence between $h_\theta(x)$ and $h_{\theta+\Delta\theta}(x)$ is small for $x \sim \mathcal{X}_t$. The corresponding Lagrangian suggests that one can instead minimize a sequence of optimization problems

$$\min_{\theta_n} \lambda_t \mathcal{L}_t(\theta_n) + \beta_t \mathbb{E}\left[\mathrm{D}_{\mathrm{KL}}(h_{\theta_{n-1}}(x)\|h_{\theta_n}(x))\right], \tag{11}$$

that approximates the application of a series of natural gradient steps.

In practice, each of optimization problems in Eq. (11) can be solved approximately via a finite number of stochastic gradient descent steps. We denote the number of steps taken to be the refinement interval B. Similar to [14], we use the Adam Optimizer with Polyak averaging [16]. We interpret $h_{\theta_{n-1}}$ as a (sub-optimal) teacher for the student model h_{θ_n}, which is trained to stay close to the teacher model while seeking to reduce the cluster assumption violation. As a result, we denote this model as Decision-boundary Iterative Refinement Training with a Teacher (DIRT-T).

Weakly-Supervised Learning. This sequence of optimization problems has a natural interpretation that exposes a connection to weakly-supervised learning. In each optimization problem, the teacher

model $h_{\theta_{n-1}}$ pseudo-labels the target samples with noisy labels. Rather than naively training the student model h_{θ_n} on the noisy labels, the additional training signal $\mathcal{L}_t$ allows the student model to place its decision boundaries further from the data. If the clustering assumption holds and the initial noisy labels are sufficiently similar to the true labels, conditional entropy minimization can improve the placement of the decision boundaries [17].

Domain Adaptation. An alternative interpretation is that DIRT-T is the *recursive* extension of VADA, where the act of pseudo-labeling of the target distribution constructs a new "source" domain (i.e. target distribution X_t with pseudo-labels). The sequence of optimization problems can then be seen as a sequence of non-conservative domain adaptation problems in which $X_s = X_t$ but $p_s(y \mid x) \neq p_t(y \mid x)$, where $p_s(y \mid x) = h_{\theta_{n-1}}(x)$ and $p_t(y \mid x)$ is the true conditional label distribution in the target domain. Since $d_{\mathcal{H}\Delta\mathcal{H}}$ is strictly zero in this sequence of optimization problems, domain adversarial training is no longer necessary. Furthermore, if $\mathcal{L}_t$ minimization does improve the student classifier, then the gap in Eq. (7) should get smaller each time the source domain is updated.

4.2 Model Evaluation and Conclusion

Source Target	MNIST MNIST-M	SVHN MNIST	MNIST SVHN	DIGITS SVHN	SIGNS GTSRB	CIFAR STL	STL CIFAR
MMD [18]	76.9	71.1	-	88.0	91.1	-	-
DANN [3]	81.5	71.1	35.7	90.3	88.7	-	-
DRCN [19]	-	82.0	40.1	-	-	66.4	58.7
DSN [20]	83.2	82.7	-	91.2	93.1	-	-
kNN-Ad [21]	86.7	78.8	40.3	-	-	-	-
ATT [4]	94.2	86.2	52.8	92.9	96.2	-	-
Π-model (aug) [22]	-	92.0	71.4	94.2	98.4	76.3	64.2
Without Instance-Normalized Input:							
Source-Only	58.5	77.0	27.9	86.9	79.6	76.3	63.6
VADA	97.7	97.9	47.5	94.8	98.8	**80.0**	73.5
DIRT-T	**98.9**	**99.4**	**54.5**	**96.1**	**99.5**	-	**75.3**
With Instance-Normalized Input:							
Source-Only	59.9	82.4	40.9	88.6	86.2	77.0	62.6
VADA	95.7	94.5	73.3	94.9	99.2	**78.3**	71.4
DIRT-T	**98.7**	**99.4**	**76.5**	**96.2**	**99.6**	-	**73.3**

Table 1: Accuracy. In all settings, both VADA and DIRT-T achieve state-of-the-art performance in all settings. DIRT-T omitted for CIFAR → STL since STL only has 5000 images in the training set.

Source Target	MNIST MNIST-M	SVHN MNIST	MNIST SVHN	DIGITS SVHN	SIGNS GTSRB	CIFAR STL	STL CIFAR
ATT	37.1	16.1	17.9	9.0	**20.5**	-	-
Π-model (aug)	-	3.7	18.1	**10.6**	1.0	**4.5**	7.4
DIRT-T	**40.4**	**22.4**	26.6	9.2	19.9	-	**11.7**
DIRT-T (W.I.N.I.)	38.8	17.0	**35.6**	7.6	13.4	-	10.7

Table 2: Additional comparison of the margin of improvement computed by taking the reported performance of each model and subtracting the reported source-only performance in the respective papers. W.I.N.I. indicates "with instance-normalized input."

We achieve state-of-the-art results across all tasks. Our experiments demonstrate that VADA achieves strong performance across several visual domain adaptation benchmarks, and DIRT-T further improves VADA performance. In four of the tasks (MNIST → MNIST-M, SVHN → MNIST, MNIST → SVHN, STL → CIFAR), we achieve substantial margin of improvement compared to previous models. In the remaining three tasks, our improvement margin over the source-only model is competitive against previous models. Our closest competitor is the Π-model. However, unlike the Π-model, we do not perform data augmentation. Our proposed models open up several possibilities for future work. One possibility is to apply DIRT-T to weakly supervised learning; another is to improve the natural gradient approximation via K-FAC [23] and PPO [24]. Given the strong performance of our models, we also recommend them for other downstream domain adaptation applications.

References

[1] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

[2] Shai Ben-David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136, 2010.

[3] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015.

[4] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. *arXiv preprint arXiv:1702.08400*, 2017.

[5] Olivier Chapelle and Alexander Zien. Semi-supervised classification by low density separation. In *AISTATS*, pages 57–64, 2005.

[6] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[7] Zhi-Hua Zhou and Ming Li. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[8] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

[9] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[10] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2005.

[11] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on Challenges in Representation Learning, ICML*, volume 3, page 2, 2013.

[12] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad gan. *arXiv preprint arXiv:1705.09783*, 2017.

[13] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

[14] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. 2017.

[15] Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

[16] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[17] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[18] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105, 2015.

[19] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. In *European Conference on Computer Vision*, pages 597–613. Springer, 2016.

[20] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.

[21] Ozan Sener, Hyun Oh Song, Ashutosh Saxena, and Silvio Savarese. Learning transferrable representations for unsupervised domain adaptation. In *Advances in Neural Information Processing Systems*, pages 2110–2118, 2016.

[22] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for domain adaptation. *arXiv preprint arXiv:1706.05208*, 2017.

[23] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417, 2015.

[24] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Smooth Neighbors on Teacher Graphs for Semi-supervised Learning

Yucen Luo[†], **Jun Zhu**[†], **Mengxi Li**[‡], **Yong Ren**[†], **Bo Zhang**[†]
[†] Department of Computer Science & Technology, Tsinghua University, Beijing, China
‡ Department of Electronic Engineering, Tsinghua University, Beijing, China
{luoyc15@mails, dcszj@mail, limq14@mails,renyong15,dcszb@mail}.tsinghua.edu.cn

Abstract

The paper proposes an inductive deep semi-supervised learning method, called *Smooth Neighbors on Teacher Graphs (SNTG)*. At each iteration during training, a graph is dynamically constructed based on predictions of the teacher model, i.e., the implicit self-ensemble of models. Then the graph serves as a similarity measure with respect to which the representations of "similar" neighboring points are learned to be smooth on the low dimensional manifold. We achieve state-of-the-art results on semi-supervised learning benchmarks. The error rates are 9.89%, 3.99% for CIFAR-10 with 4000 labels, SVHN with 500 labels, respectively. In particular, the improvements are significant when the labels are scarce. For non-augmented MNIST with only 20 labels, the error rate is reduced from previous 4.81% to 1.36%. Our method also shows robustness to incorrect labels.

1 Introduction

Recently due to the great advances of deep learning, remarkable results have been achieved on semi-supervised learning (SSL) [8, 13, 9]. Among these works, perturbation-based methods [12, 9, 15, 11] have demonstrated great promise, which enforce smooth predictions of unlabeled data under different perturbations. Consider a training set $\mathcal{D}$ consists of N examples, out of which L have labels. Let $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{L}$ be the labeled set and $\mathcal{U} = \{x_i\}_{i=L+1}^{N}$ be the unlabeled set where the input $x_i \in \mathcal{X}$ and the label $y_i \in \{1, 2, ..., K\}$. We aim to learn $f : \mathcal{X} \to [0, 1]^K$ parameterized by $\theta \in \Theta$ by solving:

$$\min_{\theta} \sum_{i=1}^{L} \ell(f(x_i; \theta), y_i) + \lambda R(\theta, \mathcal{L}, \mathcal{U}), \tag{1}$$

where ℓ is the supervised loss (e.g., cross-entropy loss) and $f(x; \theta)$ denotes the predictive distribution $p(y|x; \theta)$. The regularization term R is used to leverage unlabeled data and $\lambda \geq 0$ is the coefficient. All perturbation-based methods fit in Eq. (1) by defining R as a consistency loss:

$$R_C(\theta, \mathcal{L}, \mathcal{U}) = \sum_{i=1}^{N} \mathbb{E}_{\xi', \xi}\, d\big(\tilde{f}(x_i; \theta', \xi'), f(x_i; \theta, \xi)\big), \tag{2}$$

where $\tilde{f}$ is a "teacher" with parameters θ' and random perturbations ξ', similarly, f is a "student" with θ and ξ, and $d(\cdot, \cdot)$ denotes the divergence between two distributions (e.g., l_2 distance or KL divergence) [15]. The perturbations include the input noise and the network dropout. $\tilde{f}$ is defined as an implicit ensemble of previous student models and is expected to give better predictions than the student. $\tilde{f}(x)$ can be seen as the training targets for the unlabeled inputs and the student model is supposed to predict consistently with $\tilde{f}(x)$. Below are several ways to define the teacher model $\tilde{f}$:

Π **model [9].** $\tilde{f}$ shares the same parameters with f, i.e., $\theta' = \theta$ in Eq. (2). It evaluates the network twice under different realizations of i.i.d. perturbations ξ' and ξ and minimizes their l_2 distance.

Temporal ensembling (TempEns) [9]. To reduce the variance of the targets, TempEns maintains an exponentially moving average (EMA) of previous predictions over epochs as $\tilde{f}$. The ensemble

Table 1: Error rates (%) on benchmark datasets without augmentation, averaged over 10 runs.

Models	MNIST (L=100)	SVHN (L=1000)	CIFAR10 (L=4000)
Ladder network [12]	0.89±0.50	–	20.40±0.47
Improved GAN [13]	0.93±0.065	8.11±1.3	18.63±2.32
ALI [5]	–	7.42±0.65	17.99±1.62
Triple GAN [10]	0.91±0.58	5.77±0.17	16.99±0.36
GoodSemiBadGAN [4]	0.795±0.098	4.25±0.03	14.41±0.03
Π model [9]	0.89±0.15*	5.43±0.25	16.55±0.29
Π+SNTG (**ours**)	**0.66±0.07**	4.22±0.16	13.62±0.17
VAT [11]	1.36	5.77	14.82
VAT+Ent	–	4.28	13.15
VAT+Ent+SNTG (**ours**)	–	**4.02±0.20**	**12.49±0.36**

output is defined as $\tilde{F}^{(t)}(x_i) = \alpha\tilde{F}^{(t-1)}(x_i) + (1-\alpha)f^{(t)}(x_i;\theta,\xi)$, where $f^{(t)} : \mathcal{X} \to [0,1]^K$ is the student model at training epoch t and α is the pre-defined momentum. The target given by $\tilde{f}$ for x_i at epoch t is the debias correction of $\tilde{F}^{(t)}$, i.e., $\tilde{f}^{(t)}(x_i) = \tilde{F}^{(t)}(x_i)/(1-\alpha^t)$. Since the targets are based on EMA, the network only needs to be evaluated once, leading to a speed-up for Π model.

Mean teacher (MT) [15]. Instead of averaging previous predictions every epoch, MT updates the targets more frequently to form a better teacher, i.e., it averages parameters θ every iteration: $\theta' \leftarrow \alpha\theta' + (1-\alpha)\theta$. It also evaluates the network twice by the teacher and the student respectively.

VAT [11]. Instead of l_2 distance, VAT defines R_C as KL divergence between the model prediction and that of the input under adversarial perturbation ξ'_{adv}: $R_C = \sum_{i=1}^{N} \mathrm{KL}(\tilde{f}(x_i;\theta) \| f(x_i;\theta,\xi'_{adv}))$. It is assumed that a model trained under the worst-case (adversarial) perturbations will generalize well [11]. VAT resembles Π model but distinguishes itself in the distance metric and the type of perturbations. $\tilde{f}$ can be seen as the teacher while f with ξ'_{adv} is treated as the student.

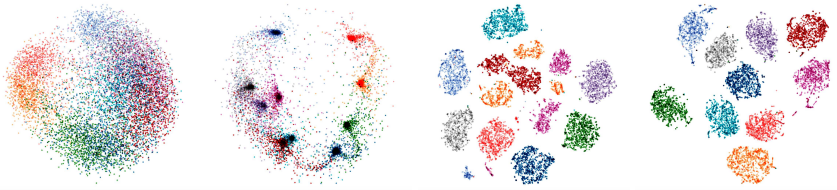
These approaches aim to fuse the inputs into coherent clusters by smoothing the mapping function locally. However, they only regularize the output to be smooth near each single data point, while ignoring the connections between data points, therefore not fully utilizing the information in unlabeled data structure, such as clusters or manifolds. An extreme situation may happen where the function is smooth in the vicinity of each unlabeled point but not smooth in the vacancy among them. This artifact could be avoided if the unlabeled data structure is taken into consideration. The connections between similar data points will help the fusing of clusters become tighter and more effective.

We present *Smooth Neighbors on Teacher Graphs* (SNTG), which considers the neighboring structure to induce smoothness on the data manifold in some local clusters. By defining a teacher graph based on the targets generated by the teacher, our model encourages features to be invariant when some perturbations are added to the neighboring points on the graph. We propose a doubly stochastic sampling algorithm to reduce the computational cost with large mini-batch sizes. Our method can be easily incorporated into existing deep SSL methods including both generative and discriminative approaches because SNTG does not increase the number of network parameters. We demonstrate significant performance improvements over state-of-the-art results while the extra time cost is negligible. SNTG is still effective with few labels or noisy labels.

2 Our Approach

In this section, we formalize SNTG by answering two key questions: (1) How to define the graph and neighbors? (2) How to induce the smoothness of neighboring points using the graph?

Building the graph with the teacher model. Most existing graph-based SSL methods [1, 16] depend on a distance metric in the input space $\mathcal{X}$, which is typically low-level (e.g., pixel values of images). For natural images, pixel distance cannot reflect semantic similarity well. Instead, we use the distance in the label space $\mathcal{Y}$, and treat the data points from the same class as neighbors. However, an issue is that the true labels of unlabeled data are unknown. We address it by learning a teacher graph using the targets generated by the teacher model. Self-ensembling is a good choice for constructing the graph because the ensemble predictions are expected to be more accurate than the outputs of current classifier. Formally, for $x_i \in \mathcal{D}$, a target prediction $\tilde{f}(x_i)$ is given by the teacher defined in the previous section. Denote the *hard* target prediction as $\tilde{y}_i = \mathrm{argmax}_k[\tilde{f}(x_i)]_k$ where $[\cdot]_k$ is the k-th component of the vector, indicating the probability that x_i is of class k. We build the graph as follows: $W_{ij} = \mathbb{1}[\tilde{y}_i = \tilde{y}_j]$, where $\mathbb{1}$ is the indicator function. W_{ij} measures the similarity



(a) Π model (b) SNTG (c) Π model (d) SNTG

Figure 1: (a,b) are the embeddings of CIFAR-10 test data projected to 2-D using PCA. (c,d) are the 2-D embeddings of MNIST test data using t-SNE. Each color denotes a class.

between x_i and x_j and those pairs with nonzero entries are treated as "neighbors". Here we restrict $W_{ij} \in \{0, 1\}$ to construct a 0-1 sparse graph.

Guiding the low dimensional mapping. Generally, a deep classifier (i.e., the student) f can be decomposed as $f = g \circ h$, where $h : \mathcal{X} \to \mathbb{R}^p$ is the mapping from the input space to the penultimate layer and $g : \mathbb{R}^p \to [0, 1]^K$ is usually parameterized by a fully-connected layer with softmax. Due to the hierarchical nature of deep networks, $h(x)$ can be seen as a low-dimensional feature of the input. And the feature space is expected to be linearly separable, as shown in the common practice that a following linear classifier g suffices. In terms of approximating the semantic similarity of two instances, the Euclidean distance of $h(x_i)$ and $h(x_j)$ is more suitable than that of $f(x)$ which represents class probabilities. Hence we use the graph to guide $h(x)$, making them distinguishable among classes. Given a $N \times N$ similarity matrix W of the sparse graph, we define our SNTG loss as

$$R_S(\theta, \mathcal{L}, \mathcal{U}) = \sum_{x_i, x_j \in \mathcal{D}} \ell_G(h(x_i; \theta), h(x_j; \theta), W_{ij}) \tag{3}$$

Here we utilize the contrastive Siamese networks [3], which can learn an invariant mapping and perform well in metric learning and face verification [7, 14]. Specifically, the loss is defined as:

$$\ell_G = \begin{cases} \|h(x_i) - h(x_j)\|^2 & \text{if } W_{ij} = 1 \\ \max\left(0, m - \|h(x_i) - h(x_j)\|\right)^2 & \text{if } W_{ij} = 0 \end{cases} \tag{4}$$

where $m > 0$ is a pre-defined margin. The margin loss is to constrain neighboring points to have consistent embeddings. Consequently, the neighbors are encouraged to have consistent predictions while the non-neighbors (i.e., the points of different classes) are pushed apart from each other with a minimum distance m.

One interpretation of why the proposed method works well is that SNTG explores more information in the teacher and improves the target quality. The teacher graph leads to better abstract representations in a smooth and coherent feature space and then aids the student f to give more accurate predictions. In turn, an improved student contributes to a better teacher model which can provide more accurate targets. Another perspective is that SNTG implements the *manifold* assumption for classification which underlies the loss ℓ_G, i.e.,, the points of same class are encouraged to concentrate together on sub-manifolds. The perturbation-based methods only keep the decision boundaries far away from each unlabeled data point while our method encourages the unlabeled data points to form tighter clusters, leading the decision boundaries to locate between the clusters.

3 Experiments

We provide results on widely adopted benchmarks. Following [13], we randomly sample 100, 1000 and 4000 labels for MNIST, SVHN and CIFAR-10, respectively. We further test with fewer labels for non-augmented MNIST as well as SVHN and CIFAR-10 with standard data augmentation. The results are averaged over 10 runs with different seeds for data splits. We use the same network architecture and hyper-parameters to our baselines. Main results are presented in Tables 1, 2.Our method surpasses previous state-of-the-arts by a large margin. More results and details on experimental setup are in Appendix A.

Table 2: Errors(%) on MNIST without augmentation.

Models	20 labels	50 labels
FM-GAN [13]	16.77±4.52	2.21±1.36
Triple-GAN [10]	4.81±4.95	1.56±0.72
Π model [9]	6.32±6.90*	1.02±0.37*
Π+SNTG (**Ours**)	**1.36±0.78**	**0.94±0.42**

Note that for VAT, its best results are achieved with an additional entropy minimization (Ent) regularization [6], leading to a much stronger baseline. We evaluate our method under its best setting VAT+Ent, and observe a further improvement with SNTG, e.g., from 13.15% to 12.49% and from 10.55% to 9.89% on CIFAR-10 without or with augmentation, respectively. In fact, we observe that Ent can also improve the performance of other baselines if it was added along with SNTG. But to focus on the efficacy of SNTG, we did not illustrate the results here.

Fewer labels. Notably, as shown in Tables 2, 4 and 3, when labels are scarce, e.g., MNIST with 20 labels (only 2 each class), SVHN with 250 labels and CIFAR-10 with 1000 labels, the improvements are even more significant. Since the labeled data only accounts for a small part, adding a strong regularizer SNTG in this case helps the model learn faster by exploring the unlabeled data structure.

Noisy Labels. SNTG can not only benefit from unlabeled data, but also learn from noisy supervision. We did experiments on supervised SVHN to show the tolerance to incorrect labels. True labels on the training set are replaced by random labels following [9]. Fig. **??** shows that TempEns+SNTG retains over 93% accuracy even when 90% of the labels are noisy while TempEns alone only obtains 73% [9]. Thus, our SNTG regularization improves the robustness and generalization performance.

Visualization of embeddings We finally visualize the embeddings of our algorithm and Π model on test data under the same settings (CIFAR-10 with 4000 labels and MNIST with 100 labels, both without augmentation). Fig. 1 shows the representations $h(x) \in \mathbb{R}^{128}$ projected to 2 dimension using PCA or tSNE. The learned representations of our model are more concentrated within clusters and are potentially easier to separate for different classes. The visualization is also consistent with our assumption and analysis.

4 Conclusions and Future Work

We developed a simple but effective SNTG for SSL that builds graphs using the teacher model and enforces smoothness of neighbor points on it. Empirically, it outperforms all baselines and achieves new state-of-the-art results. As a byproduct, we also learn an invariant mapping on a low dimensional manifold. SNTG can handle extreme cases with fewer labels and noisy labels.

References

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7(Nov):2399–2434, 2006.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *NIPS*, 1994.

[4] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *NIPS*, 2017.

[5] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. In *ICLR*, 2017.

[6] Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *NIPS*, 2005.

[7] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006.

[8] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *NIPS*, 2014.

[9] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. In *ICLR*, 2017.

[10] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *NIPS*, 2017.

[11] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv preprint arXiv:1704.03976*, 2017.

[12] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *NIPS*, 2015.

[13] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NIPS*, 2016.

[14] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[15] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, 2017.

[16] Jason Weston, Frédéric Ratle, and Ronan Collobert. Deep learning via semi-supervised embedding. In *ICML*, 2008.

This paper is a short version. See `https://arxiv.org/pdf/1711.00258.pdf` for more details.

A Experimental Setup

MNIST. It contains 60,000 gray-scale training images and 10,000 test images from handwritten digits 0 to 9. The input images are normalized to zero mean and unit variance.

SVHN. Each example in SVHN is a 32×32 color house-number images and we only use the official 73,257 training images and 26,032 test images following previous work. The augmentation of SVHN is limited to random translation between $[-2, 2]$ pixels.

CIFAR-10. The CIFAR-10 dataset consists of 32×32 natural RGB images from 10 classes such as airplanes, cats, cars and horses. We have 50,000 training examples and 10,000 test examples. The input images are normalized using ZCA following previous work [9]. We use the standard way of augmenting the CIFAR-10 dataset including horizontal flips and random translations.

Training details. In Π model and TempEns based experiments, the network architectures (shown in Table 5) and the hyper-parameters are the same as our baselines. Following [9], we apply mean-only batch normalization with momentum 0.999 to all layers and use leaky ReLU with $\alpha = 0.1$. The network is trained for 300 epochs using Adam Optimizer with mini-batches of size $n = 100$ and maximum learning rate 0.003 (exceptions are that temporal ensembling for SVHN uses 0.001 and MNIST uses 0.0001). We use the default Adam momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also ramp up the learning rate and the regularization term during the first 80 epochs with weight $w(t) = \exp\left[-5(1 - \frac{t}{80})^2\right]$ and ramp down the learning rate during the last 50 epochs. The ramp down function is $\exp\left[-12.5(1 - \frac{300-t}{50})^2\right]$. The regularization coefficient of consistency loss R_C is $\lambda_1 = 100$ for Π model and $\lambda_1 = 30$ for temporal ensembling (exception is that SVHN with $L = 250$ uses $\lambda_1 = 50$).

For comparison with Mean Teacher and VAT, we keep the same architecture and hyper-parameters settings with the corresponding baselines [15, 11]. Their network architectures are the same as shown in Table 5 but differ in some hyper-parameters such as weight normalization, training epochs and mini-batch sizes, which are detailed in their papers. We just add our SNTG loss along with their regularization R_C and keep other settings unchanged.

SNTG loss only needs three extra hyper-parameters: the regularization parameter λ_2, the margin m and the number of sub-sampled pairs s. We fix m and s, only tune λ_2. In all our experiments, the margin m in Eq. (4) is set to $m = 1$ when $\|h(x_i) - h(x_j)\|^2$ is averaged by feature dimension p. We sample half the number of mini-batch size pairs of (x_i, x_j) for computing ℓ_G, e.g., $s = 50$ for mini-batch size $n = 100$. The regularization coefficient λ_2 for SNTG loss R_S is set to $k\lambda_1$ where k is chosen from $\{0.2, 0.4, 0.5, 0.6, 1.0\}$ using the validation set and we use default ratio $k = 0.4$ for most settings. SNTG does not increase the number of model parameters and the run time is almost the same to baselines.

Code reproducing our results will be available at `https://github.com/xinmei9322/SNTG` soon.

B Doubly stochastic sampling approximation

Our overall objective is the sum of two components. The first one is cross-entropy loss on the labeled set $\mathcal{L}$, and the second is the regularization term, which encourages the smoothness for each single point (i.e., R_C) as well as for the neighboring points (i.e., R_S). Alg. 1 presents the pseudo-code.

As our model belongs to deep networks, we train it using SGD [2] with mini-batches. We construct the sub-graph in a random mini-batch B of size n to estimate R_S in Eq. (3). We need to compute W_{ij} for all the data pairs $(x_i, x_j) \in B$, which is of size n^2 in total. Although this step is fast, the computation of $\|h(x_i) - h(x_j)\|$ related to W_{ij} is $O(p)$ and then the overall computational cost is $O(n^2 p)$, which is slow for large n. We instead use doubly stochastic sampled data pairs to construct W_{ij} for Eq. (4), which is still an unbiased estimation of R_S. Specifically, in each iteration, we sample a mini-batch B and then sub-sample $s \leq n^2$ data pairs S from B.

Algorithm 1 Mini-batch training of SNTG for SSL

Input: x_i, y_i, $w(t)$ = unsupervised weight ramp-up function
$f_\theta(x)$ = neural network with trainable parameters θ
for t in [1, numepochs] **do**
  for each minibatch B **do**
    $f_i \leftarrow f_\theta(x_{i\in B})$ evaluate network outputs
    $\tilde{f}_i \leftarrow \tilde{f}(x_{i\in B})$ given by the teacher model
    for (x_i, x_j) in a minibatch pairs S from B **do**
      Compute W_{ij} according to Section 2
    end for
    loss $\leftarrow -\frac{1}{|B|}\sum_{i\in(B\cap\mathcal{L})}\log[f_i]_{y_i}$
    $+w(t)\Big[\lambda_1\frac{1}{|B|}\sum_{i\in B} d(\tilde{f}_i, f_i) + \lambda_2\frac{1}{|S|}\sum_{i,j\in S}\ell_G(h(x_i), h(x_j), W_{ij})\Big]$
    update θ using optimizers
  end for
end for
return θ

Table 3: Error rates (%) on CIFAR-10 with standard augmentation, averaged over 10 runs.

Model	1000 labels	2000 labels	4000 labels	All labels
Supervised-only			34.85±1.65	6.05±0.15
Π model [9]	31.65±1.20*	17.57±0.44*	12.36±0.31	5.56±0.10
Π+SNTG (**ours**)	21.23±1.27	14.65±0.31	11.00±0.13	**5.19±0.14**
TempEns [9]	23.31±1.01*	15.64±0.39*	12.16±0.24	5.60±0.10
TempEns+SNTG (**ours**)	**18.41±0.52**	**13.64±0.32**	10.93±0.14	5.20±0.14
VAT [11]	–	–	11.36	5.81
VAT+Ent [11]	–	–	10.55	–
VAT+Ent+SNTG (**ours**)	–	–	**9.89±0.34**	–

Table 4: Error rates (%) on SVHN with translation augmentation, averaged over 10 runs.

Model	250 labels	500 labels	1000 labels	All labels
Supervised-only	42.65±2.68	22.08±0.73	14.46±0.71	2.81±0.07
Π model [9]	9.93±1.15*	6.65±0.53	4.82±0.17	2.54±0.04
Π+SNTG (**ours**)	5.07±0.25	4.52±0.30	**3.82±0.25**	**2.42±0.05**
TempEns [9]	12.62±2.91*	5.12±0.13	4.42±0.16	2.74±0.06
TempEns+SNTG (**ours**)	5.36±0.57	4.46±0.26	3.98±0.21	2.44±0.03
Mean Teacher [15]	4.35±0.50	4.18±0.27	3.95±0.19	2.50±0.05
Mean Teacher+SNTG (**ours**)	**4.29±0.23**	**3.99±0.24**	3.86±0.27	2.42±0.06
VAT [11]	–	–	5.42	–
VAT+Ent [11]	–	–	3.86	–
VAT+Ent+SNTG (**ours**)	–	–	3.83±0.22	–

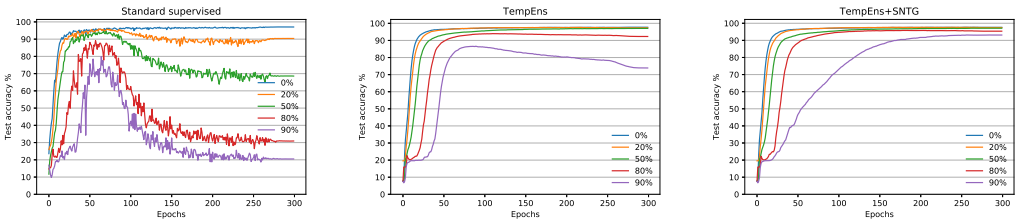


Figure 2: Test accuracy on supervised SVHN with noisy labels. Different colors denote the percentages of corrupted labels. With standard supervised training (left), the model suffers a lot and overfits to the incorrect information in labels. TempEns (middle) shows the resistance to the corruption but still has a drop in accuracy when the portion of randomized labels increases to 90%. Adding SNTG shows almost perfect robustness even when 90% labels are corrupted.

Table 5: The network architecture we used in all experiments.

Input: $32 \times 32 \times 3$ image ($28 \times 28 \times 1$ for MNIST)
Gaussian noise $\sigma = 0.15$
3×3 conv. 128 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 128 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 128 lReLU ($\alpha = 0.1$) same padding
2×2 max-pool, dropout 0.5
3×3 conv. 256 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 256 lReLU ($\alpha = 0.1$) same padding
3×3 conv. 256 lReLU ($\alpha = 0.1$) same padding
2×2 max-pool, dropout 0.5
3×3 conv. 512 lReLU ($\alpha = 0.1$) valid padding
1×1 conv. 256 lReLU ($\alpha = 0.1$)
1×1 conv. 128 lReLU ($\alpha = 0.1$)
Global average pool 6×6 (5×5 for MNIST) $\rightarrow 1 \times 1$
Fully connected $128 \rightarrow 10$ softmax

Data augmentation through space linearization

Grigorios G. Chrysos[1], **Yannis Panagakis**[1,2], **Stefanos Zafeiriou**[1]
[1] Department of Computing, Imperial College London, UK
[2] Department of Computer Science, Middlesex University London, UK
{g.chrysos, i.panagakis, s.zafeiriou}@imperial.ac.uk

Abstract

The state-of-the-art learning-based methods in machine learning tasks require a vast amount of samples to be trained. Such methods have hundreds of millions of parameters, which translates to more parameters than training samples to train them on. A standard approach to ameliorate the lack of labelled training samples is data augmentation (e.g. cropping, affine transformations). The two major categories of data augmentation methods are i) simple transformations (e.g. affine transformations, pixel intensity adaptations), ii) tailored to the task in hand (e.g. 3D model based). Instead, we propose to learn local transformations to synthesize new images that include the same scene/object with small adaptations. To that end, we introduce a three-stage approach that finds a low-dimensional (approximately) linear space. We learn a forward transformation from the image to the latent space, we perform a linear operation in this space and learn the inverse transformation (from latent to the image space) to synthesize a new image. We illustrate how this three-stage approach can be used to build powerful adaptive models for deformable facial tracking.

1 Introduction

In the era of deep learning, in which the scale of the dataset has a vast effect on the performance, different data augmentation methods are commonplace during training. These label-preserving transformations help to i) avoid over-fitting, ii) provide enough training samples for learning the millions of parameters (He et al. [2016]). We argue that the most commonly used augmentation methods are either simple (rotation, zoom) or task-specific (tailored for a handful of tasks) and propose instead a method that performs powerful local transformations to the image.

The data augmentation methods employed can be divided into i) image-level transformations, ii) model-based transformations. The first category includes affine transformations or pixel-level adjustments/noise, e.g. color perturbation (Krizhevsky et al. [2012]). Even though such methods can be applied in a variety of computer vision tasks, they consist a limited group of transformations in which partial invariance has been achieved in the past. The latter category, i.e. model-based transformations, e.g. the 3D profiling of Zhu et al. [2016] or the novel-view synthesis from 3D models (Rematas et al. [2014], Chandra et al. [2016]), typically require a very accurate 3D object model (Zhu et al. [2016], Tran et al. [2017]) or synthetic data (Rematas et al. [2014]). Such 3D models are available for only a small number of classes, while the realistic generation from 3D models/synthetic data is still an open problem (Bousmalis et al. [2017]), hence these model-based transformations are available for a subset of the computer vision tasks.

We argue that local transformations can be employed as a third category of augmentations. The code idea assumes the existence of an (approximately) linear low-dimensional space, which can describe the high dimensional nonlinear image space. We form a three-stage approach that learns the transformations from the image space to this latent space and enables us to perform a linear operation in the low-dimensional space. Specifically, i) we learn a nonlinear transformation that maps the image

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

to this low-dimensional space, then ii) we perform a linear transformation that maps the original latent representation to the representation of a slightly transformed image (e.g. a pair of successive frames in a video). As a third step, we learn the inverse transformation, i.e. mapping from the low-dimensional space to the transformed image, hence a linear operation in the latent space results in a nonlinear change in the image space. Both the forward (from image to latent space) and the inverse transformations are approximated by neural networks, specifically an Adversarial Autoencoder and a Generative Adversarial Network. Even though a generic model for augmenting all classes could be theoretically learned, we introduce object-specific transformations. We experimentally demonstrate how our proposed method can be used for augmenting the images of human faces[1].

2 Method

We want to find a low-dimensional and (approximately) linear space that can represent the object-specific attributes that our images contain. We learn both the transformation from the image space to this latent space, as well as the inverse transformation. Then, the latent representation $\boldsymbol{d}^{(t)}$ (corresponding to an arbitrary image $\boldsymbol{i}^{(t)}$) can be linearly transformed into a different representation $\boldsymbol{d}^{(t+x)}$, which we can transform back to the image space as $\boldsymbol{i}^{(t+x)}$. The newly synthesized image $\boldsymbol{i}^{(t+x)}$ differs from $\boldsymbol{i}^{(t)}$ with some local nonlinear transformation, i.e. the same object and scene are included, hence the original image has been augmented with a local transformation. We describe below the three-stage approach that achieves this augmentation.

2.1 Stage I: From image to latent representation

We follow an unsupervised learning approach for extracting the image representations. We utilize an Adversarial Autoencoder (Makhzani et al. [2015]). An Adversarial Autoencoder is composed of i) a generator (Autoencoder), ii) a discriminator. The discriminator aims in discerning the synthesized samples (outputs of generator) from the true data distribution, while the generator's task is to generate samples that resemble the true distribution's samples. The generator accepts an image $\boldsymbol{i}^{(t)}$, encodes it to $\boldsymbol{d}^{(t)}$ (we assume that the latent vector $\boldsymbol{d}^{(t)}$ lies in the latent space we want to find) and then decodes it to $\hat{\boldsymbol{i}^{(t)}}$.

2.2 Stage II: Linear transformation in the latent space

Our goal is to create a realistic synthetic image (which is a nonlinear transformation of image $\boldsymbol{i}^{(t)}$) by performing a linear transformation in the latent representation of $\boldsymbol{i}^{(t)}$. A simple but very powerful way to perform this change is to learn a linear regression model in the latent space. Assuming we have N pairs of correlated images available[2], i.e. $\boldsymbol{I} = \{(\boldsymbol{i}_k^{(t_k)}, \boldsymbol{i}_k^{(t_k+x_k)})\}$, we obtain (from Stage I) $\boldsymbol{\mathcal{D}} = \{(\boldsymbol{d}_k^{(t_k)}, \boldsymbol{d}_k^{(t_k+x_k)})\}$. This is formally expressed as:

$$\boldsymbol{d}^{(t_k+x_k)} = \boldsymbol{A} \cdot [\boldsymbol{d}^{(t_k)}; 1] + \boldsymbol{\epsilon} \tag{1}$$

where $\boldsymbol{\epsilon}$ is the noise; we compute $\boldsymbol{A}$ (closed-form solution) from real video samples.

2.3 Stage III: From latent to image representation

The last step consists in learning the transformation from the latent representation $\hat{\boldsymbol{d}}^{(t+x)}$ to the image representation $\boldsymbol{i}^{(t+x)}$. We utilize GAN (Goodfellow et al. [2014]) as they have demonstrated results of high visual quality (Ledig et al. [2017], Pathak et al. [2016]). The input to the GAN is the (transformed) latent representation $\hat{\boldsymbol{d}}^{(t+x)}$ along with noise, while the expected output is a (nonlinearly) transformed version of $\boldsymbol{i}^{(t)}$, i.e. an image $\hat{\boldsymbol{i}^{(t+x)}}$.

[1] The facial space is highly nonlinear, while such a method would have significant application in a host of face-related tasks.

[2] During the learning process those can be images from successive frames of a sequence of frames.

2.4 Prediction

Once the learning is completed, the structure for prediction is greatly simplified, see Fig. 1. Specifically, the image $\boldsymbol{i}^{(t)}$ is encoded (only the pre-trained encoder is used); the resulting representation $\boldsymbol{d}^{(t)}$ is multiplied by $\boldsymbol{A}$ to obtain $\hat{\boldsymbol{d}}^{(t+x)}$, while the last step consists of feeding $\hat{\boldsymbol{d}}^{(t+x)}$ to the GAN to synthesize a new image $\hat{\boldsymbol{i}}^{(t+x)}$.

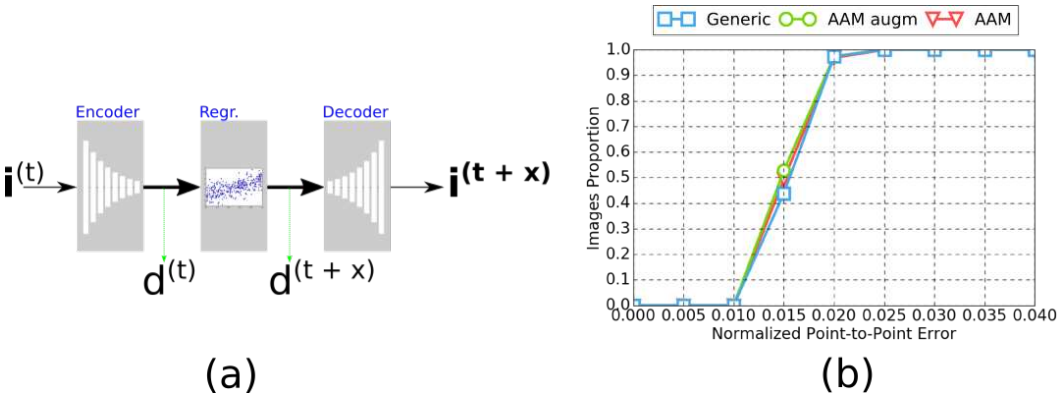


Figure 1: (Preferably viewed in color) (a) Prediction procedure from an image $\boldsymbol{i}^{(t)}$, (b) Cumulative Error plot comparing the adaptive model learned with and without augmentation (Sec. 3.1).

3 Experiments

To quantitatively validate our model, we have used our proposed method to augment the samples used for learning adaptive deformable models for tracking. Let us first provide the implementation details along with information about the datasets used.

The networks of Sec. 2.1 and Sec. 2.3 share the same architecture, i.e. the encoder/decoder in both cases is composed by 8 layers followed by batch normalization (Ioffe and Szegedy [2015]); both discriminators consist of 5 layers, while the latent dimension is fixed to 1024 for all cases.

The databases of Guo et al. [2016] (Stage I) and Chrysos and Zafeiriou [2017a] (Stages II and III) are utilized for the learning parts, while the database of Shen et al. [2015] for the experimental validation.

3.1 Augmented adaptive deformable models

Even though our data augmentation technique is not confined to geometric tasks, we indicate its application in building adaptive deformable models (Sánchez-Lozano et al. [2016], Chrysos and Zafeiriou [2017b]). Adaptive deformable models typically rely on very few samples to learn the statistics of the specific video. As already demonstrated in Chrysos and Zafeiriou [2017b] adaptive deformable models can surpass the performance of meticulously trained generic methods, hence if we can introduce additional invariances to the adaptive models, we could improve the performance of deformable tracking methods.

The experiment was conducted in a video of 300VW that was not used in any learning part. The video is comprised of more than 900 frames; the generic tracking was achieved by the state-of-the-art landmark localization of Yang et al. [2017]. Using K-Means (with clusters $K = 10$) in the extracted shapes, we have sampled one shape from each cluster and used those shapes and corresponding appearances to learn an Active Appearance Model (AAM) of Cootes et al. [2001]. Two different alternatives were considered: i) a model with the original shapes, ii) a model trained with our data augmentation method. The latter one included the original shapes/appearances along with their augmentations, i.e. each image was augmented and the new shape was extracted with Yang et al. [2017]. The model with the augmented shapes included at most two times the original number of shapes. These AAM models were then used to refine the shapes of the generic method. In Fig. 1 the CED curves comparing i) the original fitting results (denoted 'Generic'), ii) the AAM with the cluster shapes (denoted 'AAM'), iii) the AAM with the augmented shapes (denoted 'AAM augm') are

visualized. Even though very few frames were selected, both adaptive methods improve the results of the meticulously trained generic method. In addition, our augmented samples improve the AAM bases, hence result in an improved fitting.

4 Conclusion

In this work, we have introduced a three-stage approach for locating a latent space where a linear change in the image representation (in that space) results in nonlinear changes in the image space. The three proposed stages include a forward transformation (from the image to the representation space), a linear transformation in that space, an inverse transformation (from the transformed representation to the image space). We have experimentally validated that our model can be used for augmentation in facial images and illustrated how this can be used for augmenting the samples used for constructing adaptive deformable models.

Acknowledgement

The work of G. Chrysos has been funded by an Imperial College DTA. The work of Y. Panagakis is partially supported by the EPSRC project EP/N007743/1 (FACER2VM). The work of S. Zafeiriou has been partially funded by the FiDiPro program of Tekes (project number: 1849/31/2015), as well as the EPSRC project EP/N007743/1 (FACER2VM).

References

K. Bousmalis et al. Unsupervised pixel-level domain adaptation with generative adversarial networks. *CVPR*, 2017.

S. Chandra, G. Chrysos, and I. Kokkinos. Surface based object detection in rgbd images. In *BMVC*, 2016.

GG Chrysos and S. Zafeiriou. Deep face deblurring. *CVPR Workshops*, 2017a.

GG Chrysos and S. Zafeiriou. Pd^{2}t: Person-specific detection, deformable tracking. *T-PAMI*, 2017b.

TF Cootes, GJ Edwards, and CJ Taylor. Active appearance models. *T-PAMI*, 23(6):681–685, 2001.

I. Goodfellow et al. Generative adversarial nets. In *NIPS*, 2014.

Y. Guo et al. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV*, pages 87–102, 2016.

Kaiming He et al. Deep residual learning for image recognition. In *CVPR*. IEEE, 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

A. Krizhevsky et al. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

C. Ledig et al. Photo-realistic single image super-resolution using a generative adversarial network. *CVPR*, 2017.

A. Makhzani et al. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

D. Pathak et al. Context encoders: Feature learning by inpainting. In *CVPR*, pages 2536–2544, 2016.

K. Rematas et al. Image-based synthesis and re-synthesis of viewpoints guided by 3d models. In *CVPR*, 2014.

E. Sánchez-Lozano et al. Cascaded continuous regression for real-time incremental face tracking. In *ECCV*, 2016.

J. Shen et al. The first facial landmark tracking in-the-wild challenge: Benchmark and results. In *ICCV Workshops*, 2015.

Luan Tran et al. Disentangled representation learning gan for pose-invariant face recognition. In *CVPR*, 2017.

J. Yang et al. Stacked hourglass network for robust facial landmark localisation. In *CVPR Workshops*, 2017.

X. Zhu, Z. Lei, Z. Liu, H. Shi, and S Z Li. Face alignment across large poses: A 3d solution. In *CVPR*, 2016.

Efficient Multitask Feature and Relationship Learning

Han Zhao, Otilia Stretcu, Renato Negrinho, Alex Smola, Geoff Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
{han.zhao, ostretcu, negrinho, ggordon}@cs.cmu.edu
alex@smola.org

Abstract

We propose a multi-convex framework for multitask learning that improves predictions by learning relationships both between tasks and between features. Our framework is a generalization of related methods, that either learn task relationships, or feature relationships, but not both. We start with a hierarchical Bayesian model, and use the empirical Bayes method to transform the underlying inference problem into a multi-convex problem. To tackle the multi-convex optimization problem, we propose a block coordinate-wise minimization algorithm that has a closed form solution for each block subproblem. Naively these solutions would be expensive to compute, but by using the theory of doubly stochastic matrices, we are able to reduce the covariance learning subproblem to a minimum-weight perfect matching problem on a complete bipartite graph, and solve it analytically and efficiently. To solve the weight learning subproblem, we propose three different strategies that can be used no matter whether the instances are shared by multiple tasks or not. We demonstrate the efficiency of our method on both synthetic datasets and real-world datasets. Experiments show that the proposed optimization method is orders of magnitude faster than the previous projected gradient method, and our model is able to exploit the correlation structures among multiple tasks and features.

1 Introduction

Multitask learning has received considerable interest in the past decades [8, 11, 1, 2, 14, 16, 22, 21, 15]. One of the underlying assumptions behind many multitask learning algorithms is that the tasks are related to each other. Hence, a key question is how to define the notion of task relatedness, and how to capture it in the learning formulation. A common assumption is that tasks can be described by weight vectors, and that these vectors are sampled from a shared prior distribution [16, 22, 23]. Another strand of work assumes common feature representations to be shared among multiple tasks, and the goal is to learn the shared representation as well as task-specific parameters simultaneously [19, 8, 10, 2]. Moreover, when structure about multiple tasks is available, e.g., task-specific descriptors [6] or a task similarity graph [11], regularizers can often be incorporated into the formulation to explicitly penalize hypotheses that are not consistent with the given structure.

We propose a multi-convex framework for multitask learning. Our method improves predictions over tabula rasa learning by assuming that all the task vectors are sampled from a common, shared prior. There have been several attempts to improve predictions along this direction by either learning the relationships between different tasks [22], known as *Multitask Relationship Learning* (MTRL), or by exploiting the relationships between different features [2], which is known as *Multitask Feature Learning* (MTFL). Zhang and Schneider [21] proposed a multitask learning framework where both the task and feature relationships are inferred from data by assuming a sparse matrix-normal penalty on both the task and feature representations. Similar to their approach, our multitask learning framework

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

is a generalization of both MTRL and MTFL, which learns the relationships both between tasks and between features simultaneously. This property is favorable for applications where we not only aim for better generalization, but also seek to have a clear understanding about the relationships among different tasks. However, as we will see shortly, unlike the sparse regularization approach, our model makes no sparsity assumptions and leads to a much more efficient optimization method. We term our proposed framework *FEature and Task Relationship learning* (FETR).

Contributions. We propose a principled model for multitask learning that learn task and feature relationships simultaneously: both MTFL and MTRL can be viewed as special cases of FETR. On the optimization side, we design an efficient solver for the weight matrix with linear convergence rate guarantee, which is an exponential acceleration over existing works. Our algorithm computes the two covariance matrices in closed form, being the first that does not require any numerical iterative schemes. To the best of our knowledge, all existing works have to resort to expensive generic matrix optimization algorithms. We demonstrate the efficiency of the proposed optimization algorithm by comparing it with the projected gradient descent algorithm [21], showing that it always converges orders of magnitude faster and often converges to a better solution. We test the statistical performance of FETR on modeling real-world related tasks. Results show that FETR is not only able to give better predictions, but can also effectively exploit the correlation structures among multiple tasks, providing a better way to visualize the correlations between both the features and the tasks.

2 Multitask Feature and Relationship Learning

We consider the following setup. We are given m learning tasks $\{T_i\}_{i=1}^m$, where for each learning task T_i we have access to a training set $\mathcal{D}_i$ with n_i data instances $(\mathbf{x}_i^j, y_i^j), j \in [n_i]$. Here we focus on the supervised learning setting where $\mathbf{x}_i^j \in \mathcal{X}_i \subseteq \mathbb{R}^d$ and $y_i^j \in \mathcal{Y}_i$, where $\mathcal{Y}_i = \mathbb{R}$ for a regression problem and $\mathcal{Y}_i = \{1, -1\}$ for a binary classification problem. For ease of discussion, in what follows, we will assume our model for each task T_i to be a linear regression. However, our approach can also be translated into a classification setting where the predictor is a logistic regression. Due to space limit, we only provide proof sketches of the lemmas and theorems presented in the paper, and we refer interested readers to the appendix for detailed proofs.

In the linear regression model, the likelihood function for task i is given by: $y_i^j \mid \mathbf{x}_i^j, \mathbf{w}_i, \epsilon_i \sim \mathcal{N}(\mathbf{w}_i^T \mathbf{x}, \epsilon_i^2)$, where $\mathcal{N}(\mathbf{m}, \Sigma)$ is the multivariate normal distribution with mean $\mathbf{m}$ and covariance matrix Σ. Let $W = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathbb{R}^{d \times m}$ be the model parameter (regression weights) for m different tasks. Applying a hierarchical Bayes approach, we specify a prior distribution over the model parameter W. Specifically, we define the prior distribution over W to be

$$W \mid \xi, \Omega_1, \Omega_2 \sim \left(\prod_{i=1}^m \mathcal{N}(\mathbf{w}_i \mid \mathbf{0}, \xi_i \mathbf{I}_d) \right) q(W \mid \Omega_1, \Omega_2) \tag{1}$$

where $\mathbf{I}_d$ is a $d \times d$ identity matrix and $\mathbf{0} \in \mathbb{R}^d$ is a d-dimensional vector of all 0s. The form of $q(W \mid \Omega_1, \Omega_2)$ is given by $q(W \mid \Omega_1, \Omega_2) = \mathcal{MN}_{d \times m}(W \mid \mathbf{0}_{d \times m}, \Omega_1, \Omega_2)$, where $\mathcal{MN}_{d \times m}(M, A, B)$ denotes a matrix-variate normal distribution [13] with mean $M \in \mathbb{R}^{d \times m}$, row covariance matrix $A \in \mathbb{S}_{++}^d$ and column covariance matrix $B \in \mathbb{S}_{++}^m$.

Given the prior distribution over W and the likelihood function, a standard Bayesian inference approach applied here would be to specify another prior distribution over both covariance matrices Ω_1 and Ω_2 and then compute the above exact posterior distribution. However, this is computationally intractable in our case, so instead of computing the integration exactly, we take an empirical Bayes approach to approximate the intractable integration above by $p(W \mid X, \mathbf{y}) \approx \max_{\Omega_1, \Omega_2} p(\mathbf{y} \mid X, W) p(W \mid \Omega_1, \Omega_2)$, which leads us to the following optimization problem:

$$\underset{W, \Omega_1 \succ 0, \Omega_2 \succ 0}{\text{minimize}} \quad \sum_{i=1}^m \frac{1}{\xi_i^2} \mathbf{w}_i^T \mathbf{w}_i + m \log |\Omega_1| + d \log |\Omega_2| + \sum_{i=1}^m \frac{1}{\epsilon_i^2} \sum_{j=1}^{n_i} (y_i^j - \mathbf{w}_i^T \mathbf{x}_i^j)^2 + \text{tr}(\Omega_1^{-1} W \Omega_2^{-1} W^T) \tag{2}$$

It is worth pointing out here that the optimal value of (2) may not be achieved since the constraint set is open. We will fix this technical issue by imposing boundedness constraints on both Ω_1 and Ω_2. For simplicity of later discussion, we will assume that $\xi_i = \xi$ and $\epsilon_i = \epsilon$, $\forall i \in [m]$.

3 Multi-convex Optimization

It is not hard to see that the optimization problem in (2) is not convex since $m \log |\Omega_1| + d \log |\Omega_2|$ is a concave function of Ω_1 and Ω_2 [7]. Also, (2) is unbounded from below. To handle these technical issues, we introduce a boundedness constraint into the constraint set of Ω_1 and Ω_2: $\frac{1}{u} I_d \preceq \Omega_1 \preceq \frac{1}{l} I_d$ and $\frac{1}{u} I_m \preceq \Omega_2 \preceq \frac{1}{l} I_m$, where $u > l > 0$ are constants. Technically, the boundedness constraint make the feasible sets for Ω_1 and Ω_2 compact, hence by the extreme value theorem minimum is guaranteed to be achieved since the objective function is continuous. Next, we apply a well known transformation to both Ω_1 and Ω_2 so that the new optimization problem is multi-convex in terms of the transformed variables. We define $\Sigma_1 \triangleq \Omega_1^{-1}$ and $\Sigma_2 \triangleq \Omega_2^{-1}$. Both Σ_1 and Σ_2 are well-defined because Ω_1 and Ω_2 are constrained to be positive definite matrices. The transformed optimization formulation based on W, Σ_1 and Σ_2 is:

$$\begin{aligned} \underset{W,\Sigma_1,\Sigma_2}{\text{minimize}} \quad & \sum_{i=1}^{m} \sum_{j=1}^{n_i} (y_i^j - \mathbf{w}_i^T \mathbf{x}_i^j)^2 + \eta \sum_{i=1}^{m} \mathbf{w}_i^T \mathbf{w}_i - \rho(m \log |\Sigma_1| + d \log |\Sigma_2|) + \rho \text{tr}(\Sigma_1 W \Sigma_2 W^T) \\ \text{subject to} \quad & l I_d \preceq \Sigma_1 \preceq u I_d, l I_m \preceq \Sigma_2 \preceq u I_m \end{aligned} \tag{3}$$

where we define $\eta = (\epsilon/\xi)^2$ and $\rho = \epsilon^2$ to simplify the notation.

Proposition 3.1. The objective function in (3) is multi-convex.

Based on the multi-convex formulation, we propose a block coordinate-wise minimization algorithm to optimize the objective given in (3). In each iteration k we alternatively minimize over W with Σ_1 and Σ_2 fixed, then minimize over Σ_1 with W and Σ_2 fixed, and lastly minimize Σ_2 with W and Σ_1 fixed. The whole procedure is repeated until a stationary point is found or the decrease in the objective function is less than a pre-specified threshold. In what follows, we assume $n = n_i, \forall i \in [m]$ to simplify the notation. Let $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_m) \in \mathbb{R}^{n \times m}$ be the labeling matrix and $X \in \mathbb{R}^{n \times d}$ be the feature matrix shared by all the tasks. Using this notation, the objective function can be equivalently expressed in matrix form as:

$$\begin{aligned} \underset{W,\Sigma_1,\Sigma_2}{\text{minimize}} \quad & ||Y - XW||_F^2 + \eta ||W||_F^2 + \rho ||\Sigma_1^{1/2} W \Sigma_2^{1/2}||_F^2 - \rho(m \log |\Sigma_1| + d \log |\Sigma_2|) \\ \text{subject to} \quad & l I_d \preceq \Sigma_1 \preceq u I_d, l I_m \preceq \Sigma_2 \preceq u I_m \end{aligned} \tag{4}$$

3.1 Optimization w.r.t. W

In order to minimize over W when both Σ_1 and Σ_2 are fixed, we solve the following subproblem:

$$\underset{W}{\text{minimize}} \quad h(W) \triangleq ||Y - XW||_F^2 + \eta ||W||_F^2 + \rho ||\Sigma_1^{1/2} W \Sigma_2^{1/2}||_F^2 \tag{5}$$

This is an unconstrained convex optimization problem. We present three different algorithms to find the optimal solution of this subproblem. The first one guarantees to find an exact solution in closed form in $O(m^3 d^3)$ time, by using the isomorphism between $\mathbb{R}^{d \times m}$ and $\mathbb{R}^{dm}$. The second one does gradient descent with fixed step size to iteratively refine the solution, and we show that in our case a linear convergence rate can be guaranteed. The third one finds the optimal solution by solving the Sylvester equation [3] characterized by the first-order optimality condition.

Proposition 3.2. (5) can be solved in closed form in $O(m^3 d^3 + mnd^2)$ time; the optimal $\text{vec}(W^*)$ has the following form: $\left(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1\right)^{-1} \text{vec}(X^T Y)$.

W^* can then be obtained simply by reformatting $\text{vec}(W^*)$ into a $d \times m$ matrix. The computational bottleneck in the above procedure is in solving an $md \times md$ system of equations, which scales as $O(m^3 d^3)$ if no further sparsity structure is available. This can be intractable even for moderate m and d. In such cases, instead of computing an exact solution to (5), we can use gradient descent with fixed step size to obtain an approximate solution.

The objective function $h(W)$ is differentiable and its gradient can be obtained in $O(m^2 d + md^2)$ time as follows: $\nabla_W h(W) = X^T(Y - XW) + \eta W + \rho \Sigma_1 W \Sigma_2$. Let $\lambda_i(A)$ be the ith largest eigenvalue of a real symmetric matrix A. We provide a linear convergence guarantee for the gradient method in Thm. 3.1. Our proof technique is adapted from Nesterov [17] where we extend it to matrix function.

Theorem 3.1. Let $\lambda_l = \lambda_d(X^T X) + \eta + \rho l^2$, $\lambda_u = \lambda_1(X^T X) + \eta + \rho u^2$ and $\kappa = \lambda_u / \lambda_l$, the condition number. Choose $0 < t \leq 2/(\lambda_u + \lambda_l)$. For all $\varepsilon > 0$, gradient descent with step size t converges to the optimal solution within $O(\kappa \log(1/\varepsilon))$ steps.

The computational complexity to achieve an ε approximate solution using gradient descent is $O(nd^2 + \kappa \log(1/\varepsilon)(m^2 d + md^2))$. Compared with the $O(m^3 d^3 + mnd^2)$ complexity for the exact solution, the gradient descent algorithm scales much better provided the condition number κ is not too large. As a side note, when the condition number is large, we can also effectively reduce the iteration complexity to $O(\sqrt{\kappa} \log(1/\varepsilon))$ by using the conjugate gradient method [18].

A Sylvester equation [5] is a matrix equation of the form $AX + XB = C$, where the goal is to find a solution matrix X given A, B and C. For this problem, there are efficient numerical algorithms with highly optimized implementations that can obtain a solution within cubic time. For example, the Bartels-Stewart algorithm [3] solves the Sylvester equation by first transforming A and B into Schur forms by QR factorization, and then solves the resulting triangular system via back-substitution. Our third approach is based on the observation that we can equivalently transform the first-order optimality equation into a Sylvester equation by multiplying both sides of the equation by Σ_1^{-1}:

$$\Sigma_1^{-1}(X^T X + \eta I_d)W + W(\rho \Sigma_2) = \Sigma_1^{-1} X^T Y \tag{6}$$

Then finding the optimal solution of the subproblem amounts to solving the above Sylvester equation. Specifically, the solution can be obtained using the Bartels-Stewart algorithm in $O(m^3 + d^3 + nd^2)$.

Remark. Both the gradient descent and the Bartels-Stewart algorithm find optimal solutions in cubic time. However, the gradient descent algorithm is more widely applicable than the Bartels-Stewart algorithm: the Bartels-Stewart algorithm only applies to the case where all the tasks share the same instances, so that we can write down the matrix equation explicitly, while gradient descent can be applied where each task has different number of inputs and those inputs are not shared among tasks. On the other hand, as we will see shortly in the experiments, in practice the Bartels-Stewart algorithm is faster than gradient descent, and provides a more numerically stable solution.

3.2 Optimization w.r.t. Σ_1 and Σ_2

Before we delve into the detailed analysis below, we first list the final algorithms used to optimize Σ_1 and Σ_2 in Alg. 1 and Alg. 2, respectively. They are remarkably simple: each algorithm only involves one SVD, one truncation and two matrix multiplications. The computational complexities of Alg. 1 and Alg. 2 are bounded by $O(m^2 d + md^2 + d^3)$ and $O(m^2 d + md^2 + m^3)$, respectively.

Algorithm 1 Minimize Σ_1

Input: W, Σ_2 and l, u.
1: $[V, \nu] \leftarrow \text{SVD}(W \Sigma_2 W^T)$.
2: $\lambda \leftarrow \mathbb{T}_{[l,u]}(m/\nu)$.
3: $\Sigma_1 \leftarrow V \text{diag}(\lambda) V^T$.

Algorithm 2 Minimize Σ_2

Input: W, Σ_1 and l, u.
1: $[V, \nu] \leftarrow \text{SVD}(W^T \Sigma_1 W)$.
2: $\lambda \leftarrow \mathbb{T}_{[l,u]}(d/\nu)$.
3: $\Sigma_2 \leftarrow V \text{diag}(\lambda) V^T$.

Due to the symmetric roles of Σ_1 and Σ_2 in (4), we focus on analyzing the optimization w.r.t. Σ_1. A completely symmetric analysis can be applied to Σ_2 as well. In order to minimize over Σ_1 when W and Σ_2 are fixed, we solve the following subproblem:

$$\underset{\Sigma_1}{\text{minimize}} \quad \text{tr}(\Sigma_1 W \Sigma_2 W^T) - m \log |\Sigma_1|, \qquad \text{subject to} \quad l I_d \preceq \Sigma_1 \preceq u I_d \tag{7}$$

Although (7) is a convex optimization problem, it is computationally expensive to solve using off-the-shelf algorithms, e.g., the interior point method, because of the constraints as well as the iterative nature of these numerical approximate schemes. Surprisingly, we can find an *exact and closed form* optimal solution to this problem, using tools from the theory of doubly stochastic matrices [9] and reducing (7) into a minimum-weight bipartite graph matching problem. Due to the space limit, we defer our detailed derivation and the proof of correctness of Alg. 1 and Alg. 2 into appendix. The soft-thresholding operator $\mathbb{T}_{[l,u]}(z)$ used in line 2 of both algorithms are defined as: $\mathbb{T}_{[l,u]}(z) = z$ if $z \in [l, u]$, $\mathbb{T}_{[l,u]}(z) = l$ if $z < l$ and $\mathbb{T}_{[l,u]}(z) = u$ if $z > u$.

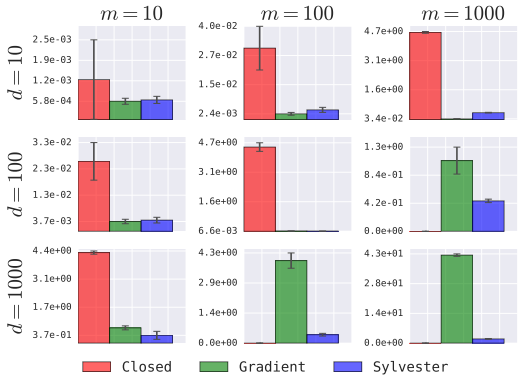
4 Experiments

4.1 Convergence Analysis

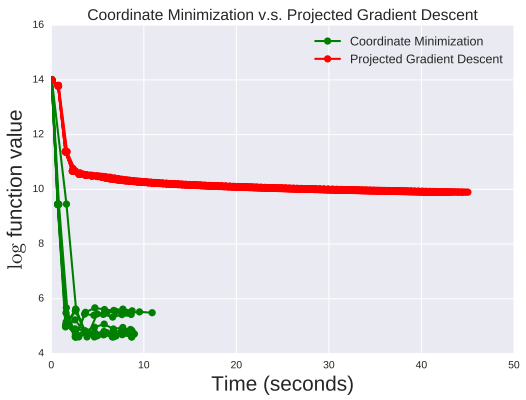
We first investigate the efficiency and scalability of the three different algorithms for minimizing w.r.t. W on synthetic data sets. For each experiment, we generate a synthetic data set which consists

of $n = 10^4$ instances that are shared among all the tasks. All the instances are randomly sampled uniformly from $[0, 1]^d$. We gradually increase the dimension of features, d, and the number of tasks, m to test scalability. The first algorithm implements the closed form solution by explicitly computing the $md \times md$ tensor product matrix and then solving the linear system. The second one is the proposed gradient descent, and the last one uses the Bartels-Stewart algorithm to solve the equivalent Sylvester equation to compute W. We use open source toolkit `scipy` whose backend implementation uses highly optimized Fortran code. For all the synthetic experiments we set $l = 0.01$ and $u = 100$, which corresponds to a condition number of 10^4. We fix the coefficients $\eta = 1.0$ and $\rho = 1.0$. The experimental results are shown in Fig. 1a.

As expected, the closed form solution does not scale to problems of even moderate size due to its large memory requirement. In practice the Bartels-Stewart algorithm is about one order of magnitude faster than the gradient descent method when either m or d is large. It is also worth pointing out here that the Bartels-Stewart algorithm is the most numerically stable algorithm among the three based on our observations. We also compare our proposed coordinate minimization algorithm with the previous projected gradient method to solve the optimization problem (4). Specifically, the projected gradient method updates W, Σ_1 and Σ_2 in each iteration and then projects Σ_1 and Σ_2 onto the corresponding feasible regions. In this experiment we set the number of instances to be 10^4, the dimension of feature vectors to be 10^4 and the number of tasks to be 10. All the instances are shared among all the tasks, so that the Sylvester solver is used to optimize W in coordinate minimization. We repeat the experiments 10 times and report the log function values versus the time used by these two algorithms (Fig. 1b). It is clear from this synthetic experiment that our proposed algorithm not only converges much faster than the projected gradient descent, but also achieves better results.



(a) Each experiment is repeated 10 times. We plot the mean run time (seconds) and the standard deviation under each experimental configuration. The closed form solution does not scale when $md \geq 10^4$.



(b) The convergence speed of coordinate minimization versus projected gradient descent. All the experiments are repeated 10 times.

Figure 1: Comparisons of convergence speed of different algorithms on synthetic datasets.

Table 1: Mean squared error on the SARCOS data and the mean of normalized mean squared error (MNMSE) on the school dataset across 10-fold cross-validation. For the SARCOS data, each column corresponds to one task (DOF).

	SARCOS							School
Method	1st	2nd	3rd	4th	5th	6th	7th	MNMSE
STL	31.40	22.90	9.13	10.30	0.14	0.84	0.46	0.9882 ± 0.0196
MTFL	31.41	22.91	9.13	10.33	0.14	**0.83**	0.45	0.8891 ± 0.0380
MTRL	31.09	22.69	**9.08**	9.74	0.14	**0.83**	0.44	0.9007 ± 0.0407
SPARSE	31.13	**22.60**	9.10	9.74	**0.13**	**0.83**	0.45	0.8451 ± 0.0197
FETR	**31.08**	22.68	**9.08**	**9.73**	**0.13**	**0.83**	**0.43**	$\mathbf{0.8134 \pm 0.0253}$

4.2 Results on Benchmark Datasets

Robot Inverse Dynamics This data relates to an inverse dynamics problem for a seven degree-of-freedom (DOF) SARCOS anthropomorphic robot arm [20]. The goal of this task is to map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. Hence there are 7 tasks and the inputs are shared among all the tasks.

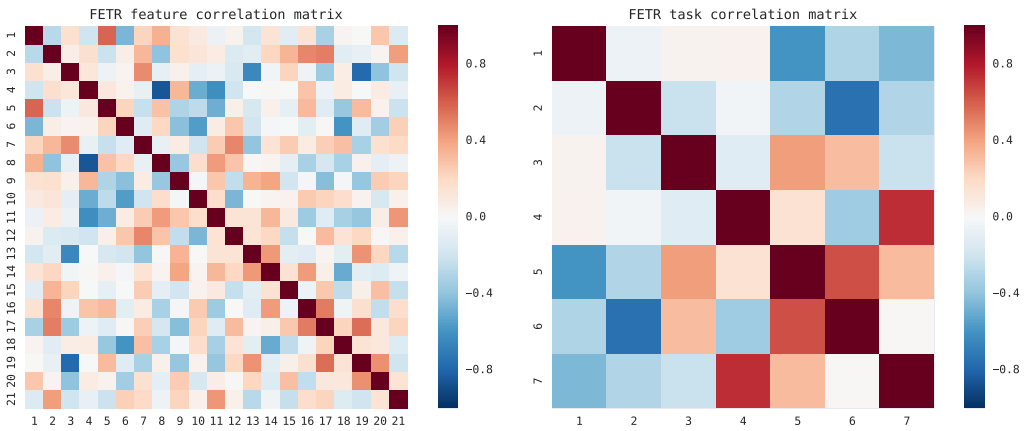


Figure 2: Visualization of the correlation matrices of feature and task vectors learned by FETR.

The training set and test set contain 44,484 and 4,449 examples, respectively. We further partition the training set into a development set and a validation set, which contain 31,138 and 13,346 instances.

School Data This dataset consists of the examination scores of 15,362 students from 139 secondary schools [12]. It has 27 input features, and contains 139 tasks. In the school dataset, instances are not shared among different tasks, hence we use our gradient descent solver for W instead of the Sylvester equation solver. We use 10-fold cross-validation to generate training and test datasets, and for each partition we compute the mean of normalized mean squared error (MNMSE) over 139 tasks. The normalized mean squared error is defined as the ratio of the MSE and the variance on a task. We show the mean MNMSE and its standard deviation across 10 cross-validation folds.

We compare FETR with multitask feature learning [10] (MTFL), multitask relationship learning [22] (MTRL), and the sparse multitask learning variant [21] (SPARSE). Both MTFL and MTRL can be treated as different special cases of our model, while SPARSE is a variant of FETR which assumes sparsity structure of both covariance matrices without the boundedness constraints. We use ridge regression as our baseline model, and denote it as single task learning (STL). All the methods share the same experimental setting, including model selection. In all the experiments we fix $l = 10^{-3}$ and $u = 10^3$. The hyperparameters range from $\eta, \rho \in \{10^{-5}, \ldots, 10^2\}$, and we use the validation set for model selection. For each method, the best model on the validation set is used to do prediction. The results for both datasets are summarized in Table 1 (the lower the better). Among all the methods, FETR consistently achieves lower test set MSEs. FETR can learn both covariance matrices over features and tasks simultaneously, while the other two methods can only estimate one of them. We show the covariance matrices estimated by FETR in Fig. 2. As we can see, the task correlation matrix learned by FETR successfully exhibits a block diagonal structure of the underlying task, where the last three torques are positively correlated with each other, while the first four and the last three torques are negatively correlated.

5 Conclusion

We develop a multi-convex framework for multitask learning that improves predictions by learning relationships both between tasks and between features. Our framework admits a multi-convex formulation, which allows us to design an efficient block coordinate-wise algorithm to optimize. To solve the weight learning subproblem, we propose three different strategies that can be used no matter whether the instances are shared by multiple tasks or not. By using the theory of doubly stochastic matrices, we are able to reduce the underlying matrix optimization subproblem to a minimum weight perfect matching problem, and solve it exactly in closed form. To the best of our knowledge, all the previous works have to resort to expensive iterative schemes to solve this problem. Experiments show that our method is orders of magnitude faster than the previous projected gradient descent method.

References

[1] A. Argyriou, M. Pontil, Y. Ying, and C. A. Micchelli. A spectral regularization framework for multi-task structure learning. In *Advances in neural information processing systems*, 2007.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[3] R. H. Bartels and G. Stewart. Solution of the matrix equation AX+ XB= C [F4]. *Communications of the ACM*, 15(9):820–826, 1972.

[4] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[5] R. Bhatia and P. Rosenthal. How and why to solve the operator equation ax- xb= y. *Bulletin of the London Mathematical Society*, 29(01):1–21, 1997.

[6] E. V. Bonilla, F. V. Agakov, and C. Williams. Kernel multi-task learning using task-specific features. In *International Conference on Artificial Intelligence and Statistics*, pages 43–50, 2007.

[7] S. Boyd and L. Vandenberghe. Convex optimization. 2004.

[8] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[9] F. Dufossé and B. Uçar. Notes on birkhoff–von neumann decomposition of doubly stochastic matrices. *Linear Algebra and its Applications*, 497:108–115, 2016.

[10] A. Evgeniou and M. Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007.

[11] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[12] H. Goldstein. Multilevel modelling of survey data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 40(2):235–244, 1991.

[13] A. K. Gupta and D. K. Nagar. *Matrix variate distributions*, volume 104. CRC Press, 1999.

[14] T. Kato, H. Kashima, M. Sugiyama, and K. Asai. Multi-task learning via conic programming. In *Advances in Neural Information Processing Systems*, pages 737–744, 2008.

[15] Y. Li, X. Tian, T. Liu, and D. Tao. Multi-task model and feature joint learning. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3643–3649. AAAI Press, 2015.

[16] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient l 2, 1-norm minimization. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 339–348. AUAI Press, 2009.

[17] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[18] J. R. Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[19] S. Thrun. Explanation-based neural network learning. In *Explanation-Based Neural Network Learning*, pages 19–48. Springer, 1996.

[20] S. Vijayakumar and S. Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1079–1086. Morgan Kaufmann Publishers Inc., 2000.

[21] Y. Zhang and J. G. Schneider. Learning multiple tasks with a sparse matrix-normal penalty. In *Advances in Neural Information Processing Systems*, pages 2550–2558, 2010.

[22] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. 2010.

[23] Y. Zhang and D.-Y. Yeung. Multi-task learning using generalized t process. In *AISTATS*, pages 964–971, 2010.

A Derivation on Solving Covariance Matrices

Since $\Sigma_2 \in \mathbb{S}^m_{++}$, it follows that $W\Sigma_2 W^T \in \mathbb{S}^d_+$. Without loss of generality, we can reparametrize $\Sigma_1 = U\Lambda U^T$, where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_d)$ with $u \geq \lambda_1 \geq \lambda_2 \cdots \geq \lambda_d \geq l$ and $U \in \mathbb{R}^{d\times d}$ with $U^TU = UU^T = I_d$ using spectral decomposition. Similarly, we can represent $W\Sigma_2 W^T = VNV^T$ where $V \in \mathbb{R}^{d\times d}$, $V^TV = VV^T = I_d$ and $N = \text{diag}(\nu_1, \ldots, \nu_d)$ with $0 \leq \nu_1 \leq \cdots \leq \nu_d$. Note that the eigenvectors in N corresponds to eigenvalues in increasing order rather than decreasing order, for reasons that will become clear below.

Using the new representation and realizing that U is an orthonormal matrix, we have

$$\log|\Sigma_1| = \log|U\Lambda U^T| = \log|\Lambda|$$

and

$$\text{tr}(\Sigma_1 W\Sigma_2 W^T) = \text{tr}(\Lambda U^T VNV^T U)$$

Set $K = U^TV$. Since both U and V are orthonormal matrices, K is also an orthonormal matrix. We can further transform $\text{tr}(\Lambda U^T VNV^T U)$ to be

$$\text{tr}(\Lambda U^T VNV^T U) = \text{tr}((\Lambda K)(KN)^T)$$

Note that the mapping between U and K is bijective since V is a fixed orthonormal matrix. Combining all the results above, we can equivalently transform the optimization problem (7) into the following new form:

$$\begin{aligned} \underset{K,\Lambda}{\text{minimize}} \quad & -m\log|\Lambda| + \text{tr}((\Lambda K)(KN)^T) \\ \text{subject to} \quad & l\,\text{diag}(\mathbf{1}_d) \leq \Lambda \leq u\,\text{diag}(\mathbf{1}_d) \\ & K^TK = KK^T = I_d \end{aligned} \tag{8}$$

where $\mathbf{1}_d$ is a d-dimensional vector of all ones.

At first glance one may question that the new form of optimization is more complicated to solve since it is not even a convex problem due to the quadratic equality constraint. However, as we will see shortly, the new form helps to decouple the interaction between K and Λ in that K does not influence the first term $-m\log|\Lambda|$. This implies that we can first partially optimize over K, finding the optimal solution as a function of Λ, and then optimize over Λ. Mathematically, it means:

$$\begin{aligned} & \min_{K,\Lambda} -m\log|\Lambda| + \text{tr}((\Lambda K)(KN)^T) \\ \Leftrightarrow & \min_{\Lambda} -m\log|\Lambda| + \min_{K} \text{tr}((\Lambda K)(KN)^T) \end{aligned} \tag{9}$$

Consider the sub-minimization problem over K:

$$\text{tr}((\Lambda K)(KN)^T) = \sum_{i=1}^d \sum_{j=1}^d \lambda_i K_{ij}^2 \nu_j = \lambda^T P \nu$$

where we define $P = K \circ K$, i.e., the elementwise product of K, $\lambda = (\lambda_1, \cdots, \lambda_d)^T$ and $\nu = (\nu_1, \cdots, \nu_d)^T$. Since K is an orthonormal matrix, we have the following two equations hold:

$$\sum_{j=1}^d P_{ij} = \sum_{j=1}^d K_{ij}^2 = 1, \sum_{i=1}^d P_{ij} = \sum_{i=1}^d K_{ij}^2 = 1,\ \forall i,j \in [d]$$

which implies that P is a doubly stochastic matrix [9]. The partial minimization over K can be equivalently solved by the partial minimization over P:

$$\min_K \text{tr}((\Lambda K)(KN)^T) = \min_P \lambda^T P \nu \tag{10}$$

Note that the above minimization problem is a linear program of a doubly stochastic matrix P over its feasible set, which is known as the *Birkhoff polytope*. In order to solve it, we need to introduce the following theorems:

Theorem A.1 (Optimality of extreme points [4])**.** Consider the minimization of a linear programming problem over a polyhedron $\mathcal{P}$. Suppose that $\mathcal{P}$ has at least one extreme point and that there exists an optimal solution. Then there exists an optimal solution which is an extreme point of $\mathcal{P}$.

Definition A.1 (Birkhoff polytope)**.** The *Birkhoff polytope* B_d is the set of $d \times d$ doubly stochastic matrices. B_d is a convex polytope.

Theorem A.2 (Birkhoff-von Neumann theorem)**.** Let B_d be the Birkhoff polytope. B_d is the convex hull of the set of $d \times d$ permutation matrices. Furthermore, the vertices (extreme points) of B_d are the permutation matrices.

Lemma A.1. There exists an optimal solution P to the optimization problem (10) that is a $d \times d$ permutation matrix.

Given that there exists an optimal solution that is a permutation matrix, we can reduce (10) into a minimum-weight perfect matching problem on a complete bipartite graph:

Definition A.2 (Minimum-weight perfect matching)**.** Let $G = (V, E)$ be an undirected graph with edge weight $w : E \to \mathbb{R}_+$. A *matching* in G is a set $M \subseteq E$ such that no two edges in M have a vertex in common. A matching M is called perfect if every vertex from V occurs as the endpoint of some edge in M. The weight of a matching M is $w(M) = \sum_{e \in M} w(e)$. A matching M is called a minimum-weight perfect matching if it is a perfect matching that has the minimum weight among all the perfect matchings of G.

We construct the graph as follows: for any $\lambda, \nu \in \mathbb{R}^d_+$, we can construct a weighted $d - d$ bipartite graph $G = (V_\lambda, V_\nu, E, w)$ as follows:

- For each λ_i, construct a vertex $v_{\lambda_i} \in V_\lambda, \forall i$.
- For each ν_j, construct a vertex $v_{\nu_j} \in V_\nu, \forall j$.
- For each pair $(v_{\lambda_i}, v_{\nu_j})$, construct an edge $e(v_{\lambda_i}, v_{\nu_j}) \in E$ with edge wight $w(e(v_{\lambda_i}, v_{\nu_j})) = \lambda_i \nu_j$.

Theorem A.3. The minimum value of (10) is equal to the minimum weight of a perfect matching on $G = (V_\lambda, V_\nu, E, w)$. Furthermore, the optimal solution P of (10) can be constructed from the minimum-weight perfect matching on G.

Perhaps the most interesting and surprising part is that, by a combinatorial analysis, we do not even need to run standard matching algorithms to solve our matching problem!

Theorem A.4. Let $\lambda = (\lambda_1, \ldots, \lambda_d)$ and $\nu = (\nu_1, \ldots, \nu_d)$ with $\lambda_1 \geq \cdots \geq \lambda_d$ and $\nu_1 \leq \cdots \leq \nu_d$. The minimum-weight perfect matching on G is $\pi^* = \{(v_{\lambda_i}, v_{\nu_i}) : 1 \leq i \leq d\}$ with the minimum weight $w(\pi^*) = \sum_{i=1}^d \lambda_i \nu_i$.

The proof of this theorem depends on an exchange lemma we prove in appendix (B.5) and uses induction on d. The permutation matrix that achieves the minimum weight is $P^* = I_d$ since $\pi^*(\lambda_i) = \nu_i$. Note that $P = K \circ K$, it follows that the optimal K^* is also I_d. Hence we can solve for the optimal U^* matrix by solving the equation $U^{*T} V = I_d$, which leads to $U^* = V$.

Now plug in the optimal $K^* = I_d$ into (9). The optimization w.r.t. Λ decomposes into d independent optimization problems, each of which being a simple scalar optimization problem:

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \quad & \sum_{i=1}^{d} \lambda_i \nu_i - m \log \lambda_i \\ \text{subject to} \quad & l \leq \lambda_i \leq u, \quad \forall i = 1, \ldots, d \end{aligned} \tag{11}$$

Depending on whether the value m/ν_i is within the range $[l, u]$, the optimal solution λ_i^* for each scalar minimization problem may take different forms. Define a soft-thresholding operator $\mathbb{T}_{[l,u]}(z)$:

$$\mathbb{T}_{[l,u]}(z) = \begin{cases} l, & z < l \\ z, & l \leq z \leq u \\ u, & z > u \end{cases} \tag{12}$$

Using this soft-thresholding operator, we can express the optimal solution λ_i^* as $\lambda_i^* = \mathbb{T}_{[l,u]}(m/\nu_i)$, as shown in the pseudocode listed at the beginning of this section to optimize Σ_1 and Σ_2.

B Proofs

B.1 Proofs of Proposition 3.1

Proposition 3.1. The objective function in (3) is multi-convex.

Proof. First, it is straightforward to check that the constraint set $lI_d \preceq \Sigma_1 \preceq uI_d$ and $lI_m \preceq \Sigma_2 \preceq uI_m$ are convex. For any fixed Σ_1 and Σ_2, the objective function in terms of W can be expressed as

$$\sum_{i=1}^{m}\left(\sum_{j=1}^{n_i}(y_i^j - \mathbf{w}_i^T\mathbf{x}_i^j)^2 + \eta||\mathbf{w}_i||_2^2\right) + \rho\text{tr}(\Sigma_1 W \Sigma_2 W^T)$$

The first term decomposes over tasks and for each task vector $\mathbf{w}_i$, $\sum_{j=1}^{n_i}(y_i^j - \mathbf{w}_i^T\mathbf{x}_i^j)^2 + \eta||\mathbf{w}_i||_2^2$ is a quadratic function for each $\mathbf{w}_i$, so the summation is also convex in W. Since $\Sigma_1 \in \mathbb{S}_+^d$ and $\Sigma_2 \in \mathbb{S}_+^m$, we can further rewrite the second term above as

$$\rho\text{tr}(\Sigma_1 W \Sigma_2 W^T) = \rho\text{tr}((\Sigma_1^{1/2} W \Sigma_2^{1/2})(\Sigma_1^{1/2} W \Sigma_2^{1/2})^T) = \rho||\Sigma_1^{1/2} W \Sigma_2^{1/2}||_F^2$$

This is the Frobenius norm of the matrix $\Sigma_1^{1/2} W \Sigma_2^{1/2}$, which is a linear transformation of W when both Σ_1 and Σ_2 are fixed, hence this is also a convex function of W. Overall the objective function with respect to W when both Ω_1 and Ω_2 are fixed is convex. For any fixed W and Σ_2, consider the objective function with respect to Σ_1:

$$-\rho m \log|\Sigma_1| + \rho\text{tr}(\Sigma_1 C)$$

where $C = W\Sigma_2 W^T$ is a constant matrix. Since $\log|\Sigma_1|$ is concave in Σ_1 and $\text{tr}(\Sigma_1 C)$ is a linear function of Σ_1, it directly follows that $-\rho m \log|\Sigma_1| + \rho\text{tr}(\Sigma_1 C)$ is a convex function of Σ_1. A similar argument can be applied to Σ_2 as well. ■

B.2 Proof of Proposition 3.2

Proposition 3.2. (5) can be solved in closed form in $O(m^3d^3 + mnd^2)$ time; the optimal $\text{vec}(W^*)$ has the following form: $\left(I_m \otimes (X^TX) + \eta I_{md} + \rho\Sigma_2 \otimes \Sigma_1\right)^{-1}\text{vec}(X^TY)$.

To prove this claim, we need the following facts about tensor product:

Fact B.1. Let A be a matrix. Then $||A||_F = ||\text{vec}(A)||_2$.

Fact B.2. Let $A \in \mathbb{R}^{m_1\times n_1}$, $B \in \mathbb{R}^{n_1\times n_2}$ and $C \in \mathbb{R}^{n_2\times m_2}$. Then $\text{vec}(ABC) = (C^T \otimes A)\text{vec}(B)$.

Fact B.3. Let $S_1 \in \mathbb{R}^{m_1\times n_1}, S_2 \in \mathbb{R}^{n_1\times p_1}$ and $T_1 \in \mathbb{R}^{m_2\times n_2}, T_2 \in \mathbb{R}^{n_2\times p_2}$. Then $(S_1 \otimes S_2)(T_1 \otimes T_2) = (S_1S_2) \otimes (T_1T_2)$.

Fact B.4. Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$. Let $\{\mu_1, \ldots, \mu_n\}$ be the spectrum of A and $\{\nu_1, \ldots, \nu_m\}$ be the spectrum of B. Then the spectrum of $A \otimes B$ is $\{\mu_i\nu_j : 1 \le i \le n, 1 \le j \le m\}$.

we can show the following result by transforming W into its isomorphic counterpart:

Proof.

$$\begin{aligned}
&||Y - XW||_F^2 + \eta||W||_F^2 + \rho||\Sigma_1^{1/2} W \Sigma_2^{1/2}||_F^2 \\
&= ||\text{vec}(Y - XW)||_2^2 + \eta||\text{vec}(W)||_2^2 + \rho||\text{vec}(\Sigma_1^{1/2} W \Sigma_2^{1/2})||_2^2 && \text{(By Fact B.1)} \\
&= ||\text{vec}(Y) - (I_m \otimes X)\text{vec}(W)||_2^2 + \eta||\text{vec}(W)||_2^2 + \rho||(\Sigma_2^{1/2} \otimes \Sigma_1^{1/2})\text{vec}(W)||_2^2 && \text{(By Fact B.2)} \\
&= \text{vec}(W)^T\left((I_m \otimes X)^T(I_m \otimes X) + \eta I_{md} + \rho(\Sigma_2^{1/2} \otimes \Sigma_1^{1/2})^T(\Sigma_2^{1/2} \otimes \Sigma_1^{1/2})\right)\text{vec}(W) \\
&\quad - 2\text{vec}(W)^T(I_m \otimes X^T)\text{vec}(Y) + \text{vec}(Y)^T\text{vec}(Y) \\
&= \text{vec}(W)^T\left((I_m \otimes X^TX) + \eta I_{md} + \rho(\Sigma_2 \otimes \Sigma_1)\right)\text{vec}(W) \\
&\quad - 2\text{vec}(W)^T(I_m \otimes X^T)\text{vec}(Y) + \text{vec}(Y)^T\text{vec}(Y) && \text{(By Fact B.3)}
\end{aligned}$$

The last equation above is a quadratic function of vec(W), from which we can read off that the optimal solution W^* should satisfy:

$$\text{vec}(W^*) = \left(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1\right)^{-1} \text{vec}(X^T Y) \tag{13}$$

W^* can then be obtained simply by reformatting vec(W^*) into a $d \times m$ matrix. The computational bottleneck in the above procedure is in solving an $md \times md$ system of equations, which scales as $O(m^3 d^3)$ if no further structure is available. The overall computational complexity is $O(m^3 d^3 + mnd^2)$. ■

B.3 Proof of Thm. 3.1

To analyze the convergence rate of gradient descent in this case, we start by bounding the smallest and largest eigenvalue of the quadratic system shown in (**??**).

Lemma B.1 (Weyl's inequality)**.** Let A, B and C be n-by-n Hermitian matrices, and $C = A + B$. Let $a_1 \geq \cdots \geq a_n$, $b_1 \geq \cdots \geq b_n$ and $c_1 \geq \cdots \geq c_n$ be the eigenvalues of A, B and C respectively. Then the following inequalities hold for $r + s - 1 \leq i \leq j + k - n$, $\forall i = 1, \ldots, n$:

$$a_j + b_k \leq c_i \leq a_r + b_s$$

Let $\lambda_k(A)$ be the k-th largest eigenvalue of matrix A.

Lemma B.2. If Σ_1 and Σ_2 are feasible in (4), then

$$\lambda_1(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \leq \lambda_1(X^T X) + \eta + \rho u^2$$
$$\lambda_{md}(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \geq \lambda_d(X^T X) + \eta + \rho l^2$$

Proof. By Weyl's inequality, setting $r = s = i = 1$, we have $c_1 \leq a_1 + b_1$. Set $j = k = i = n$, we have $c_n \geq a_n + b_n$. We can bound the largest and smallest eigenvalues of $I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1$ as follows:

$$\begin{aligned}
&\lambda_1(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \\
&\leq \lambda_1(I_m \otimes (X^T X)) + \lambda_1(\eta I_{md}) + \lambda_1(\rho \Sigma_2 \otimes \Sigma_1) && \text{(By Weyl's inequality)} \\
&= \lambda_1(I_m)\lambda_1(X^T X) + \eta + \rho \lambda_1(\Sigma_1)\lambda_1(\Sigma_2) && \text{(By Fact B.4)} \\
&\leq \lambda_1(X^T X) + \eta + \rho u^2 && \text{(By the feasibility assumption)}
\end{aligned}$$

and

$$\begin{aligned}
&\lambda_{md}(I_m \otimes (X^T X) + \eta I_{md} + \rho \Sigma_2 \otimes \Sigma_1) \\
&\geq \lambda_{md}(I_m \otimes (X^T X)) + \lambda_{md}(\eta I_{md}) + \lambda_{md}(\rho \Sigma_2 \otimes \Sigma_1) && \text{(By Weyl's inequality)} \\
&= \lambda_m(I_m)\lambda_d(X^T X) + \eta + \rho \lambda_m(\Sigma_1)\lambda_d(\Sigma_2) && \text{(By Fact B.4)} \\
&\geq \lambda_d(X^T X) + \eta + \rho l^2 && \text{(By the feasibility assumption)}
\end{aligned}$$

■

We will first prove the following two lemmas using the fact that the spectral norm of the Hessian matrix $\nabla^2 h(W)$ is bounded.

Lemma B.3. Let $f(W) : \mathbb{R}^{d \times m} \mapsto \mathbb{R}$ be a twice differentiable function with $\lambda_1(\nabla^2 f(W)) \leq L$. $L > 0$ is a constant. The minimum value of $f(W)$ can be achieved. Let $W^* = \arg\min_W f(W)$, then

$$f(W^*) \leq f(W) - \frac{1}{2L} ||\nabla f(W)||_F^2$$

Proof. Since $f(W)$ is twice differentiable with $\lambda_1(\nabla^2 f(W)) \leq L$, by the Lagrangian mean value theorem, $\forall W, \widetilde{W}$, we can find a value $0 < t(W, \widetilde{W}) < 1$, such that

$$\begin{aligned}
f(\widetilde{W}) &= f(W) + \text{tr}(\nabla f(W)^T (\widetilde{W} - W)) + \frac{1}{2} \text{vec}(\widetilde{W} - W)^T \nabla^2 f(tW + (1-t)\widetilde{W}) \text{vec}(\widetilde{W} - W) \\
&\leq f(W) + \text{tr}(\nabla f(W)^T (\widetilde{W} - W)) + \frac{L}{2} ||\widetilde{W} - W||_F^2
\end{aligned}$$

Since W^* achieves the minimum value of $f(W)$, we can use the above result to obtain:

$$\begin{aligned} f(W^*) &= \inf_{\widetilde{W}} f(\widetilde{W}) \\ &\leq \inf_{\widetilde{W}} f(W) + \text{tr}(\nabla f(W)^T(\widetilde{W} - W)) + \frac{L}{2}||\widetilde{W} - W||_F^2 \\ &= f(W) - \frac{1}{2L}||\nabla f(W)||_F^2 \end{aligned}$$

where the last equation comes from the fact that the minimum of a quadratic function with respect to $\widetilde{W}$ can be achieved at $\widetilde{W} = W - \frac{1}{L}\nabla f(W)$. ■

Lemma B.4. Let $f(W) : \mathbb{R}^{d\times m} \mapsto \mathbb{R}$ be a convex, twice differentiable function with $\lambda_1(\nabla^2 f(W)) \leq L$. $L > 0$ is a constant, then $\forall W_1, W_2$:

$$\text{tr}\left((\nabla f(W_1) - \nabla f(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

Proof. For all W_1, W_2, we can construct the following two functions:

$$f_{W_1}(Z) = f(Z) - \text{tr}\left(\nabla f(W_1)^T Z\right), \qquad f_{W_2}(Z) = f(Z) - \text{tr}\left(\nabla f(W_2)^T Z\right)$$

Since $f(W)$ is a convex, twice differentiable function with respect to W, it follows that both $f_{W_1}(Z)$ and $f_{W_2}(Z)$ are convex, twice differentiable functions with respect to Z. The first-order optimality condition of convex functions gives the following conditions to hold for Z which achieves the optimality:

$$\nabla f_{W_1}(Z) = \nabla f(Z) - \nabla f(W_1) = 0, \qquad \nabla f_{W_2}(Z) = \nabla f(Z) - f(W_2) = 0$$

Plug in W_1 and W_2 into the above optimality conditions respectively. From the first-order optimality condition we know that W_1 and W_2 achieves the optimal solutions of $f_{W_1}(Z)$ and $f_{W_2}(Z)$, respectively.

Now applying Lemma B.3 to $f_{W_1}(Z)$ and $f_{W_2}(Z)$, we have:

$$\left((W_2) - \text{tr}\left(\nabla f(W_1)^T W_2\right)\right) - \left((W_1) - \text{tr}\left(\nabla f(W_1)^T W_1\right)\right) \geq \frac{1}{2L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

$$\left((W_1) - \text{tr}\left(\nabla f(W_2)^T W_1\right)\right) - \left((W_2) - \text{tr}\left(\nabla f(W_2)^T W_2\right)\right) \geq \frac{1}{2L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

Adding the above two equations leads to

$$\text{tr}\left((\nabla f(W_1) - \nabla f(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{L}||\nabla f(W_1) - \nabla f(W_2)||_F^2$$

■

We can now proceed to show Thm. 3.1.

Theorem 3.1. Let $\lambda_l = \lambda_d(X^T X) + \eta + \rho l^2$, $\lambda_u = \lambda_1(X^T X) + \eta + \rho u^2$ and $\kappa = \lambda_u/\lambda_l$, the condition number. Choose $0 < t \leq 2/(\lambda_u + \lambda_l)$. For all $\varepsilon > 0$, gradient descent with step size t converges to the optimal solution within $O(\kappa \log(1/\varepsilon))$ steps.

Proof. Define function $g(W)$ as follows:

$$g(W) = h(W) - \frac{\lambda_l}{2}||W||_F^2$$

Since we have already bounded that $\lambda_{md}(\nabla^2 h(W)) \geq \lambda_l$, it follows that $g(W)$ is a convex function and furthermore $\lambda_1(\nabla^2 g(W)) \leq \lambda_u - \lambda_l$. Applying Lemma B.4 to g, $\forall W_1, W_2 \in \mathbb{R}^{d\times m}$, we have:

$$\text{tr}\left((\nabla g(W_1) - \nabla g(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{\lambda_u - \lambda_l}||\nabla g(W_1) - \nabla g(W_2)||_F^2$$

Plug in $\nabla g(W) = \nabla h(W) - \lambda_l W$ into the above inequality and after some algebraic manipulations, we have:

$$\text{tr}\left((\nabla h(W_1) - \nabla h(W_2))^T(W_1 - W_2)\right) \geq \frac{1}{\lambda_u + \lambda_l}||\nabla h(W_1) - \nabla h(W_2)||_F^2 + \frac{\lambda_u \lambda_l}{\lambda_u + \lambda_l}||W_1 - W_2||_F^2 \tag{14}$$

Let $W^* = \arg\min_W h(W)$. Within each iteration of Alg. 3, we have the update formula as $W^+ = W - t\nabla h(W)$, we can bound $||W^+ - W^*||_F^2$ as follows

$$\begin{aligned}
||W^+ - W^*||_F^2 &= ||W - W^* - t\nabla h(W)||_F^2 \\
&= ||W - W^*||_F^2 + t^2||\nabla h(W)||_F^2 - 2t\text{tr}\left((W - W^*)^T\nabla h(W)\right) \\
&\leq (1 - 2t\frac{\lambda_u\lambda_l}{\lambda_u + \lambda_l})||W - W^*||_F^2 + t(t - \frac{2}{\lambda_u + \lambda_l})||\nabla h(W)||_F^2 \quad \text{(By inequality 14)} \\
&\leq (1 - 2t\frac{\lambda_u\lambda_l}{\lambda_u + \lambda_l})||W - W^*||_F^2 \quad \text{(For } 0 < t \leq 2/(\lambda_u + \lambda_l)\text{)}
\end{aligned}$$

Apply the above inequality recursively for T times, we have

$$||W^{(T)} - W^*||_F^2 \leq \gamma^T||W^{(0)} - W^*||_F^2$$

where $\gamma = 1 - 2t\frac{\lambda_u\lambda_l}{\lambda_u + \lambda_l}$. For $t = 2/(\lambda_u + \lambda_l)$, we have

$$\gamma = 1 - 4\lambda_u\lambda_u/(\lambda_l + \lambda_u)^2 = \left(\frac{\lambda_u - \lambda_l}{\lambda_u + \lambda_l}\right)^2$$

Now pick $\forall \varepsilon > 0$, setting the upper bound $\gamma^T||W^{(0)} - W^*||_F^2 \leq \varepsilon$ and solve for T, we have

$$T \geq \log_{1/\gamma}(C/\varepsilon) = O(\log_{1/\gamma}(1/\varepsilon)) = O(\kappa\log(1/\varepsilon))$$

where $C = ||W^{(0)} - W^*||_F^2$ is a constant, and $\kappa = \lambda_u/\lambda_l$ is the condition number. ■

The pseudocode of the gradient descent is shown in Alg. 3.

Algorithm 3 Gradient descent with fixed step-size.

Input: Initial W, X, Y and approximation accuracy ε.

1: $\lambda_u \leftarrow \lambda_1(X^TX) + \eta + \rho u^2$.
2: $\lambda_l \leftarrow \lambda_d(X^TX) + \eta + \rho l^2$.
3: Step size $t \leftarrow 2/(\lambda_l + \lambda_u)$.
4: **while** $||\nabla h(W)||_F > \varepsilon$ **do**
5: $\quad W \leftarrow W - t\left(X^T(Y - XW) + \eta W + \rho\Sigma_1 W\Sigma_2\right)$.
6: **end while**

B.4 Proof of Lemma A.1

Lemma A.1. There exists an optimal solution P to the optimization problem (10) that is a $d \times d$ permutation matrix.

Proof. Note that the optimization problem (10) in terms of P is a linear program with the Birkhoff polytope being the feasible region. It follows from Thm. A.1 and Thm. A.2 that at least one optimal solution P is an $d \times d$ permutation matrix. ■

B.5 Proof of Thm. A.3

Theorem A.3. The minimum value of (10) is equal to the minimum weight of a perfect matching on $G = (V_\lambda, V_\nu, E, w)$. Furthermore, the optimal solution P of (10) can be constructed from the minimum-weight perfect matching on G.

Proof. By Lemma A.1, the optimal value is achieved when P is a permutation matrix. Given a permutation matrix P, we can understand P as a bijective mapping from the index of rows to the index of columns. Specifically, construct a permutation $\pi_P : [d] \to [d]$ from P as follows. For each row index $i \in [d]$, $\pi_P(i) = j$ iff $P_{ij} = 1$. It follows that π_P is a permutation of $[d]$ since P is assumed to be a permutation matrix. The objective function in (10) can be written in terms of π_P as

$$\lambda^T P \nu = \sum_{i=1}^{d} \lambda_i \nu_{\pi_P(i)}$$

which is exactly the weight of the perfect matching on $G(V_\lambda, V_\nu, E, w)$ given by π_P:

$$w(\pi_P) = w(\{(i, \pi_P(i)) : 1 \le i \le d\}) = \sum_{i=1}^{d} \lambda_i \nu_{\pi_P(i)}$$

Similarly, in the other direction, given any perfect matching $\pi : [d] \to [d]$ on the bipartite graph $G(V_\lambda, V_\nu, E, w)$, we can construct a corresponding permutation matrix P_π: $P_{\pi,ij} = 1$ iff $\pi(i) = j$, otherwise 0. Since π is a perfect matching, the constructed P_π is guaranteed to be a permutation matrix.

Hence the problem of finding the optimal value of (10) is equivalent to finding the minimum weight perfect matching on the constructed bipartite graph $G(V_\lambda, V_\nu, E, w)$. Note that the above constructive process also shows how to recover the optimal permutation matrix P_{π^*} from the minimum weight perfect matching π^*. ■

B.6 Proof of Thm. A.4

Note that $\lambda = (\lambda_1, \ldots, \lambda_d)$ and $\nu = (\nu_1, \ldots, \nu_d)$ are assumed to satisfy $\lambda_1 \ge \cdots \ge \lambda_d$ and $\nu_1 \le \cdots \le \nu_d$. To make the discussion more clear, we first make the following definition of an *inverse pair*.

Definition B.1 (Inverse pair)**.** Given a perfect match π of $G(V_\lambda, V_\nu, E, w)$, $(\lambda_i, \lambda_j, \nu_k, \nu_l)$ is called an *inverse pair* if $i \le j$, $k \le l$ and $(v_{\lambda_i}, v_{\nu_l}) \in \pi$, $(v_{\lambda_j}, v_{\nu_k}) \in \pi$.

Lemma B.5. Given a perfect match π of $G(V_\lambda, V_\nu, E, w)$ and assuming π contains an inverse pair $(\lambda_i, \lambda_j, \nu_k, \nu_l)$. Construct $\pi' = \pi \backslash \{(v_{\lambda_i}, v_{\nu_l}), (v_{\lambda_j}, v_{\nu_k})\} \cup \{((v_{\lambda_i}, v_{\nu_k}), (v_{\lambda_j}, v_{\nu_l})\}$. Then $w(\pi') \le w(\pi)$.

Proof. Let us compare the weights of π and π'. Note that since $i \le j, k \le l$, we have $\lambda_i \ge \lambda_j$ and $\nu_k \le \nu_l$.

$$\begin{aligned} w(\pi') - w(\pi) &= (\lambda_i \nu_k + \lambda_j \nu_l) - (\lambda_i \nu_l + \lambda_j \nu_k) \\ &= (\lambda_i - \lambda_j)(\nu_k - \nu_l) \\ &\le 0 \end{aligned}$$

Intuitively, this lemma says that we can always decrease the weight of a perfect matching by re-

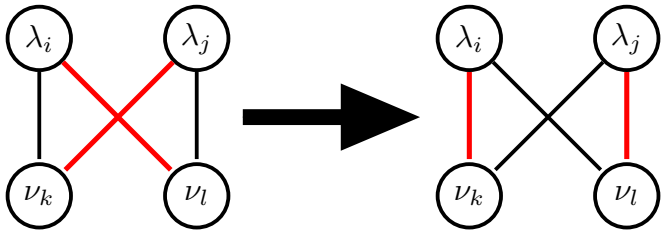


Figure 3: Re-matching an inverse pair $(\lambda_i, \lambda_j, \nu_k, \nu_l) = \{(v_{\lambda_i}, v_{\nu_l}), (v_{\lambda_j}, v_{\nu_k})\}$ on the left side to a match with smaller weight $\{(v_{\lambda_i}, v_{\nu_k}), (v_{\lambda_j}, v_{\nu_l})\}$. Red color is used to highlight edges in the perfect matching.

matching an inverse pair. Fig. 3 illustrates this process. It is worth emphasizing here that the above re-matching process only involves four nodes, i.e., $v_{\lambda_i}, v_{\nu_l}, v_{\lambda_j}$ and v_{ν_k}. In other words, the other parts of the matching stay unaffected. ■

Using Lemma B.5, we are now ready to prove Thm. A.4:

Theorem A.4. Let $\lambda = (\lambda_1, \ldots, \lambda_d)$ and $\nu = (\nu_1, \ldots, \nu_d)$ with $\lambda_1 \geq \cdots \geq \lambda_d$ and $\nu_1 \leq \cdots \leq \nu_d$. The minimum-weight perfect matching on G is $\pi^* = \{(v_{\lambda_i}, v_{\nu_i}) : 1 \leq i \leq d\}$ with the minimum weight $w(\pi^*) = \sum_{i=1}^{d} \lambda_i \nu_i$.

Proof. We will prove by induction.

- **Base case**. The base case is $d = 2$. In this case there are only two valid perfect matchings, i.e., $\{(v_{\lambda_1}, v_{\nu_1}), (v_{\lambda_2}, v_{\nu_2})\}$ or $\{(v_{\lambda_1}, v_{\nu_2}), (v_{\lambda_2}, v_{\nu_1})\}$. Note that the second perfect matching $\{(v_{\lambda_1}, v_{\nu_2}), (v_{\lambda_2}, v_{\nu_1})\}$ is an inverse pair. Hence by Lemma B.5, $w(\{(v_{\lambda_1}, v_{\nu_1}), (v_{\lambda_2}, v_{\nu_2})\}) = \lambda_1\nu_1 + \lambda_2\nu_2 \leq \lambda_1\nu_2 + \lambda_2\nu_1 = w(\{(v_{\lambda_1}, v_{\nu_2}), (v_{\lambda_2}, v_{\nu_1})\})$.
- **Induction step**. Assume Thm. A.4 holds for $d = n$. Consider the case when $d = n + 1$. Start from any perfect matching π. Check the matches of node $v_{\lambda_{n+1}}$ and $v_{\nu_{n+1}}$. Here we have two subcases to discuss:
  - If $v_{\lambda_{n+1}}$ is matched to $v_{\nu_{n+1}}$ in π. Then we can remove nodes $v_{\lambda_{n+1}}$ and $v_{\nu_{n+1}}$ from current graph, and this reduces to the case when $n = d$. By induction assumption, the minimum weight perfect matching on the new graph is given by $\sum_{i=1}^{n} \lambda_i \nu_i$, so the minimum weight on the original graph is $\sum_{i=1}^{n} \lambda_i \nu_i + \lambda_{n+1}\nu_{n+1} = \sum_{i=1}^{n+1} \lambda_i \nu_i$.
  - If $v_{\lambda_{n+1}}$ is not matched to $v_{\nu_{n+1}}$ in π. Let v_{ν_j} be the match of $v_{\lambda_{n+1}}$ and v_{λ_i} be the match of $v_{\nu_{n+1}}$, where $i \neq n + 1$ and $j \neq n + 1$. In this case we have $i < n + 1$ and $j < n + 1$, so $(\lambda_i, \lambda_{n+1}, \nu_j, \nu_{n+1})$ forms an inverse pair by definition. By Lemma B.5, we can first re-match $v_{\lambda_{n+1}}$ to $v_{\nu_{n+1}}$ and v_{λ_i} to v_{ν_j} to construct a new match π' with $w(\pi') \leq w(\pi)$. In the new matching π' we have the property that $v_{\lambda_{n+1}}$ is matched to $v_{\nu_{n+1}}$, and this becomes the above case that we have already analyzed, so we still have the minimum weight perfect matching to be $\sum_{i=1}^{n+1} \lambda_i \nu_i$.

Intuitively, as shown in Fig. 3, an inverse pair corresponds to a cross in the matching graph. The above inductive proof basically works from right to left to recursively remove inverse pairs (crosses) from the matching graph. Each re-matching step in the proof will decrease the number of inverse pairs at least by one. The whole process stops until there is no inverse pair in the current perfect matching. Since the total number of possible inverse pairs, the above process can stop in finite steps. We illustrate the process of removing inverse pairs in Fig. 4. ■

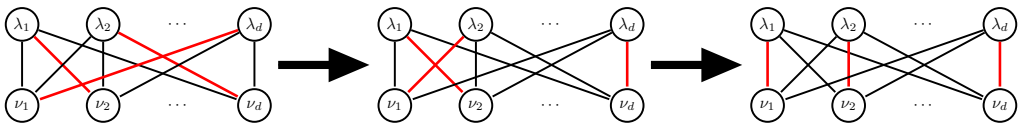


Figure 4: The inductive proof works by recursively removing inverse pairs from the right side of the graph to the left side of the graph. The process stops until there is no inverse pair in the matching. Red color is used to highlight edges in the perfect matching.

Using adversarial autoencoders to infer actions from the peripheral nervous system

Jos van der Westhuizen[1,2], **Tris Edwards**[1], **Raphael Schmetterling**[1,2], **Robert Tinn**[1,2]
jos@cbas.global, tris@cbas.global, raph_s@cbas.global, rob_t@cbas.global

Oliver Armitage [1], **Joan Lasenby**[2], **Emil Hewage**[1]
oliver@cbas.global, jl221@cam.ac.uk, emil@cbas.global

Cambridge Bio-Augmentation Systems[1]
Cambridge University[2]

Abstract

Advances in neural interface technology are giving rise to a new challenge in computational neuroscience. Chronic implants are able to record the activity of neural populations over several months or even years, creating datasets that are difficult to analyze using current neuroinformatic techniques due to their size and crude labels. Here we propose a practical solution, an adversarial autoencoder comprised of LSTMs to generate a label-like latent representation. For this novel data paradigm, we introduce a dataset of local field potential signals over a 6-week recording from a porcine specimen with natural animal mobilization. Our initial implementation yields a classification accuracy of 83.3%, demonstrating the suitability for neuro-prosthesis applications.

1 Introduction

The peripheral nervous system (PNS) holds a plethora of information on physiological processes conveyed from the central nervous system to other systems in the body. Recent improvements in neural interfaces enable long-term recordings of the PNS, generating datasets too large and containing features too nuanced for manual human analysis. In this paper, we propose an adversarial autoencoder to infer actions from such signals in the presence of limited crude labels. The data are multi-channel signals recorded from the tibial nerve of a healthy porcine specimen.

This problem fits under the umbrella of learning representations, more specifically semi-supervised sequence-to-sequence models. Other generalizations of this problem have been solved in seminal work by Makhzani et al. (2015); Srivastava et al. (2015); Hinton and Salakhutdinov (2006). This scenario is pertinent in the medical field where datasets tend to be large and require expert labeling, warranting the use of unsupervised learning as a practical approach to feature learning (Längkvist et al., 2014). Being able to infer the actions of an agent from peripheral neural signals using semi-supervised learning could mitigate the issue of scant or difficult-to-generate labels and as a result, provide useful control signals for neurological human-machine interfaces with potential benefits for amputees.

This is also the introduction of a potentially very rich new data paradigm for the neuro-prosthesis community. Advances in neural interface technology have allowed the collection of peripheral data in a chronic setting, over long periods of time and in natural mobilization settings. However, compared to the shorter and restricted laboratory studies, the new datasets are expensive to label.

The proposed solution comprises a sequence-to-sequence model to encode a given snippet of the neurological signals into a fixed-size representation. The encoder and decoder are long short-term

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

memory recurrent neural networks (Hochreiter and Schmidhuber, 1997). Inspired by the work in Makhzani et al. (2015), we augment the sequence-to-sequence model with adversarial discriminator networks that regularize the generated latent representations. The aim is to capture the latent factors that best explain the data and label the data accordingly. Autoencoders such as sequence-to-sequence models are well known for being able to extract meaningful latent representations (Murphy, 2012). To the best of our knowledge, this work-in-progress paper is the first analysis of peripheral neurological signals with a sequence-to-sequence model.

2 Model architecture

As in Bowman et al. (2015) we make use of a single layer long short-term memory (LSTM) recurrent neural network as an encoder and decoder in an autoencoder fashion. As illustrated in Figure 1 the encoder, denoted as $q(z, y|x)$, generates two fixed size latent representations z and y of an arbitrary length sequence x. The decoder network then utilizes both the z and y representations to reconstruct the original input signal. We assume that the data analysed by the model are generated by a latent class variable y that comes from a categorical distribution as well as a latent style variable z that comes from a Gaussian distribution:

$$p(y) = Cat(y) \qquad\qquad p(z) = \mathcal{N}(z|0, I)$$

For each step in the decoder, the y section of the LSTM memory is replaced by the original y, where the rest of the vector is left to change over time. This places more emphasis on generating an informative y representation for the decoder to utilize at each step. To simultaneously make learning stable and the model robust, we alternate the input to the decoder at each training iteration to either be the true input x, or the output from the previous step in the LSTM. Moreover, we found that reversing the output when decoding (Srivastava et al., 2015) improves training by allowing the model to start off with low-range correlations.

In order to ensure that the y representation is label-like, a discriminator network is employed to introduce an additional loss term. This follows a generative adversarial network approach (Goodfellow et al., 2014; Makhzani et al., 2015) with the generator being an encoder recurrent neural network. The discriminator learns to distinguish between samples from a categorical distribution (random one-hot vectors) and the y representation generated by the encoder. This encourages the y representation towards a categorical distribution from which we can infer actions.

A second discriminator network is employed to encourage the z representation to be Gaussian distributed. This discriminator has to distinguish between generated representations z and samples from a Gaussian distribution $\mathcal{N}(z|0, I)$. Hence by concatenating a selected y representation, $\tilde{y}$, and a $\tilde{z}$ sampled from a Gaussian distribution, one is able to generate neural-like signals from the decoder. Additionally, this allows generation of mixed categories in y.

Batch normalization (Ioffe and Szegedy, 2015) is applied to the input and the two ReLu (Nair and Hinton, 2010) activated hidden layers with 50 and 20 units respectively. This is followed by minibatch discrimination (Salimans et al., 2016) before being linearly mapped into a scalar value. The structure of both discriminators is the same. Batch normalization, minibatch discrimination, and the Wasserstein generative adversarial network (Arjovsky et al., 2017) were found essential to prevent mode collapse during training.

The model is trained in 3 separate stages. First, the encoder and decoder are trained with the mean squared error loss of the input reconstruction. Second, the discriminator function f learns the difference between labels y generated from the generator function g and categorical samples y'. For N data points with the input denoted as x, the y discriminator loss is

$$L_{D_y} = \frac{1}{N}\sum_{n=1}^{N}(-f(y'_n) + f(g(x_n))) \tag{1}$$

where each y'_n is sampled at random from a categorical distribution. Effectively, the discriminator is trained to produce negative values when the input is generated and positive values when the input is sampled from a categorical distribution. Third, the generator (encoder) is trained to generate a y representation that is one-hot-like by 'fooling' the discriminator. The following loss function

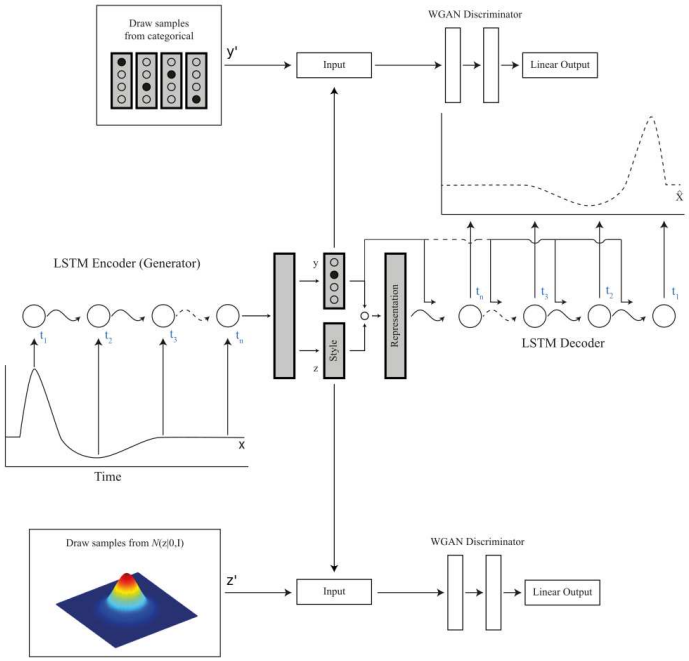


Figure 1: The sequence-to-sequence model regularized by 2 discriminative neural networks. The top discriminator encourages the y representation to be one-hot-like (categorical). The bottom discriminator encourages the z representation to be Gaussian distributed. In the decoder, the original y representation replaces the corresponding elements of the LSTM memory at each time step.

encourages the y encoder to generate a y such that the now fixed function f yields positive values,

$$L_{G_y} = -\frac{1}{N}\sum_{n=1}^{N} f(g(x_n)) \tag{2}$$

The discriminator and generator training updates for the z representation are the same as those detailed above for y, with the exception of replacing the categorical y' with samples z' from a Gaussian distribution. Both z and y generators are updated every third training iteration.

3 Experimental setup

In order to confirm that the model works as intended, we evaluated it on 2 well-understood datasets. The first is a synthetic dataset with 4 classes (sine-, cosine-, saw-tooth-, and square-waves). Here 150,000 samples were generated with unit amplitudes and random periods between 5 and 40 time steps. All the waveforms had a length of 50 time steps. 30,000 samples were held out for testing. The second dataset was a low-resolution version of MNIST (LeCun, 1998) rescaled to range from 0 to 1. Here the MNIST images were resized from a size of 28×28 pixels to 7×7. The images were processed in scanline order (Cooijmans et al., 2016), the shorter sequences making learning easier.

The experimental neural data was collected continuously during natural mobilization of a porcine specimen over a 6-week period. Fifteen channels of local field potential (LFP) data were gathered from the tibial nerve at a sampling rate of 30 kHz using an IT2 PNS implant (Cambridge Bio-Augmentation Systems, Cambridge, UK). The data used in this study is from a 2-hour window 3 days post surgical implantation. Field potentials were identified by means of a 26 µV threshold.

Two variations of the neural data were constructed for investigation. The first dataset consisted of the raw signals on all 15 channels for 50 time steps after a local field potential (LFP) was detected on any channel. A total of 250,911 LFPs were detected over all channels in the recorded period and the data were rescaled to range from 0 to 1. The second dataset consisted of the spike counts on each channel

using a 100ms time window. The counts were similarly rescaled and were then sliced into segments of 50 consecutive counts, resulting in a total of 6,840 data points. The spike counts reduced some of the noise present in the raw data and spanned longer periods. For both datasets above, a single data point had a sequence length of 50 and 15 variables.

Two four-minute video segments of the porcine specimen were recorded during the LFP recordings used for training. Through a retrospective analysis of the video, we were able to identify five distinct motion classes during the time of the video. The actions were: walking forwards, standing, shuffling, reversing, and turning. With the video synchronized to the recorded time series, we were able to label segments of the raw signal according to the identified actions with a granularity of 0.1s. Of the total number of data points in the raw dataset and the count dataset, 3003 and 74 were labeled respectively. The labeled data allowed a quantitative evaluation of the y representation by using the classification accuracy as a proxy. The labeled data were removed from the datasets and not used during training.

For each of the 4 datasets, a validation set was constructed by randomly splitting the training data with an 80:20 (training:validation) ratio. The best model was selected based on the lowest reconstruction error achieved on the validation set over the course of training. Adam (Kingma and Ba, 2014) with a learning rate of 0.001 and no learning rate decay was used for optimization. We denote the number of elements in representations z and y to be S_z and S_y respectively. In order to prevent overfitting on the smaller neural-count dataset, we set $S_y = 20$ and $S_z = 44$. For the synthetic, MNIST, and neural-raw datasets, we chose $S_y = 30$ and $S_z = 98$. Larger y-representations were chosen based on the work in Makhzani et al. (2015) showing that this results in more accurate classifications.

4 Related work

Our work is closely related to the thorough study of Pandarinath et al. (2017), who made use of a variational sequence-to-sequence model to analyze, with unprecedented accuracies, the neurological signals from electrodes implanted in the motor cortex. Our focus is on peripheral neurological signals instead of signals measured in the cortex: one can argue that the former exhibits less interference from other neurological signals, and is certainly a more convenient site for human-machine interfacing for amputees. We also explicitly cluster the signals using the adversarial autoencoder structure.

The model designed as part of this work is similar to that proposed for unsupervised learning in Makhzani et al. (2015). Our model employs recurrent neural networks instead of convolutional neural networks. Kingma et al. (2014) proposed a theoretically grounded technique for doing unsupervised training. The proposed technique was surpassed by the Gumbel-softmax method (Jang et al., 2016), however the adversarial approach Makhzani et al. (2015) remained superior. Our sequence-to-sequence model is based on Bowman et al. (2015). The difference lies in our model using the adversarial autoencoder approach with WGANs for semi-supervised labeling.

For a thorough review of unsupervised feature learning with machine learning, we refer the reader to the fascinating work by Längkvist et al. (2014). The review also gives an overview of all the unsupervised learning applied to signals such as electrocardiograms and electroencephalographs. Serruya et al. (2002) have used motor cortex neurons in primates to control a mouse on a screen. Utilizing the signals in peripheral nerves could replace current electromyography (EMG) systems with improved resolution. EMG have yielded recognition above 95% when combined with neural networks and hidden Markov models (Ahsan et al., 2009) and recognition rates above 90% in a study by Ahsan et al. (2009). To our knowledge, no research has been done to date on a chronic recording of peripheral neurological signals with adversarial sequence-to-sequence networks.

5 Results

Owing to the generated y representation not being fully one-hot encoded, we plot the generated representations for the test sets of the MNIST dataset and the neural-raw dataset in Figure 2 using t-distributed Stochastic Neighbour Embedding (t-SNE) (van der Maaten and Hinton, 2008). From the figure, it is clear that the different classes are much better separated for the MNIST dataset. As expected for the neural-raw dataset, there is more inter-class overlap in the t-SNE plot. Here each data point spans a time of only 0.00167s and thus there could be several data points that are not related to the action we labeled it as. It should also be noted that the clusters in the t-SNE plots are not necessarily the eventual clusters of labels derived from the maximum values in each y representation.

To ensure that the data points are not simply clustered with respect to the time of recording, we overlay the t-SNE plot with a heatmap of the recording time. From this heatmap, it is evident that clusters contain data points from various points in time.

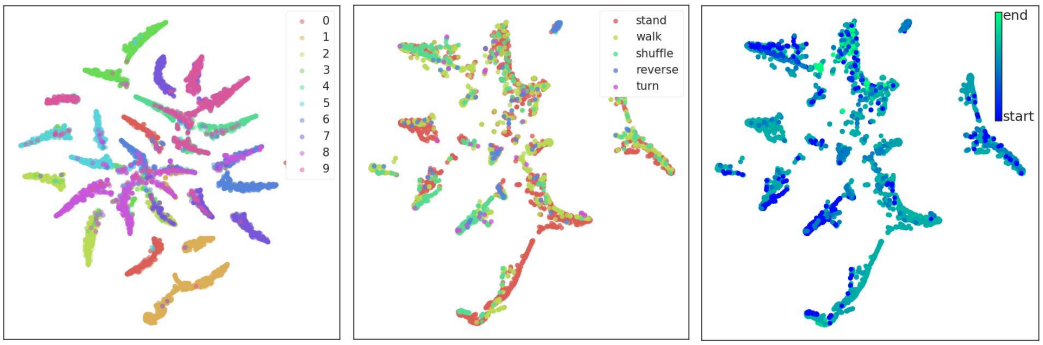


Figure 2: t-SNE plots of the y representation. *Left*: the low-resolution MNIST dataset. *Middle*: The neural-raw dataset colored by class. *Right*: The neural-raw dataset colored by time of recording.

We evaluated classification accuracy as follows: for all dimensions of y find the set $X_i \subset X$ of data points that have maximum probability $p(y|x),\ x \in X$ in dimension i; weight the true class $c \in C_i^{N \times k}$ (where k is the number of true classes) of $x \in X_i$ by the probability $p(y = i|x)$; assign to y_i the maximum class of the average weighted true classes C_i, $\arg\max_k \frac{1}{N} \sum_{n=0}^{N} p(y = i|x_n)c_n$ by means of a hashmap. The accuracy is then computed based on the labels assigned to each data point.

The classification accuracies obtained with the aforementioned datasets are shown in Table 1. The accuracies reported are the averages over five independent runs. We also tabulate the squared error achieved on the test sets to show the efficacy of the data reconstruction achieved by the model. High accuracies were achieved for both the synthetic and MNIST datasets, which confirms that the model works as intended. For the MNIST data, we did not expect similar accuracies to Makhzani et al. (2015) because we use a low-resolution version making the digits harder to distinguish. A higher classification accuracy was achieved on the neural-count dataset compared to the neural-raw dataset. The neural-count dataset reduces the noise present in the raw signal and allows analysis over a longer section of time. Moreover, a single local field potential as measured in the neural-raw dataset (0.00167s) does not necessarily contain information about an action, but a set of them could. The results obtained exceeded expectations given the coarse labeling procedure.

Table 1: Model performance on test data

Dataset	Accuracy [%]	Reconstruction error
Synthetic	89 ± 1.9	0.001316
Low-resolution MNIST	83.5 ± 2.1	0.000177
Neural-raw	63 ± 1.5	0.012113
Neural-count	83.3 ± 1.4	0.006509

Nearly perfect reconstruction was achieved on the MNIST and synthetic dataset. However, the neural data was harder to model and as a result, the reconstructions are smoothed versions of the original signal as shown in Appendix A. Both reconstructions with and without teacher forcing (Goodfellow et al., 2016) are illustrated.

6 Discussion and conclusion

This study, part of a larger on-going project, has proved the utility of the novel, long-term dataset. By means of an adversarial autoencoder, we were able to successfully characterise the dynamics of peripheral neurological signals. We believe this could be of great benefit to amputees by allowing control of smart prosthetic devices through the nerves in a limb. The chosen approach allows the use of a continuous y representation as input to the decoder, which could provide the benefit of allowing combinations of actions as a signal to a prosthetic device. Although the data already span a long

period of time, future work will include validating that the model is able to generalize over much longer periods. One of the main goals is to stitch together datasets collected from different patients as in Pandarinath et al. (2017) to make a large part of the model agnostic to the specific patient. In further work we also aim to compare the model with other architectures.

References

Ahsan, M. R., Ibrahimy, M. I., Khalifa, O. O., and others (2009). EMG signal classification for human computer interaction: A review. *European Journal of Scientific Research*, 33(3):480–501.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating Sentences from a Continuous Space. *arXiv:1511.06349 [cs]*.

Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., and Courville, A. (2016). Recurrent Batch Normalization. *arXiv:1603.09025 [cs]*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*.

Jang, E., Gu, S., and Poole, B. (2016). Categorical Reparameterization with Gumbel-Softmax. *arXiv:1611.01144 [cs, stat]*.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. (2014). Semi-supervised Learning with Deep Generative Models. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24.

LeCun, Y. A. (1998). The MNIST Database of Handwritten Digits. *http://yann.lecun.com/exdb/mnist/*.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial Autoencoders. *arXiv:1511.05644 [cs]*.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., Trautmann, E. M., Kaufman, M. T., Ryu, S. I., Hochberg, L. R., Henderson, J. M., Shenoy, K. V., Abbott, L. F., and Sussillo, D. (2017). Inferring single-trial neural population dynamics using sequential auto-encoders. *bioRxiv*, page 152884.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]*.

Serruya, M. D., Hatsopoulos, N. G., Paninski, L., Fellows, M. R., and Donoghue, J. P. (2002). Brain-machine interface: Instant neural control of a movement signal. *Nature*, 416(6877):141–142.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2.

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

A Appendix: Example reconstructions

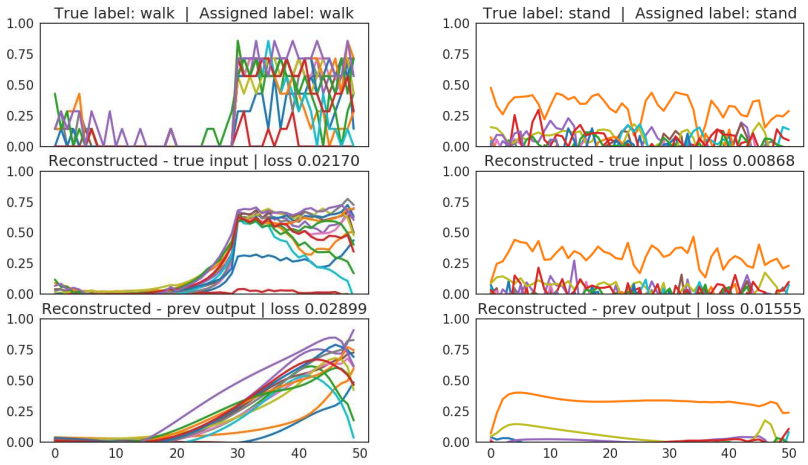


Figure 3: Example reconstructions for the neural-count dataset (left) and the neural-raw data (right). The colors represent the different channels of the signal. The top plot is the original input x, the middle is the reconstructed input where the correct input was fed to the decoder at each step. The bottom graphs are the reconstructions where the output from the previous step is used as input to the next.

Learning Loss Functions for Semi-supervised Learning via Discriminative Adversarial Networks

Cicero Nogueira dos Santos*
IBM Research AI
Yorktown Heights, NY

Kahini Wadhawan
IBM Research AI
Yorktown Heights, NY

Bowen Zhou
JD.com
Beijing, China

Abstract

We propose discriminative adversarial networks (DAN) for semi-supervised learning and loss function learning. Our DAN approach builds upon generative adversarial networks (GANs) and conditional GANs but includes the key differentiator of using two discriminators instead of a generator and a discriminator. DAN can be seen as a framework to learn loss functions for predictors that also implements semi-supervised learning in a straightforward manner. We propose instantiations of DAN for two different prediction tasks: classification and ranking. Our experimental results on three datasets of different tasks demonstrate that DAN is a promising framework for both semi-supervised learning and learning loss functions for predictors. For all tasks, the semi-supervised capability of DAN can significantly boost the predictor performance for small labeled sets with minor architecture changes across tasks. Moreover, the loss functions automatically learned by DANs are very competitive and usually outperform the standard pairwise and negative log-likelihood loss functions for semi-supervised learning.

1 Introduction

One of the challenges in developing semi-supervised learning (SSL) algorithms is to define a loss (cost) function that handles both labeled and unlabeled data. Many SSL methods work by changing the original loss function to include an additional term that deals with the unlabeled data [32, 20, 23]. Recent advances in generative models have allowed the development of successful approaches that perform SSL while doing data generation, which allows the use of unlabeled data in more flexible ways. The two main families of successful generative approaches are based on variational autoencoders (VAE) [13] and generative adversarial networks [7]. Most GAN-based SSL approaches change the loss function of the discriminator to combine a supervised loss (e.g. negative log likelihood with respect to the ground truth labels) with the unsupervised loss normally used in the discriminator [26]. While VAE-based SSL approaches have achieved good results for tasks in both computer vision [14, 16] and natural language processing (NLP) domains [28, 30], GAN-based SSL have primarily targeted tasks from the computer vision domain [24, 22, 5, 15]. The main reason is that applying GANs to discrete data generation problems, e.g. natural language generation, is difficult because the generator network in GAN is designed to be able to adjust the output continuously, which does not (naturally) work on discrete data generation.

In this paper, we propose discriminative adversarial networks (DAN) for SSL and loss function learning. DAN builds upon GAN and conditional GAN but includes the key differentiator of using two discriminators instead of a generator and a discriminator. The first discriminator (the *predictor* P) produces the prediction y given a data point x, and the second discriminator (the *judge* J) takes in a pair (x, y) and judges if it is a *predicted label* pair or *human labeled* pair. While GAN can be seen as a method that implicitly learns loss functions for generative models, DAN can be seen as a method

*Corresponding author. Email: cicerons@us.ibm.com

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

that learns loss functions for predictors. The main benefits of DAN are: (1) The predictor P does not use information from labels, therefore unlabeled data can be used in a transparent way; (2) We do not need to manually define a loss function that handles both labeled and unlabeled data, the judge J implicitly learns the loss function used to optimize P; (3) Different from VAE and GAN-based SSL approaches, in DAN we do not have to perform data generation. This allows the application of SSL using adversarial networks for NLP sidestepping troubled discrete data generation; (4) Prediction problems with complex/structured outputs can benefit from DAN's implicit loss function learning capability. This is important because for many structured prediction tasks such as ranking and coreference resolution, researchers normally use surrogate loss functions since the best loss function for the task is too expensive to compute or, in some cases, because a good loss function is not known.

We have applied DAN for two NLP tasks: answer selection and text classification. We introduced new scoring functions for the judge network that makes the training more stable. Our experimental results demonstrate that: (1) DAN can boost the performance when using a small number of labeled samples; (2) the loss functions learned by DAN outperform standard-pairwise and negative log-likelihood loss functions for the semi-supervised setup, and is also very competitive in the supervised setting.

2 Methods

2.1 Generative Adversarial Nets and Conditional GANs

Generative adversarial networks are an effective approach for training generative models [7]. The GAN framework normally comprises two "adversarial" networks: a generative net G that "learns" the data distribution, and a discriminative net D that estimates the probability that a sample came from the real data distribution rather than generated by G. In order to learn a generator distribution p_g over data x, the generator builds a mapping function from a prior noise distribution $p_z(z)$ to the data space as $G(z;\theta_g)$. The discriminator receives as input a data point x and outputs a single scalar, $D(x;\theta_d)$, which represents the probability that x came from training data rather than p_g . G and D are trained simultaneously by adjusting parameters for G to minimize $log(1-D(G(z))$ and adjusting parameters for D to minimize $logD(x)$, as if they are following a two-player min-max game with the following value function $V(G,D)$:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\left[logD(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[log(1-D(G(z)))\right] \quad (1)$$

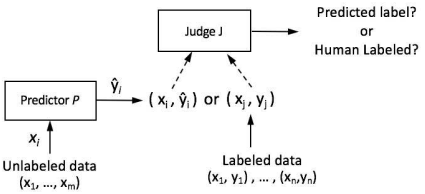
Generative adversarial nets can perform conditional generation if both the generator and discriminator are conditioned on some extra information y [19], generally a class label. Normally y is a class label and the conditioning is performed by feeding y into both the discriminator and generator as an additional input. In the generator, the prior input noise $p_z(z)$ and y are combined in a joint hidden representation. Usually this consists of simply concatenating a vector representation of y to the input vector z. The discriminator receives both x and y as inputs and has to discriminate between real x, y and generated $G(z,y)$. The objective function of the two-player minimax game can be formulated as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x,y\sim p_{data}(x,y)}\left[logD(x,y)\right] + \mathbb{E}_{z\sim p_z(z),y\sim p_y(y)}\left[log(1-D(G(z,y),y))\right] \quad (2)$$

2.2 Discriminative Adversarial Networks

We call DAN the adversarial network framework that uses discriminators only. In our DAN formulation (Fig. 1) we use two discriminators: the *Predictor* P and the *Judge* J. P receives as input a data point x and outputs a prediction $P(x)$. The prediction can be a simple probability distribution over class labels or any other sort of structured predictions such as trees or document rankings. The Judge network J receives as input a data point x and a label y [2] and produces a single scalar, $J(x,y)$, which represents the

Figure 1: DAN framework.



[2] We are using the term *label* in a loose way to also mean any type of structured prediction.

probability that x, y came from the labeled training data
rather than predicted by P. While in conditional GANs
the idea is to generate x conditioned on y, in DAN we want to predict y conditioned on x. The min-max game value function V(J,P) becomes:

$$\min_P \max_J V(J,P) = \mathbb{E}_{x,y\sim p_{data}(x,y)}\ [logJ(x,y)] + \mathbb{E}_{x\sim p_{data}(x)}\ [log(1-J(x,P(x)))] \quad (3)$$

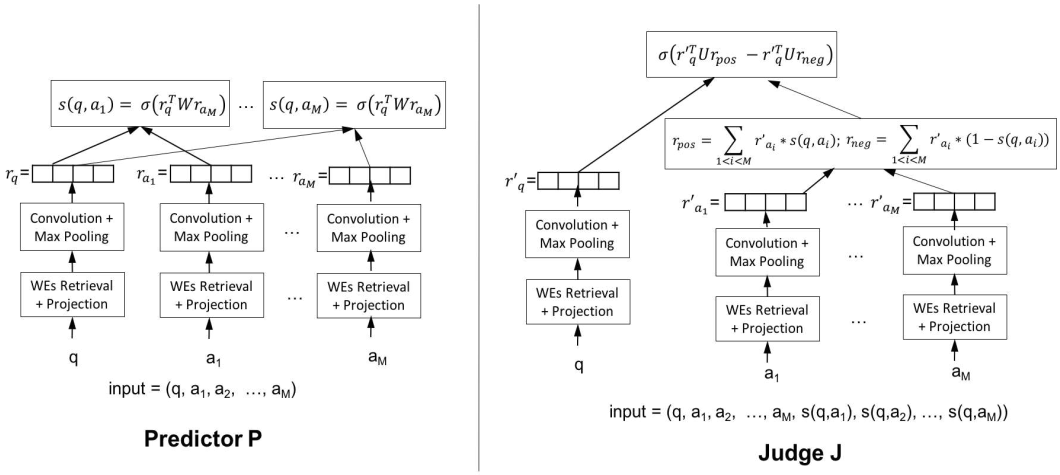


Figure 2: DAN Architecture for Answer Selection

2.2.1 DAN for Answer Selection / Ranking

In the answer selection task, given a question q and a candidate answer pool $P = (a_1, a_2, \cdots, a_M)$ for q, the goal is to search for and select the candidate answer(s) $a \in P$ that correctly answers q. This task can be viewed as a ranking problem where the goal is to rank the candidate answers from the best to the worst. People normally use the following pairwise ranking loss function (hinge loss) when optimizing neural network based rankers: $L = \max\{0, l - s_\theta(q,a^+) + s_\theta(q,a^-)\}$, where a^+ is a positive answer, a^- is a negative answer and l is a margin. However, pairwise ranking loss is known to be suboptimal [2]. Our goal on choosing this ranking task is two fold: (1) we believe that the semi-supervised nature of DANs can help to reduce the need of labeled data for answer selection; (2) we believe that DANs can learn a good listwise loss function by taking into consideration the scoring of the whole set of candidate answers.

As depicted in the left hand side of Fig. 2, the Predictor P takes as input the list $(q, a_1, a_2, \ldots, a_M)$ containing a question q and M candidate answers. Pairs of (q, a_i) are processed in parallel by first generating fixed-length continuous vector representations r_q and r_{a_i} and then performing the operation $\sigma(r_q^T W r_{a_i})$, where W is a matrix of learnable parameters and σ is the sigmoid function. Since we are using a sigmoid, note that the score produced by P is a number between 0 and 1. The parameters of WE projection layer, convolution layer and W are shared among question and all candidate answers.

The right hand side of Fig. 2 details the Judge J, which uses a similar architecture as the predictor, except for the scoring function. There is no parameter sharing between P and J. Note that J also receives as input the score for each candidate answer, which means that J performs a listwise evaluation. For labeled instances, the score for a correct answer is 1 and for an incorrect answer is 0.

After creating the representation r'_q, r'_{a_1}, ..., r'_{a_m}, the Judge J uses the scores s_{a_1}, ..., s_{a_m} to compute representations r_{pos} and r_{neg} as follows:

$$r_{pos} = \sum_{1<i<=M} r'_{a_i} * s(q,a_i) \qquad r_{neg} = \sum_{1<i<=M} r'_{a_i} * (1 - s(q,a_i)) \quad (4)$$

We can think of r_{pos} and r_{neg} as a way to summarize, according to the scores, the similarities and dissimilarities, respectively, between the question and the list of candidate answers. The final scoring is given by $\sigma(r'^T_q U r_{pos} - r'^T_q U r_{neg})$. The rationale behind this scoring function is that, if the given

list of scores is good, the representation of the question, r'_q, should be more similar to r_{pos} than to r_{neg}. As far as we know, this scoring function is novel, and we further extended it for the classification task as presented in the next section.

2.2.2 DAN for Text Classification

Our DAN architecture for text classification is similar to the answer selection architecture in many aspects. We use one layer CNNs on P and J, and J employs a scoring function that is similar to the one used in the answer selection architecture. More architecture details are in the Appendix A.1.

3 Related work

Our approach is mainly related to recent works on semi-supervised GANs and conditional GANs. Springenberg [26] proposed a categorical generative adversarial network (CatGAN) which can be used for unsupervised and semi-supervised learning, where the discriminator outputs a distribution over classes and is trained to minimize the predicted entropy for real data and maximize the predicted entropy for fake data. Salimans et al. [24] proposed a semi-supervised GAN model in which the discriminator outputs a softmax over classes rather than a probability of real vs. fake. An additional "generated" class is used as the target for generated samples. Kumar et al. [15] use the same GAN-like SSL setup proposed in [24], but use tangents from the generator's mapping to further improve on SSL. Different from these works, in DAN we do not perform a generation step, therefore it is easier to apply for discrete data.

Regarding loss function learning using GAN-like approaches, Isola et al. [9] proposed conditional GANs for image-to-image translation problems, and showed that their models not only learns good mappings but also learns a loss function to train the mapping. Finn et al. [6] presented a connection between GAN-based loss function learning for generative models and cost function learning in reinforcement learning (aka inverse reinforcement learning). They demonstrated that certain IRL methods are mathematically equivalent to GANs. While previous work focus on learning loss functions for generative models, in DAN we focus on learning loss functions for discriminative models.

Another recent line of work consists of using adversarial examples [27, 8] based on unlabeled data to regularize the training [20]. For the NLP domain, the work by Miyato et al. [21] extended the adversarial and virtual adversarial training approaches by adding small perturbations to embeddings. They report good performance for semi-supervised text classification tasks. In DAN, instead of adding an extra regularization term to the supervised loss, we implicitly learn the loss function.

4 Experiments and results

We use two different datasets to perform our answer selection experiments: SelQA [10] and WikiQA [29]. For both datasets, we use the subtask that assumes that there is at least one correct answer for a question. For the text classification task, we use the Stanford Sentiment Tree-bank (SSTb) dataset and focus on binary classification only.

For both tasks, answer selection and sentiment classification, we perform semi-supervised experiments where we randomly sample a limited number of labeled instances and use the rest as unlabeled data. We use the term CNN-DAN to refer to the DAN architecture for that respective task. However, in the CNN-DAN setup, the instances presented to P are the exact same instances that appear in the labeled set. Therefore, CNN-DAN is basically trying to learn a better loss function using the available labeled data, no semi-supervised learning is performed. We use the term CNN-DAN$_{unlab.}$ to refer to the DAN setup where we feed P with additional unlabeled data.

In Tables 1, 2 and 3 we present the experimental results for SelQA, WikiQA and SSTB2, respectively. We present results for: CNN-DAN; CNN-DAN$_{unlab.}$, that uses unlabeled data in P; CNN$_{hinge_loss}$, which is the same CNN-based architecture of the predictor P in our DAN for answer selection (Fig. 2), but that is trained using the hinge loss function instead of the DAN framework; CNN$_{nll}$, which is the same CNN-based architecture of the predictor P in our DAN for text classification (Fig. 6), but that is trained using the negative log likelihood (NLL) loss function instead of the DAN framework. We present detailed results for datasets containing a different number of labeled instances: 10, 50 and

Table 1: Experimental Results for the SelQA dataset.

Model	10 labeled instances			50 labeled instances			Full dataset		
	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG
CNN_{hinge_loss}	.4610	.4661	.5889	.6455	.6545	.7331	**.8758**	**.8812**	**.9079**
CNN-DAN	.5749	.5811	.6780	.6248	.6332	.7170	.8655	.8730	.9012
$\text{CNN-DAN}_{unlab.}$	**.6891**	**.6978**	**.7667**	**.6928**	**.7017**	**.7695**	-	-	-

Table 2: Experimental Results for the WikiQA dataset.

Model	10 labeled instances			50 labeled instances			Full dataset		
	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG
CNN_{hinge_loss}	.5447	.5577	.6575	.5919	.6071	.6942	.6511	.6669	.7402
CNN-DAN	.5437	.5582	.6566	.6047	.6201	.7042	**.6663**	**.6822**	**.7516**
$\text{CNN-DAN}_{unlab.}$	**.5927**	**.6068**	**.6945**	**.6127**	**.6274**	**.7104**	-	-	-

Table 3: Experimental Results for the SSTB2 dataset.

Model	10 instances	50 instances	Full dataset
	Average Accuracy	Average Accuracy	Average Accuracy
CNN_{nll}	$60.42 \pm (3.68)$	$71.36 \pm (3.04)$	**87.0**
CNN-DAN	$63.25 \pm (3.78)$	$\mathbf{74.23 \pm (2.02)}$	**87.1**
$\text{CNN-DAN}_{unlab.}$	$\mathbf{65.62 \pm (3.46)}$	$72.37 \pm (1.79)$	-

Table 4: Comparison with previously reported results for SelQA, WikiQA and SSTB2.

SelQA			WikiQA			SSTB2	
Model	MAP	MRR	Model	MAP	MRR	Model	Acc.
[10]	.8643	**.8759**	[29]	.6520	.6652	[3]	85.5
CNN-DAN	**.8655**	.8730	[31]	.6600	.6770	CNN-DAN	**87.1**
-	-	-	CNN-DAN	.6663	.6822	[11]	**87.2**
-	-	-	[4]	**.6886**	**.6957**		

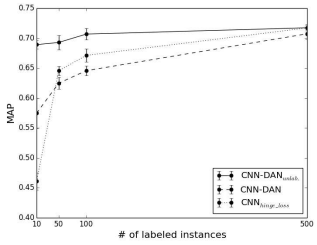


Figure 3: SelQA

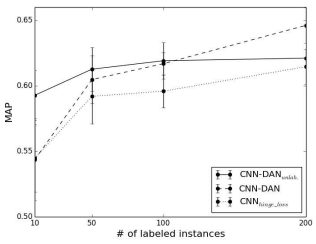


Figure 4: WikiQA

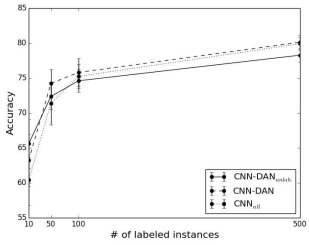


Figure 5: SSTB2

full dataset. In Figs. 3 and 4 we also present the MAP, and the Acc. in Fig 5, for datasets with 100 and 500 labeled instances.

We can see in Figs. 3, 4 and 5 that the semi-supervised DAN, $\text{CNN-DAN}_{unlab.}$, gives a significant boost in performance when a small amount of labeled instances is available. These results are evidence that DAN is a promising approach for semi-supervised learning. Comparing CNN-DAN, that does not use unlabeled data, with CNN_{hinge_loss} and CNN_{NLL} is a reasonable way to check whether the learned loss function is doing better or not than the pairwise hinge loss and the NLL loss. We can see that in most situations the loss function implicitly learned by the judge performs at least as good as the NLL loss and the pairwise hinge loss.

References

[1] A. Bordes, J. Weston, and N. Usunier. Open question answering with weakly supervised embedding models. In *Proc. of ECML*, pages 165–180, 2014.

[2] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: From pairwise approach to listwise approach. In *Proc. of ICML*, ICML '07, pages 129–136, 2007.

[3] C. N. dos Santos and M. Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proc. of COLING*, pages 69–78, 2014.

[4] C. N. dos Santos, M. Tan, B. Xiang, and B. Zhou. Attentive pooling networks. *CoRR, abs/1602.03609*, 2016.

[5] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. In *Proc. of ICLR*, 2017.

[6] C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. In *NIPS Workshop on Adversarial Training*, 2016.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. of NIPS*, page 2672, 2014.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. of CVPR*, 2017.

[10] T. Jurczyk, M. Zhai, and J. D. Choi. Selqa: A new benchmark for selection-based question answering. *arXiv preprint arXiv:1606.08513*, 2016.

[11] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[14] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Proc. of NIPS*, pages 3581–3589, 2014.

[15] A. Kumar, P. Sattigeri, and P. T. Fletcher. Improved semi-supervised learning with gans using manifold invariances. *CoRR*, abs/1705.08850, 2017.

[16] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *Proc. of ICML*, pages 1445–1453, 2016.

[17] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proc. of ACL*, pages 142–150, 2011.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, 2013.

[19] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[20] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

[21] T. Miyato, A. M. Dai, and I. Goodfellow. Virtual adversarial training for semi-supervised text classification. In *Proc. of ICLR*, 2016.

[22] A. Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[23] M. Sajjadi, M. Javanmardi, and T. Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proc. of NIPS*, pages 1163–1171, 2016.

[24] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Proc. of NIPS*, pages 2226–2234, 2016.

[25] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, page 1642, 2013.

[26] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. In *Proc. of ICLR*, 2016.

[27] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[28] W. Xu, H. Sun, C. Deng, and Y. Tan. Variational autoencoder for semi-supervised text classification. In *Proc. of AAAI*, 2017.

[29] Y. Yang, W.-t. Yih, and C. Meek. Wikiqa: A challenge dataset for open-domain question answering. In *Proc. of EMNLP*, pages 2013–2018, 2015.

[30] Z. Yang, Z. Hu, R. Salakhutdinov, and T. Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *Proc. of ICML*, 2017.

[31] W. Yin, H. Schütze, B. Xiang, and B. Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*, 2015.

[32] X. Zhu. Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.

A Appendix

A.1 DAN Architecture for Text Classification

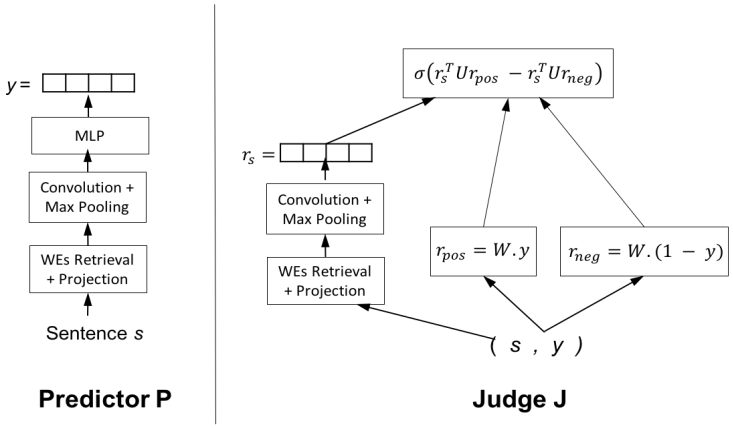


Figure 6: DAN Architecture for Text Classification

As illustrated in left hand side of Fig. 6, the Predictor P is a standard CNN-based text classifier that classifies a given sentence s into one of N classes. It takes in sentence s as input and outputs y a probability distribution over N classes. We first retrieve the word embeddings (WEs) and project them using a fully connected layer. Next, a convolutional layer followed by a MLP is used to perform the prediction.

The Judge J takes in a pair (x,y) consisting of a sentence and its class label, and classifies the pair as being predicted label (fake) or human labeled pair (real). For the human labeled pairs, y is encoded as the one hot representation of the class label. For predicted pairs, $\hat{y}$ is the output of P, which is a probability distributions over the class labels. As in the Predictor, we create a representation r_s of the sentence using a convolution. In the Judge, as can be noticed in right hand side of Fig. 6, we create two representations of the class label y, r_{pos} and r_{neg} using a embedding matrix W. The representation r_{pos}, can be seen as the embedding of the positive/correct label. While the representation r_{neg} can be understood as the average embedding of the negative classes. The final scoring is done by first measuring the similarity between r_s and r_{pos}, and between r_s and r_{neg} using bilinear forms: $r_s^T U r_{pos}$ and $r_s^T U r_{pos}$, where U is a matrix of learnable parameters. This type of bilinear similarity measure has been previously used for embedding comparison in [1]. Next, the difference between the two similarities are passed through the sigmoid function (σ). The rationale behind this scoring function is that, if the given label is correct, the representation of the sentence, r_s, should be more similar to r_{pos} than to r_{neg}. In our experiments, this scoring approach has shown to be empirically easier to train under the min-max game than concatenating r_{pos} and r_s and giving the resulting vector as input to a logistic regression (or MLP). We developed this scoring approach for the ranking task first (next section) and later we realized that it also works well for classification.

A.2 Experimental setup details

For both SelQA and WikiQA datasets, we use the subtask that assumes that there is at least one correct answer for a question. For the WikiQA, the corresponding dataset consists of 873 questions in the training set (20,360 question/candidate pairs), 126 in the dev set (1,130 pairs) and 243 questions in the test set (2,352 pairs). For SelQA, the corresponding dataset contains of 5529 questions in the training set, 785 in the dev set and 1590 questions in the test set. SelQA is more than 6 times larger than WikiQA in number of questions.

The Stanford Sentiment Tree-bank (SSTb) dataset is a movie review dataset proposed by [25] which includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences. In our experiments we focus on binary classification only. This dataset, which is known as SSTB2, contains 6,920 training sentences, 872 dev. sentences and 1,821 test sentences. When performing experiments with the full dataset we follow the protocol used in [25], which consists in traininig with the set of all phrases instead of just the complete sentences. However, in the semi-supervised experiments we train using complete sentences only.

In all experiments, we use word embeddings of size 400, which were pre-trained using the word2vec tool [18]. For the answer selection task we use a dump of Enlgish Wikipedia. For sentiment classification, we pretrain the word embeddings using the IMDB data proposed by [17].

We use the ADAM optimizer [12], and kept the values of most of the hyperparameters fixed for all the experiments. For both the Predictor and the Judge, the word embeddings projection layer has 200 units, the convolutional layer has 400 filters, with context window of sizes 3 and 5 words in the case of answer selection and text classification, respectively. The U matrix has dimensionally $\mathbb{R}^{400\times400}$. When training using the full dataset, we alternately update J and P one time each. We use a learning rate of $\lambda = 0.0005$ for the answer selection task, and $\lambda = 0.0001$ for the text classification task. Validation sets are used to perform early stopping. Normally it is needed less than 100 epochs to achieve the best performance in the validation set.

For the semi-supervised experiments, since the set of unlabeled instances is much large than the one of labeled, we noticed that we need to update P more frequently than J in order to avoid overfitting J. For better results in the semi-supervised setup, we normally update P 10 times after each update of J. However, in this case we also had to use a smaller learning rates for P ($\lambda = 0.00005$) and J ($\lambda = 0.0001$).

For both tasks, answer selection and sentiment classification, we perform semi-supervised experiments where we randomly sample a limited number of labeled instances and use the rest of the dataset as unlabeled data. In all experiments reported in the next two sections, we repeat the random sampling 10 times and average the results. Additionally, in the experiments using the full labeled dataset we repeat the experiments 10 times with different seeds for the random number generator and average the results.

For both tasks, we use the term CNN-DAN to refer to the DAN architecture for that respective task (Figs. 6 and 2). However, in the CNN-DAN setup, the instances presented to P are the exact same instances that appear in the labeled set. Therefore, CNN-DAN is basically trying to learn a better loss function using the available labeled data, no semi-supervised learning is performed. We use the term CNN-DAN$_{unlab.}$ to refer to the DAN setup where we feed P with additional unlabeled data.

Co-trained Ensemble Models for Weakly Supervised Cyberbullying Detection

Elaheh Raisi
Department of Computer Science
Virginia Tech
elaheh@vt.edu

Bert Huang
Department of Computer Science
Virginia Tech
bhuang@vt.edu

Abstract

Social media has become an inevitable part of individuals' social and business lives. Its benefits come with various negative consequences. One major concern is the prevalence of detrimental online behavior on social media, such as online harassment and cyberbullying. In this study, we aim to address the computational challenges associated with harassment detection in social media by developing a machine-learning framework with three distinguishing characteristics. (1) It uses minimal supervision in the form of expert-provided key phrases that are indicative of bullying or non-bullying. (2) It detects harassment with an ensemble of two learners that co-train one another; One learner examines the language content in the message, and the other learner considers the social structure. (3) It incorporates distributed word and graph-node representations by training nonlinear deep models. The model is trained by optimizing an objective function that balances a co-training loss with a weak-supervision loss. We evaluate the effectiveness of our approach using post-hoc, crowdsourced annotation of Twitter data, finding that our deep ensembles outperform previous non-deep methods for weakly supervised harassment detection. We also evaluate on a new benchmark to measure the sensitivity of the detectors to language describing particular social groups.

1 Introduction

The advent of social media has revolutionized human communication. Social media owes its increasing popularity to its uncountable positive influences on individuals' social and business lives. It makes people closer to each other, provides access to enormous real-time information, and eases marketing and business. Despite these benefits, social media has amplified some detrimental aspects of society. Online harassment and cyberbullying are among the major adverse consequences of social media's popularity. According to the American Academy of Child and Adolescent Psychiatry [1], victims of bullying can be suffer interference to social and emotional development and even be drawn to extreme behavior such as attempted suicide. Any widespread bullying enabled by technology represents a serious social health threat.

In this paper, we consider a machine-learning approach for harassment-based cyberbullying detection. We approach to the cyberbullying detection problem from different angle than many machine-learning algorithms proposed thus far. Most machine learning methods for this problem consider supervised text-based cyberbullying detection, classifying social media posts as "bullying" or "non-bullying." In these approaches, crowdsource workers annotate the data, and then a supervised classifier is applied to classify the posts. There are, however several challenges related to these approaches. Fully annotating data requires human intervention, which is costly and time consuming. And without considering social context, differentiating bullying from less harmful behavior is difficult due to complexities underlying cyberbullying and related behavior. Our approach aims to encode such complexities into an efficiently learnable model.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

We use machine learning with weak supervision, which significantly alleviates the need for human experts to perform tedious data annotation. Our weak supervision is in the form of expert-provided key phrases that are highly indicative of bullying. We refer to our proposed framework as the *co-trained ensemble* method, which trains two detectors to extrapolate from the weak supervision to form a rich, multi-faceted model. One detector identifies bullying by examining the language content in messages; another detector considers the social structure to detect bullying. Each detector is using different body of information, and the individual detectors co-train one another to come to an agreement about the bullying concept. They seek consensus on whether examples in unlabeled data are cases of cyberbullying or not.

We represent the language and users as vectors of real numbers with embedding models. For example, word2vec [15, 16] is a popular word-embedding model that represents words with low-dimensional vectors. And node2vec [6] is a framework for learning continuous feature representations for nodes in networks. We use word and user vectors as the input to language-based and user-based classifiers, respectively. We examine two strategies when incorporating vector representations of words and users. First, using existing doc2vec [13]—an extension of word embedding—models as inputs to the learners. Second, creating new embedding models specifically geared for our specific task of harassment detection, which we train in an end-to-end manner during optimization of the model, incorporating the unsupervised doc2vec and node2vec loss function into our co-training objective.

To train the model, we construct an optimization problem made up of a regularizer and two sets of loss functions: a co-training loss that penalizes the disagreement between the deep language-based model and the deep user-based model, and a weak-supervision loss that is the classification loss on weakly labeled messages.

We evaluate our approach on Twitter data, which is one of the public-facing social media platforms with the most frequency of cyberbullying. We use a human-curated list of key phrases indicative of bullying as the weak supervision, and assess the precision of detections by variations of the framework. We evaluate the effectiveness of our approach using post-hoc, crowdsourced annotation of Twitter. We quantitatively demonstrate that our weakly supervised deep models improves precision over a non-deep variation of the model. In addition, we measure how biased and discriminative the proposed algorithm is against particular targeted groups including but not limited to race, gender, religion, and sexual orientations. Our experiments show that our proposed framework—which combines of weak supervision, co-training, and deep, nonlinear detectors—performs better than a model lacks any one of these three characteristics.

2 Related Work

Many researchers have proposed computational methods for automated online harassment and cyberbullying detection. Most methods developed so far use supervised classification algorithms to classify messages as "bullying" or "non-bullying" by extracting language features. Some proposed gender-specific language features to classify users into male and female groups to improve the discrimination capacity of a classifier for cyberbullying detection [4]. Others applied a lexical syntactic feature (LSF) [3] approach to detect offensive content in social media and users who send offensive messages. Others focused on detecting of textual cyberbullying in YouTube comments by manually labeling 4,500 YouTube comments and applying binary and multi-class classifiers [5]. Another approach used the number, density and the value of offensive words as features for cyberbullying identification on the Formspring service [21]. There have been many contributions in designing special features: using features learned by topic models as well as curse words weighted by TF-IDF [17], using sentiment features [24], applying vulgar language expansion using string similarity [19], extracting features based on association rule techniques [14], and using static, social structure features [12]. Additionally, some studies have involved firsthand accounts of young persons, yielding insights on new features for bullying detection and strategies for mitigation [2]. Hosseinmardi et al. [7, 8, 9, 10] conducted several studies analyzing cyberbullying on different online platforms, with findings that highlight cultural differences among the platforms.

Our work directly builds off a recent paper that introduced the *participant-vocabulary consistency* (PVC) method [20], which uses a similar paradigm of viewing the learning tasks as seeking consensus between language-based and user-based perspectives of the problem. PVC uses simple key-phrase presence and a two-parameter user characterization, scoring how much a user tends to bully and

how much they tend to be victimized, as its vocabulary and participant detectors, respectively. Our approaches replaces these with richer classifiers.

Recent reactions to a Google Jigsaw-released tool for quantifying toxicity of online conversations (see e.g., [22]) have highlighted an important aspect of any automated harassment or bullying detection: fairness, especially in the context of false positives. A serious concern of these detectors is how differently they flag language used by or about particular groups of people. We begin to address this issue with a benchmark analysis in our experiments.

3 Co-Trained Ensembles

Our learning framework uses co-trained ensembles of weakly supervised detectors. In this section, we first describe them generally, then we describe the specific instantiations we use in our experiments. For social media data, we consider a set of users U and a set of messages M. Each message $m \in M$ is sent from user $s(m)$ to user $r(m)$. In other words, the functions s and r return the sender and receiver, respectively, of their input message. The input data takes on this form, with some of the messages annotated with weak supervision.

General Framework We define two types of classifiers for harassment detection: message classifiers and user-relationship classifiers (or user classifiers for short). Message classifiers take a single message as input and output a classification score for whether the message is an example of harassment, i.e., $f : M \mapsto \mathbb{R}$. User classifiers take an ordered pair of users as input and output a score indicating whether one user is harassing the other user, i.e., $g : U^2 \mapsto \mathbb{R}$. For message classifiers, our framework accommodates a generalized form of weakly supervised loss function ℓ (which could be straightforwardly extended to also allow full or partial supervision). Let Θ be the model parameters for the combined ensemble of both classifiers. The training objective is

$$\min_{\Theta} \underbrace{\frac{1}{2|M|} \sum_{m \in M} \left(f(m;\Theta) - g\left(s(m), t(m);\Theta\right)\right)^2}_{\text{consistency loss}} + \underbrace{\frac{1}{|M|} \sum_{m \in M} \ell\left(f(m;\Theta)\right)}_{\text{weak supervision loss}}, \tag{1}$$

where the first loss function is a consistency loss, and the second loss function is the weak supervision loss. The consistency loss penalizes the disagreement between the scores output by the message classifier for each message and the user classifier for the sender and receiver of the message.

We experiment with different variations of this framework that arise from instantiating the weak supervision loss and different classification models for the user and message classifiers.

Key-Phrase Weak Supervision Loss Our weak supervision relies on annotated lists of key-phrases that are indicative or counter-indicative of harassment. For example, various swear words and slurs are common indicators of bullying, while positive-sentiment phrases such as "thanks" are counter-indicators. Let there be a set of indicator phrases and a set of counter-indicator phrases for harassment. Our weak supervision loss ℓ is based on the fraction of indicators and counter-indicators in each message, so for a message containing $n(m)$ total key-phrases, let $n^+(m)$ denote the number of indicator phrases in message m and $n^-(m)$ denote the number of counter-indicator phrases in the message. Our weak supervision loss is then

$$\ell(y_m) = -\log\left(\min\left\{1, 1 + (1 - \tfrac{n^-(m)}{n(m)}) - y_m\right\}\right) - \log\left(\min\left\{1, 1 + y_m - \tfrac{n^+(m)}{n(m)}\right\}\right). \tag{2}$$

Models For the message classifiers, we use a randomly hashed bag of n-grams (BoW) model with 1,000 hash functions [23], a linear classifier based on the pre-trained doc2vec vector of messages trained on our Twitter dataset [13], a custom-trained embedding model with each word represented with 100 dimensions (emb), and a recurrent neural network (LSTM) with 2 hidden layers of 100 dimensionality (RNN). The emb and RNN models are trained end-to-end to optimize our overall loss function, and the vector-based models (BoW, doc2vec) are trained to only adjust the linear classifier weights given the fixed vector representations for each message.

For user classifiers, we use a linear classifier on concatenated vector representations of the sender and receiver user nodes. For our first user classifier (vec), we pre-train a node2vec [6] representation of the communication graph, which uses an algorithm designed to find vector representations that organize nodes based on their network roles and communities they belong to. Our other user classifier (emb) directly trains vector embeddings of the nodes to optimize our objective function in an end-to-end manner.

4 Experiments

Mirroring the setup initially used to evaluate PVC [20], we construct our weak supervision signal by collecting a dictionary of 3,461 offensive key-phrases (unigrams and bigrams) [18]. We augment this with a list of positive opinion words in [11]. The offensive phrases are our weak indicators and the positive words are our counter-indicators.

We use the data collected by Raisi & Huang [20]. They collected data from Twitter's public API, extracting tweets containing offensive-language words posted between November 1, 2015, and December 14, 2015. They then extracted conversations and reply chains that included these tweets. They then used snowball sampling to gather tweets in a wide range of topics. After some preprocessing, the Twitter data contains 180,355 users and 296,308 tweets.

4.1 Precision Analysis

We use post-hoc human annotation to measure how well the outputs of the algorithms agree with annotator opinions about bullying. We asked crowdsourcing workers from Amazon Mechanical Turk to evaluate the discovery of cyberbullying interactions of all methods. First, we average the user and message classification score of each message. Then, we extract the 100 messages most indicated to be bullying by each method. Finally, we collected the full set of messages sent between the sender and receiver of these messages. We showed the annotators the anonymized conversations and asked them, "Do you think either user 1 or user 2 is harassing the other?" The annotators indicated either "yes," "no," or "uncertain." We collected five annotations per conversation.

In Fig. 1, we plot the precision@k of the top 100 interactions for all the combinations of message and user detectors. We compare these methods with each other and against PVC [20]. The precision@k is the proportion of the top k interactions returned by each method that the majority of annotators agreed seemed like bullying. For each of the five annotators, we score a positive response as +1, a negative response as -1, and an uncertain response as 0. We sum these annotation scores for each interaction, and we consider the interaction to be harassment if the score is greater than or equal to 3.

Considering the BoW message detector, the best precision is when BoW message detector is combined with the node2vec user detector. Interestingly, BoW by itself does quite well; its precision is slightly lower than BoW with the node2vec user learner. The BoW message learner with the embedding user detector does not perform well. The embedding message detector, after interaction 20, with and without the user detector does about the same, but noticeably much better than PVC. The RNN message detector by itself has high precision, similarly to when combined with the node2vec user detector, which leads to a slightly lower precision. RNN when combined with the embedding user detector has the lowest precision, similar to PVC. The word2vec message detector when combined with the node2vec user learner has the best precision, better than the word2vec message detector alone. The precision of word2vec combined with the embedding user detector is worse than PVC.

To sum up, the BoW and word2vec message detectors when combined with the node2vec user detector had better precision than when combined with the embedding user detector and not having a user learner at all. The RNN message detector has the best precision without any user learner, slightly better than when combined with the node2vec user learner. The precision of the embedding message learner is almost the same for all user learners. For all of the methods, there is a significant improvement over PVC. It is worth mentioning that the embedding user learner has the best performance when combined with embedding message learners, otherwise its precision is poor.

4.2 Identity Statements

We create a corpus of sentences using the combination of some sensitive keywords describing different attributes: sexual orientation, race, gender, and religion. We generate statements of identity (e.g., "I am a black woman.") that are not harassment. An ideal, fair language-based detector should treat these keywords equitably, not estimating higher scores for any keyword over any other.

Score-Based Comparison Using different combination of message and user learners (12 methods in total), we computed the average score of sentences containing each keyword. Since none of these statements are in fact reasonable examples of harassment, the ideal score should be low; any high scores are false positives. The rnn_emb combination has the lowest score of 0.147. Next, rnn_node2vec (0.257) and emb_none (0.381) also have reasonable low scores. Methods that returned

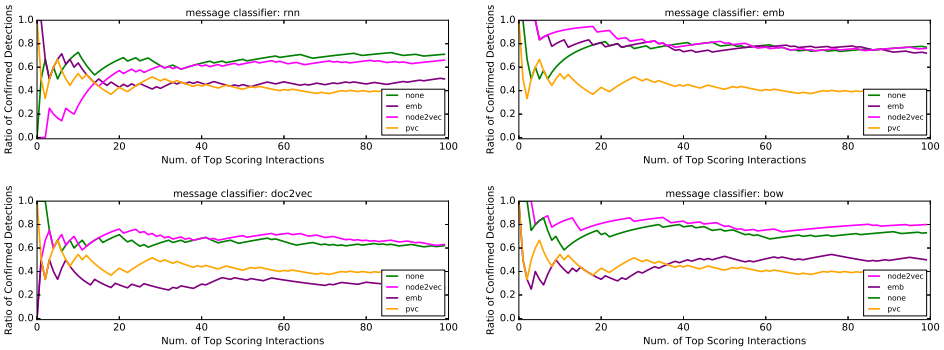


Figure 1: Precision@k for bullying interactions on Twitter using the combination of message and user learners, and PVC.

high average scores include bow_node2vec (0.536), bow_none (0.543), and emb_node2vec (0.588). The RNN message detector when combined with a user detector (node2vec or embedding) are among the least sensitive to these identity statements. The BoW message detector generally (with and without a user learner) and the embedding message detector combined with the node2vec user learner have higher scores. We believe these differences may illustrate the fact that the RNN model attempts to consider the sequence of language and its structure more than the orderless bag-of-words representation, but more study is necessary to make definitive conclusions.

Keyword Score Comparisons We compute the average score of sensitive keywords according to each method to find out which keywords have highest score (more falsely positive), and which keywords have the lowest score. We show the results for two methods: rnn_node2vec and rnn_none. In Fig. 2, three keywords "black," "boy," and "queer" have the highest score by both methods. The words "black" and "queer" were in our negative seed words, so this reveals a need to be more careful about which words are included in the weak supervision. One hypothesis why the word "boy" is among the highest-scored words is that this word co-occurs with many offensive words in the Twitter data. Generally, the score of all keywords returned by rnn_node2vec is lower than rnn_none, which suggests that the learning algorithm that co-trains using social structure helps to reduce the bias and sensitivity of the RNN message detector. The score of "woman" was not higher than the score of "man" for both methods, and the score of "girl" was much lower than the score of "boy," while in a model biased against women, we expect to observe the reverse behavior.

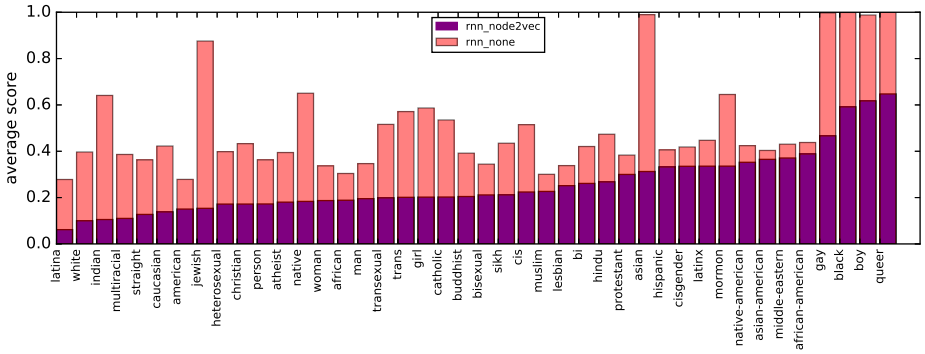


Figure 2: Average score of statements containing each keyword by rnn_node2vec and rnn_none.

5 Conclusion

We present a method for detecting online harassment using weak supervision. Harassment detection requires managing the time-varying nature of language, the difficulty of labeling the data, and complexity of understanding the social structure behind these behaviors. We developed a weakly supervised framework in which two learners train each other to form a consensus whether the social interaction is bullying by incorporating nonlinear embedding models. Our preliminary experiments show that co-training can help improve precision as well as produce more equitable models.

References

[1] American Academy of Child Adolescent Psychiatry. Facts for families guide. the American Academy of Child Adolescent Psychiatry. *http://www.aacap.org/AACAP/*, 2016.

[2] Z. Ashktorab and J. Vitak. Designing cyberbullying mitigation and prevention solutions through participatory design with teenagers. In *Proc. of the CHI Conf. on Human Factors in Computing Systems*, pages 3895–3905, 2016.

[3] Y. Chen, Y. Zhou, S. Zhu, and H. Xu. Detecting offensive language in social media to protect adolescent online safety. *Intl. Conf. on Social Computing*, pages 71–80, 2012.

[4] M. Dadvar, F. de Jong, R. Ordelman, and D. Trieschnigg. Improved cyberbullying detection using gender information. *Dutch-Belgian Information Retrieval Workshop*, pages 23–25, February 2012.

[5] K. Dinakar, R. Reichart, and H. Lieberman. Modeling the detection of textual cyberbullying. *ICWSM Workshop on Social Mobile Web*, 2011.

[6] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[7] H. Hosseinmardi, A. Ghasemianlangroodi, R. Han, Q. Lv, and S. Mishra. Towards understanding cyberbullying behavior in a semi-anonymous social network. *IEEE/ACM International Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 244–252, August 2014.

[8] H. Hosseinmardi, S. Li, Z. Yang, Q. Lv, R. I. Rafiq, R. Han, and S. Mishra. A comparison of common users across Instagram and Ask.fm to better understand cyberbullying. *IEEE Intl. Conf. on Big Data and Cloud Computing*, 2014.

[9] H. Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra. Analyzing labeled cyberbullying incidents on the Instagram social network. In *Intl. Conf. on Social Informatics*, pages 49–66, 2015.

[10] H. Hosseinmardi, S. A. Mattson, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra. Detection of cyberbullying incidents on the Instagram social network. *Association for the Advancement of Artificial Intelligence*, 2015.

[11] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[12] Q. Huang and V. K. Singh. Cyber bullying detection using social and textual analysis. *Proceedings of the International Workshop on Socially-Aware Multimedia*, pages 3–6, 2014.

[13] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

[14] H. Margono, X. Yi, and G. K. Raikundalia. Mining Indonesian cyber bullying patterns in social networks. *Proc. of the Australasian Computer Science Conference*, 147, January 2014.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[17] V. Nahar, X. Li, and C. Pang. An effective approach for cyberbullying detection. *Communications in Information Science and Management Engineering*, 3(5):238–247, May 2013.

[18] noswearing.com. List of swear words & curse words. *http://www.noswearing.com/dictionary*, 2016.

[19] M. Ptaszynski, P. Dybala, T. Matsuba, F. Masui, R. Rzepka, and K. Araki. Machine learning and affect analysis against cyber-bullying. In *Linguistic and Cognitive Approaches to Dialog Agents Symposium*, pages 7–16, 2010.

[20] E. Raisi and B. Huang. Cyberbullying detection with weakly supervised machine learning. In *Proceedings of the IEEE/ACM International Conference on Social Networks Analysis and Mining*, 2017.

[21] K. Reynolds, A. Kontostathis, and L. Edwards. Using machine learning to detect cyberbullying. *Intl. Conf. on Machine Learning and Applications and Workshops (ICMLA)*, 2:241–244, 2011.

[22] C. Sinders. Toxicity and tone are not the same thing: analyzing the new Google API on toxicity, PerspectiveAPI. https://medium.com/@carolinesinders/toxicity-and-tone-are-not-the-same-thing-analyzing-the-new-google-api-on-toxicity-perspectiveapi-14abe4e728b3.

[23] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proc. of the Intl. Conf. on Machine Learning*, pages 1113–1120, 2009.

[24] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards. Detection of harassment on Web 2.0. *Content Analysis in the WEB 2.0*, 2009.

SRL4ORL: Improving Opinion Role Labelling Using Multi-Task Learning With Semantic Role Labeling

Ana Marasović
Research Training Group AIPHES
Department of Computational Linguistics
Heidelberg University
marasovic@cl.uni-heidelberg.de

Anette Frank
Research Training Group AIPHES
Department of Computational Linguistics
Heidelberg University
frank@cl.uni-heidelberg.de

Abstract

For over a decade, machine learning has been used to extract opinion-holder-target structures from text to answer the question *Who expressed what kind of sentiment towards what?*. Recent neural approaches do not outperform the state-of-the-art feature-based model for Opinion Role Labelling (ORL). We suspect this is due to the scarcity of labelled training data and address this issue using different multi-task learning techniques with a related task which has substantially more data, i.e. Semantic Role Labelling (SRL). Despite difficulties of the benchmark MPQA corpus, we show that indeed the ORL model benefits from SRL knowledge.

1 Introduction and related work

Fine-Grained Opinion Analysis aims to: detect explicit opinion expressions (O) (such as *has supported* in the example (1)[1]), measure their intensity (e.g. strong), identify their holders (H), i.e. entities that express an opinion (e.g. *Mexico*), identify their targets (T), i.e. entities or propositions at which the sentiment is directed (e.g. *the OPEC cutbacks*) and classify target-dependent sentiment (e.g. positive).

(1) Traditionally, [Mexico]$_{H_1}$ [has supported]$_{O_1(pos)}$ [the OPEC cutbacks]$_{T_1}$; however, [analysts]$_{H_2}$ [agreed]$_{O_2(pos)}$ that [it]$_{T_2,H_3}$ [will now tend to support]$_{O_3(pos)}$ [the United States]$_{T_3}$, the main world consumer, and the one that would be adversely affected by a possible increase in crude prices.

As the commonly accepted benchmark corpus MPQA [1] uses span-based annotations to represent *opinion entities* (opinions, holders and targets), the task is usually approached with sequence labeling techniques and the BIO encoding scheme [2, 3, 4]. Initially were proposed pipeline models which first predict opinion expressions and then, given an opinion, label its *opinion roles*, i.e. holders and targets [5]. Pipeline models have been substituted with the so-called joint models that simultaneously identify all opinion entities and predict which opinion role is related to which opinion [2, 3, 4]. Recently an LSTM-based joint model was proposed [4] that unlike the prior work [2, 3] does not depend on external resources (such as syntactic parsers, named entity recognizers, etc.). The neural variant does not outperform the feature-based CRF model [3] in Opinion Role Labeling (ORL).

Both the neural and the CRF joint models achieve circa 55% F1 score for predicting which targets relate to which opinions in MPQA, meaning that these models are not ready to answer the question this line of research is usually motivated with: *Who expressed what kind of sentiment towards what?*

Our goal is to investigate the limitations of neural models in solving different subtasks of fine-grained opinion analysis on MPQA and to gain a better understanding of what is solved and what is next.

[1]The example drawn from MPQA [1]. Opinions in MPQA can also be beliefs, emotions, speculations, etc.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

We suspect that one of the fundamental obstacles for neural models trained on MPQA is its small size. One way to cope with scarcity of labeled data is to use multi-task learning with appropriate auxiliary tasks. A good auxiliary task candidate for ORL is Semantic Role Labeling (SRL), the task of predicting predicate-argument structure of a sentence, which answers the question *Who did what to whom, where and when?*. For the first 23 tokens of example (1) the output of the SRL demo[2] is:

	Traditionally	,	Mexico	has	supported	the	OPEC	cutbacks	;	however	,	analysts	agreed	that	it	will	now	tend	to	support	the	United	States
support.01	AM-TMP	-	A0	-	-	A1	A1	A1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
cutback.01	-	-	-	-	-	-	A0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
agree.01	-	-	-	-	-	-	-	-	-	AM-DIS	-	A0	-	A1	A1	A1	A1	A1	A1	A1	A1	A1	A1
tend.02	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A1	AM-MOD	AM-TMP	-	A2	A2	A2	A2	A2
support.01	-	-	-	-	-	-	-	-	-	-	-	-	-	-	A0	-	-	-	-	-	A1	A1	A1

If we compare the semantic roles of the predicates *support*, *agree* and *(tend to) support* we can notice significant overlap with the opinion roles of the corresponding opinions according to MPQA (marked blue). For this reason, the output of a SRL system has been commonly used for feature-based models for fine-grained opinion analysis [5, 2, 3]. Additionally, there is a considerable amount of available annotated data for training SRL models (Table 1), which made neural SRL models successful [6, 7].

Although SRL is a reasonable auxiliary task, an obstacle for properly exploiting SRL training data with MTL could be imprecisions and incompleteness of the MPQA annotations. In example (1), *agreed* is considered an opinion with positive sentiment and its target is marked as the token *it*, meaning that annotators understood that the *analysts* are positive towards *Mexico*, just because *analysts* agree that *Mexico will now tend to support the United States*. Even if they have expressed an opinion, the target would not be *it*, but *it will now tend to support the United States*. Regarding incompleteness, prior work [4] has shown that their model makes reasonable predictions in sentences which do not have annotations at all, e.g. $[\text{mothers}]_{H_1}$ $[\text{care}]_{O_1}$ for $[\text{their young}]_{T_1}$, in: *From the fact that mothers care for their young, we can not deduce that they ought to do so, Hume argued.* Both types of shortcomings in the annotation ground truth will be revealed by correct SRL annotations.

In spite of these challenges, we adopt one of the recent successful architectures for SRL [6], experiment with different multi-task learning frameworks and show that indeed an ORL model can benefit from MTL with SRL.

2 Neural MTL for SRL and ORL

As a general neural architecture for single- and multi-task learning we use the recently proposed SRL model [6] (Z&H) which successfully labels semantic roles without any syntactic guidance. This model consists of a stack of bi-directional LSTMs and a CRF which makes the final prediction. Every sentence is processed as many times as there are predicates in it. The inputs to the first LSTM are not only token embeddings but three additional features: embedding of the predicate, embedding of the context of the predicate and an indicator feature (1 if the current token is in the predicate context, 0 otherwise). Adapting this model for labeling of opinion roles is straightforward, the only difference being that opinion expressions can be multi-words and only two opinion roles are assigned: H and T.

Multi-task learning (MTL) techniques aim to learn several tasks jointly by leveraging knowledge from all tasks. In the context of neural networks, MTL is commonly used such that is predefined which layers have tied parameters and which are task-specific (i.e. hard-parameter sharing). There are various ways of defining which parameters should be shared and how to train them.

Fully-shared (FS) MTL model. In a fully-shared model (Fig. 1), all parameters of the general model except the output layer are shared. Each task has a task-specific output layer which makes the prediction based on the representation produced by the final LSTM. When training on a mini-batch of a certain task, parameters of the output layer of the other tasks are not updated.

Hierarchical MTL (H-MTL) model. For NLP applications, often some given task (high-level task) is supposed to benefit from another task (low-level task) more than other way around, e.g. parsing from POS tagging. This intuition lead to designing hierarchical MTL models [8, 9] in which predictions for low-level tasks are not made on the basis of the representation produced at the final LSTM, but on the representation produced by a lower-layer LSTM (Fig. 2).

Shared-private (SP) MTL model. In the state-private model in addition to the stack of shared LSTMs, each task has a stack of task-specific LSTMs [10] (Fig. 3). Representations at the outermost

[2] `http://barbar.cs.lth.se:8081`

	task	train size	dev size	test size	$\|\mathcal{Y}\|$	OOV rate %	$H_{\mathcal{Y}}$
CoNLL'05	SRL	90750	3248	6071	106	14.53	1.87
MPQA	ORL	3951	1498	450	7	5.97	1.03

Table 1: Datasets w/ nb. of SRL predicates/ORL opinions in train, dev & test set, size of label inventory, percent. of training words not in pre-trained GloVe embeddings, entropy of label distribution.

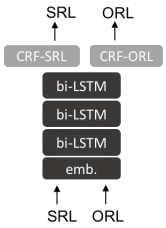


Figure 1: Fully-shared (FS) MTL model.

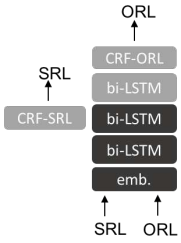


Figure 2: Hierarchical MTL model (H-MTL).

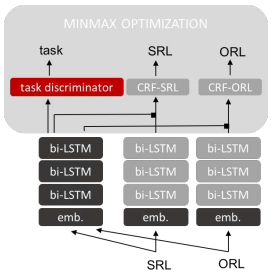


Figure 3: (Adversarial) state-private ((A)SP) MTL model.

shared LSTM and the task-specific LSTM are concatenated and passed to the task-specific output layer. This architecture enables the model to selectively utilize the shared and task-specific information.

Adversarial shared-private (ASP) model. The limitation of the SP model is that it does not prevent the shared layers from capturing task-specific features. To prune shared layers, the SP model is extended with a *task discriminator* [10]. The task discriminator (Fig. 3, marked red) predicts to which task the current batch of data belongs, based on the representation produced by the shared LSTMs. If the shared LSTMs are task-invariant, this discriminator should perform badly, so we update the shared parameters such that the entropy of the predicted task distribution is maximized. At the same time we want the discriminator to challenge the shared LSTMs, so we update the discriminator's parameters to minimize its cross-entropy loss. This minmax optimization is known as *adversarial training* and recently it gained a lot of attention for NLP applications [10, 11, 12, 13, 14, 15, 16, 17, 18].

3 Experimental setup

3.1 Datasets

For SRL we use the newswire CoNLL-2005 shared task dataset [19], annotated with PropBank predicate-argument structures. Sections 2-21 of the WSJ [20] are used for training and section 24 for devset. The test set consists of Section 23 of WSJ and 3 sections of the Brown corpus.

For ORL we use the MPQA corpus [1] which contains news documents manually annotated for opinions and other private states. We follow prior work which set aside 132 documents for development and used the remaining 350 documents for evaluation. MPQA allows annotating implicit opinions as explicit with a special implicit tag label, which in some cases annotators missed to indicate. To capture all implicit opinions, we discard one-character long opinions and opinions whose holder is the writer of the document. More data statistics can be found in Table 1.

3.2 Evaluation metrics

For both tasks we adopt evaluation metrics from prior work. For SRL, precision is defined as the proportion of semantic roles predicted by a system which are correct, recall is the proportion of gold roles which are predicted by a system, F1 score is the harmonic mean of precision and recall.

In case of ORL, we report 10-fold CV[3] with two measures: *binary F1 score* and *proportional F1 score*, for holders and targets separately. Binary F1 score is the same as the SRL F1 score described above, just for opinion roles. *Proportional recall* measures proportion of the overlap between a

[3]We used the same splits as the prior work [4]. We thank the authors for providing the information.

gold holder (target) and an overlapping predicted holder (target), *proportional precision* measures proportion of the overlap between a predicted holder (target) and an overlapping gold holder (target). Again, F1 score is the harmonic mean of the corresponding precision and recall.

3.3 Training details

Input representation. We used 100d GloVe word embeddings [21] pre-trained on Gigaword and Wikipedia and to avoid overfitting, did not fine-tune them. For MTL models vocabulary was built from all the words in the train data of both tasks, and OOV words were replaced with an UNK token. The embedding of the context a predicate or an opinion is the average of the embeddings of the predicate or the opinion phrase, of 2 preceding words and 2 words after.

Weights initialization. The size of all LSTM hidden states was set to 100. The number of the backward and the forward LSTM layers is set to 3, which counts for 6 LSTM layers in Z&H. Z&H achieved circa 2% higher SRL F1 score with 8 LSTM layers, but that deep models would cause overfitting on small-sized ORL data. In H-MTL model SRL is supervised at the 2nd LSTM layer. We initialized the LSTM weights with random orthogonal matrices [22], all other weight matrices with the *He initialization* [23]. LSTM forget biases were initialized with 1s [24], all other biases with 0s.

Optimization. We trained our model in mini-batches of size 32 using Adam [25] with the learning rate of 10^{-3}. For MTL we alternate batches from different tasks. We clip gradients by global norm [26], with a clipping value set to 10. Single-task models were trained for 10K iterations and MTL models for 20K. The entropy of the predicted task distribution is scaled with 0.05. We stop training if the arithmetic mean of proportional F1 scores of holders and targets is not improved in 3 epochs.

Regularization. We used l2-regularization on output layers with λ set to $8.35 \cdot 10^{-3}$. Dropout [27] with a keep probability $k_p \in 0.85$ was applied to the outputs of the LSTMs and to the input embeddings with $k_p \in 0.75$. HPs were not tuned.

4 Results

All models are evaluated every 1000 iterations on the ORL devset and saved if they achieve a higher arithmetic mean of proportional F1 scores of holders and targets on the ORL devset. The saved models are used for evaluation on the ORL test set. The mean of F1 scores over 10 folds, μ, and the standard deviation, σ, of all models are reported in Table 2. Evaluation metrics follow Section 3.2. We define the F1 score interval as $[\mu - \sigma, \mu + \sigma]$. The lower part of Table 2 compares the F1 scores of MTL models with the F1 scores of the single-task model (Z&H) and measures the difference between the point $\mu - \sigma$ of MTL models (P1) and the point $\mu + \sigma$ of the single-task Z&H model (P2). If P1 is larger than P2 we can expect that the difference of the F1 scores is significant.

Single-task vs. MTL. Table 2 shows that all MTL models improve in large margins over the single-task Z&H model with all evaluation measures, for both holders and targets. On devset, the F1 score intervals of FS and H-MTL do not overlap with the F1 score interval of Z&H, meaning that improvements on the dev set are probably significant. However, the picture changes on the test set. This shift in results could be due the small-size of the test set (Table 1), which results with a high-variance estimate. Indeed, if we compare standard deviations on the test set we notice they are always much higher than on devset. Larger improvements are visible in labelling of holders, which is not surprising given that holders are usually short, less ambiguous and clearly resemble the A0 role.

Comparing MTL models. Simpler models (FS, H-MTL) with 677828 parameters achieve better results than their more complex competitors (SP, ASP) with 1987630 parameters. We experimented with lowering the number of shared layers and the number of task-specific layers in the SP model to lower the number of parameters, but this did not lead to improvements. If we compare FS with H-MTL we can notice that in 2 cases H-MTL performs better, in 2 cases FS, otherwise they perform nearly the same (difference is smaller than 0.1). If we compare their F1 score intervals, they always overlap. Therefore we concluded that for ORL there is no benefit in choosing FS over H-MTL and vice versa. Finally, as expected SP benefits from adversarial training for predicting targets in devset.

SRL results. SRL results with MTL (test F1-scores in range $[60.70, 61.10]$) are lower than the performance of the single-task SRL model (test F1-score 77.51). However, this is expected as we stop training MTL models after 20K iterations, when models converge on ORL data, but not yet on

	dev (MPQA)				test (MPQA)			
	holder		target		holder		target	
	bin. F1 $\mu \pm \sigma$	**prop. F1** $\mu \pm \sigma$	**bin. F1** $\mu \pm \sigma$	**prop. F1** $\mu \pm \sigma$	**bin. F1** $\mu \pm \sigma$	**prop. F1** $\mu \pm \sigma$	**bin. F1** $\mu \pm \sigma$	**prop. F1** $\mu \pm \sigma$
Z&H	70.61 ± 1.05	67.75 ± 1.16	69.39 ± 1.01	63.70 ± 1.47	70.63 ± 2.67	68.63 ± 2.53	70.24 ± 3.29	63.34 ± 2.45
FS	**75.39 ± 0.65**	**72.77 ± 0.63**	**73.39 ± 1.35**	67.71 ± 1.69	75.03 ± 3.42	**72.90 ± 3.08**	**75.16 ± 2.19**	68.74 ± 3.00
H-MTL	75.13 ± 0.97	72.73 ± 1.14	72.95 ± 0.72	**67.92 ± 1.02**	**75.10 ± 2.91**	72.89 ± 2.76	75.07 ± 1.55	**69.09 ± 2.21**
SP	74.21 ± 1.60	71.26 ± 1.84	71.35 ± 1.01	64.55 ± 0.80	74.43 ± 3.26	72.22 ± 3.34	73.10 ± 1.11	65.05 ± 1.83
ASP	73.88 ± 1.49	71.07 ± 1.79	71.78 ± 1.03	64.99 ± 0.99	73.57 ± 2.98	71.18 ± 2.88	73.07 ± 1.94	65.21 ± 2.59

	dev (MPQA)				test (MPQA)			
	holder		target		holder		target	
	bin. F1	**prop. F1**	**bin. F1**	**prop. F1**	**bin. F1**	**prop. F1**	**bin. F1**	**prop. F1**
ΔF1(FS , Z&H)	4.78	5.03	3.99	4.01	4.40	4.27	4.91	5.40
ΔF1(H-MTL , Z&H)	4.52	4.98	3.56	4.22	4.46	4.26	4.83	5.76
ΔF1(SP , Z&H)	3.60	3.51	1.96	0.85	3.80	3.58	2.86	1.71
ΔF1(ASP , Z&H)	3.27	3.32	2.39	1.29	2.94	2.54	2.82	1.88
Δ(FS $\mu - \sigma$, Z&H $\mu + \sigma$)	3.08	3.24	1.64	0.85	-1.69	-1.34	-0.56	-0.06
Δ(H-MTL $\mu - \sigma$, Z&H $\mu + \sigma$)	2.50	2.67	1.83	1.73	-1.11	-1.03	-0.01	1.09
Δ(SP $\mu - \sigma$, Z&H $\mu + \sigma$)	0.96	0.51	-0.06	-1.42	-2.13	-2.28	-1.54	-2.58
Δ(ASP $\mu - \sigma$, Z&H $\mu + \sigma$)	0.73	0.37	0.35	-1.16	-2.71	-2.87	-2.40	-3.17

Table 2: ORL results and comparison of MTL models with the single-task ORL model (Z&H).

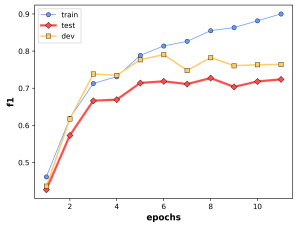


Figure 4: Learning curve of FS-MTL for holders with prop. F1.

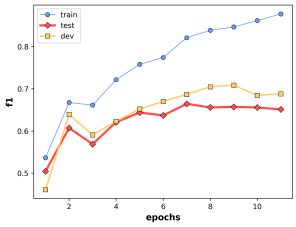


Figure 5: Learning curve of FS-MTL for targets with prop. F1.

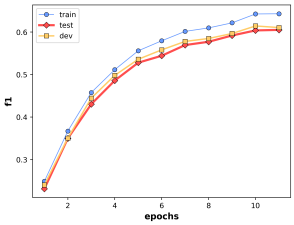


Figure 6: Learning curve of FS-MTL for SRL.

SRL data as it is visible on Figures 4–6 (1 epoch counts for 1K iterations). These figures illustrate the insight shown by the related work [28]: MTL works when the main task (ORL) has a flattening learning curve (Figures 4 & 5), but the auxiliary task (SRL) curve is still steep (Figure 6).

5 Conclusions and future directions

We address the problem of scarcity of annotated training data for labelling of opinion holders and targets (ORL) with multi-task learning (MTL) with Semantic Role Labelling (SRL). We experimented with different MTL frameworks and found that simpler MTL models achieve the best improvements over the single-task model. However, we still do not know what kind of SRL knowledge is transferred and what it is that makes simpler MTL models more successful.

Although simplicity is desirable, we expect more from MTL models; namely, *interpretability, flexibility* and the possibility of *continual learning*. Interpretability could help understanding what kind of SRL knowledge is helpful for ORL. By flexibility we mean possibilities to integrate other tasks (e.g. dependency parsing), another annotation schema (FrameNet) and cross-lingual labelling. Although this might seem trivial, the obstacle is that the different (NLP) tasks perform best with different types of architectures. The SP model can be extended with different architectures for different tasks, languages and annotation schemas. Finally, we would like to learn from all tasks beneficial for ORL (over different languages, datasets and annotation schemes), never forget gained knowledge and let the model decide what to use. Shared layers in the SP model can be seen as off-the-shelf knowledge and be used for unseen new tasks [10].

In future work we will design a model that marries the simplicity of vanilla MTL models with the flexibility, possibility of continual learning and interpretability of the more complex MTL models.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1. We would like to thank anonymous reviewers for useful comments.

References

[1] Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*, 39:165–210, 2005.

[2] Yejin Choi, Eric Breck, and Claire Cardie. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Sydney, Australia, July 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W06/W06-1651`.

[3] Bishan Yang and Claire Cardie. Joint Inference for Fine-grained Opinion Extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August 2013. URL `http://www.aclweb.org/anthology/P13-1161`.

[4] Arzoo Katiyar and Claire Cardie. Investigating LSTMs for Joint Extraction of Opinion Entities and Relations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 919–929, Berlin, Germany, August 2016. URL `http://www.aclweb.org/anthology/P16-1087`.

[5] Soo-Min Kim and Eduard Hovy. Extracting Opinions, Opinion Holders, and Topics Expressed in Online News Media Text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 1–8, Sydney, Australia, July 2006. URL `http://www.aclweb.org/anthology/W/W06/W06-0301`.

[6] Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China, July 2015. URL `http://www.aclweb.org/anthology/P15-1109`.

[7] Bishan Yang and Tom Mitchell. A joint sequential and relational model for frame-semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1258–1267, Copenhagen, Denmark, September 2017. URL `https://www.aclweb.org/anthology/D17-1129`.

[8] Anders Søgaard and Yoav Goldberg. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany, August 2016. Association for Computational Linguistics. URL `http://anthology.aclweb.org/P16-2038`.

[9] Kazuma Hashimoto, caiming xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 446–456, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D17-1046`.

[10] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-1001`.

[11] Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. Adversarial multi-criteria learning for chinese word segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1193–1203, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-1110`.

[12] Young-Bum Kim, Karl Stratos, and Dongchan Kim. Adversarial adaptation of synthetic or stale data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1297–1307, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-1119`.

[13] Lianhui Qin, Zhisong Zhang, Hai Zhao, Zhiting Hu, and Eric Xing. Adversarial connective-exploiting networks for implicit discourse relation classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1006–1017, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/P17-1093`.

[14] Yi Wu, David Bamman, and Stuart Russell. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1779–1784, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1187.

[15] Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. Part-of-speech tagging for twitter with adversarial neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2401–2410, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1255.

[16] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2147–2159, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1229.

[17] Meng Zhang, Yang Liu, Huanbo Luan, and Maosong Sun. Adversarial training for unsupervised bilingual lexicon induction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1959–1970, Vancouver, Canada, July 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/P17-1179.

[18] Shafiq Joty, Preslav Nakov, Lluís Màrquez, and Israa Jaradat. Cross-language learning with adversarial neural networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 226–237, Vancouver, Canada, August 2017. Association for Computational Linguistics. URL http://aclweb.org/anthology/K17-1024.

[19] Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W05/W05-0620.

[20] Eugene Charniak, Don Blaheta, Niyu Ge, Keith Hall, John Hale, and Mark Johnson. Bllip 1987-89 wsj corpus release 1. *Linguistic Data Consortium, Philadelphia*, 36, 2000.

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1162.

[22] Mikael Henaff, Arthur Szlam, and Yann LeCun. Recurrent orthogonal networks and long-memory tasks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2034–2042, 2016.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[24] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2342–2350, 2015.

[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, 2015.

[26] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1310–1318, 2013.

[27] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1): 1929–1958, 2014.

[28] Joachim Bingel and Anders Søgaard. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 164–169, Valencia, Spain, April 2017. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E17-2026.

Object category learning and retrieval with weak supervision

Steven Hickson, Anelia Angelova, Irfan Essa, Rahul Sukthankar
Google Brain / Google Research
(shickson, anelia, irfanessa, sukthankar)@google.com

Abstract

We consider the problem of retrieving objects from image data and learning to classify them into meaningful semantic categories with minimal supervision. To that end, we propose a fully differentiable unsupervised deep clustering approach to learn semantic classes in an end-to-end fashion without individual class labeling using only unlabeled object proposals. The key contributions of our work are 1) a kmeans clustering objective where the clusters are learned as parameters of the network and are represented as memory units, and 2) simultaneously building a feature representation, or embedding, while learning to cluster it. This approach shows promising results on two popular computer vision datasets: on CIFAR10 for clustering objects, and on the more complex and challenging Cityscapes dataset for semantically discovering classes which visually correspond to cars, people, and bicycles. Currently, the only supervision provided is segmentation objectness masks, but this method can be extended to use an unsupervised objectness-based object generation mechanism which will make the approach completely unsupervised.

1 Introduction

Unsupervised discovery of common patterns is a long standing task for artificial intelligence as shown in Barlow (1989); Bengio, Courville, and Vincent (2012). Recent deep learning approaches have offered major breakthroughs in classification into multiple categories with millions of labeled examples (e.g. Krizhevsky (2009); Szegedy et al. (2015); He et al. (2016) and many others). These methods rely on a lot of annotated data for training in order to perform well. Unfortunately, labeling is an inefficient and expensive progress, so learning from unlabeled data is desirable for many complex tasks. At the same time, much of human knowledge and learning is obtained by unsupervised observations Grossberg (1994).

The goal of this work is to show that semantically meaningful classes can be learned with minimal supervision. Given a set of objectness proposals, we use the activations of foreground objects in order to learn deep features to cluster the available data while simultaneously learning the embedding in an end-to-end manner. More specifically, we propose a differentiable clustering approach that learns better separability of classes and embedding. The main idea is to store the potential cluster means as *weights* in a neural network at the higher levels of feature representation. This allows them to be *learned* jointly with the potential feature representation. This differentiable clustering approach is integrated with Deep Neural Networks (e.g. Szegedy et al. (2015)) to learn semantic classes in an end-to-end fashion without manual class labeling.

The idea of doing this 'end-to-end' is that gradient descent can not only learn good weights for clustering, it can also change the embedding to allow for better clustering without the use of labels. We see that this leads to better feature representation. Our results show also that different object categories emerge and can later be retrieved from test images never before seen by the network, resulting in clusters of meaningful categories, such as cars, persons, bicycles.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

In this work we use given segmentation objectness masks, which are candidate objects without labels. This can be extended by using an independent objectness-based object generation mechanism Pathak et al. (2017); Faktor and Irani (2014) or by using unsupervised motion segmentation in videos or structure from motion Vijayanarasimhan et al. (2017).

2 Related Work

Unsupervised learning (Barlow (1989)) and unsupervised deep learning (Bengio, Courville, and Vincent (2012), Bengio (2012), Bengio and others (2009)) are central topics to Machine Learning. Unsupervised deep learning has been shown to improve results on classification tasks per Erhan et al. (2010), especially given small datasets and complicated high dimensional data such as video. This has been explored by many representations including sequence to sequence learning and textual representations (Radford, Jozefowicz, and Sutskever (2017), Ramachandran, Liu, and Le (2016)).

Our work focuses on unsupervised deep learning for discovering visual object categories. This has also been shown to improve results such as in Doersch and Zisserman (2017). Unsupervised discovery of visual objects has been a large topic of interest in computer vision (Sivic et al. (2005); Russell et al. (2006); Singh, Gupta, and Efros (2012); Bach and Jordan (2005); Kwak et al. (2015); Pathak et al. (2017)). Building specialized, deep embeddings to help computer vision tasks is also a popular approach such as in Agrawal, Carreira, and Malik (2015). Transfer learning from supervised tasks has proven to be very successful. Further, Agrawal, Carreira, and Malik (2015) propose learning the lower dimensional embedding through unsupervised learning and show improved performance when transfered to other supervised tasks.

Despite the popularity of building different embeddings, there is little work investigating the use of clustering to modify the embedding in an end-to-end deep learning framework. Bottou and Bengio (1995) investigate a differentiable version of the kmeans algorithm and examine its convergence properties. Our work focuses on *learnable* feature representations (instead of fixed ones as in Bottou and Bengio (1995)) and introduces memory units for the task.

3 Unsupervised deep clustering

Our unsupervised deep clustering is inspired by Bottou and Bengio (1995), who consider differentiable clustering algorithms. We differ from this approach because the features we cluster also change with backpropogation. In our work, we add a kmeans-like loss that is integrated end-to-end. Our idea is to store the potential cluster means as *weights* in the network and thus have them be *learned*. The proposed clustering is done simultaneously while building an embedding. Given information of a potential object vs background (binary labels), clustering in a differentiable way provides a better embedding for the input data. We show that this method can be used for meaningful semantic retrieval of related objects.

3.1 Embedding with clustering

We train a convolutional neural network (CNN) to predict foreground and background using oracle labels of patches of objects and background images. Concurrently, we learn the clustering of objects by imposing constraints that will force the embedding to be partitioned into multiple semantically coherent clusters of objects without explicit labels for different objects.

For our experiments, we use random initialization on the fully-connected layers (the last two layers) and we add the differentiable clustering module after the second to last layer. Note that we only cluster the foreground labels as background activations are not of interest for clustering; the classifier can predict foreground vs background with high accuracy (above 90%).

The objective function is shown in Equation 1.

$$L_k = \frac{1}{2N} \sum_{n=1}^{N} min_k[(x_n - w_k)^2] \tag{1}$$

In this equation, N is the number of samples, k is the number of defined clusters, w is the "weight" (theoretically and typically the mean of the cluster) for each k, and x is the activations from the fully

connected layer before the classification fully connected layer. This is differentiable and the gradient descent algorithm is shown in Equation 2.

$$\delta w_k = w'_k - w_k = \sum_{n=1}^{N} \begin{cases} l_r(x_n - w_k) & \text{if } k = s(x_n, w) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $s(x_n, w) = argmin_k[x_n]$ and l_r is the learning rate. We also add L_2 regularization over the weights to the loss $L_2 = \sum_j w_j^2$. Furthermore, we use a custom clustering regularization loss L_C that enforces that the clusters are evenly distributed as defined in Equation 3 and Equation 4.

$$L_C = \frac{1}{NK} \sum_{k=0}^{K} \sum_{j=k}^{K} |count_k - count_j| \tag{3}$$

$$count_k = \sum_{n=0}^{N} \begin{cases} 0 & \text{if } argmin_k[x_n] = 0 \\ 1 & \text{if } argmin_k[x_n] = 1 \end{cases} \tag{4}$$

The final loss to be optimized is shown in (Equation 5)

$$L = L_k + \alpha_r L_2 + \alpha_c L_C \tag{5}$$

where α_r and α_c are hyperparameters which are tuned during the training. For our method, we use $\alpha_r = 0.25$ and $\alpha_c = 1$. We apply this loss to every point that is labeled as potentially an object and ignore the background ones when clustering. This way we learn foreground vs background and then learn clustering of the foreground activations. Optimization was performed with a 'RMSProp' optimizer, with a learning rate of 0.045, momentum 0.9, decay factor 0.9, and ϵ of 1.0.

4 Experimental evaluation

We experiment with a toy example using CIFAR10 and a more challenging example using Cityscapes.

4.1 CIFAR10 dataset

We first test the proposed unsupervised clustering approach on the CIFAR10 Krizhevsky (2009) dataset. The goal of this experiment is to test if clustering can uncover separate categories in a simple toy problem with a two class setting.

Clusters	Automobile	Dog
Cluster 0	68.5%	17.9%
Cluster 1	31.5%	82.1%

Table 1: Unsupervised clustering results on CIFAR10 for discovery of two classes. Per cluster accuracy for each of the two given classes on the test set (class labels are unknown during training).

We selected as an example the dog and automobile classes to label as foreground. We then train a network from scratch based on the Network in Network architecture (NiN) of Lin, Chen, and Yan (2013) from scratch for our experiments. All other classes of CIFAR are considered background for this experiment. By attaching our modified clustering objective function to the next to last layer, we attempt to cluster dog and automobile without labels.

We can see in our simple experimental results that classes are naturally clustered with the majority of examples correctly assigned. Table 1 shows quantitative results on the test set. As seen 68.5% of the automobile classes and 82.1% of the dog examples are correctly assigned to separate clusters. Note that in these cases, the concepts and classes of dog and automobile are unknown to the training algorithm and we are just looking at them after clustering for evaluation.

Classes	Cluster 0	Cluster 1
Person	4320	138
Rider	676	138
Car	1491	4399
Truck	60	69
Bus	49	89
Train	17	16
Motorcycle	88	205
Bicycle	795	787

Table 2: Unsupervised clustering of objects from Cityscapes using our method. The table shows number of examples assigned to each learned cluster (for K=2).

4.2 Cityscapes dataset

The Cityscapes dataset (Cordts et al. (2016)) is a large-scale dataset that is used for evaluating various classification, detection, and segmentation algorithms related to autonomous driving. It contains 2975 training, 500 validation, and 1525 test images, where the test set is provided for the purposes of the Cityscape competition only. In this work, we used the training set for training and the validation set for testing and visualizing results (as the test set has no annotation results). Annotation is provided for classes which represent moving agents in the scene, such as pedestrian, car, motorcycle, bicycle, bus, truck, rider, train. In this work we only use foreground/background labels and intend to discover semantic groups from among the moving objects.

4.3 Weakly supervised discovery of classes

In this experiment we considered the larger, real-life dataset, Cityscapes (Cordts et al. (2016)), described above to see if important class categories, e.g. the moving objects in the scene can be clustered into semantically meaningful classes. We extract the locations and extents of the moving objects and use that as weak supervision. Note the classes are uneven and car and person dominate.

We show results clustering 8 categories into 2 and 3 clusters despite the rarity of some of them (such as bicycle). All results below are presented on the validation set. We report the results in terms of the number of object patches extracted from the available test images. For this dataset, the CNN architecture is based on the Inception architecture proposed by Szegedy et al. (2015). Since there are a small number of examples, we pre-train only the convolutional layers of the network.

Results on clustering the 8 classes of moving objects into 2 and 3 clusters are presented in Table 2 and Table 3 respectively for the learned embedding by the proposed approach and the baseline embedding. The baseline embedding is calculated by fine-tuning the same architecture in the same manner, but without our loss (Equation 5); it uses the same amount of information as input as our embedding. For this experiment, we apply standard kmeans on both activations after training is completed. We see here that our method provides better clustering for the two dominant classes in the dataset (car and person). On the other hand, the baseline embedding clusters on one class only, similar to the two class case. We have consistently observed this behavior for different runs and hypothesize this is due to the sparse nature of the baseline embedding and it's activations.

Figure 1 visualizes the three retrieved clusters (color-coded) when clustering into 3 clusters with our approach. We can see that people (in blue) and cars (in green) are often correctly retrieved. Bikes are more rare and may be more often mistaken, for example in cases where a portion of the patch contains part of a car, or since the bicycle very often has a person riding it. Still this is exciting result, given that it is learned by not providing a single class label during training.

5 Conclusions

We propose a differentiable clustering objective which learns to separate classes during learning and build a better embedding. The key idea is to be able to learn the clusters which are stored as weights, and simultaneously learn the feature representation and the clustering of the data. Our results show that the proposed approach is useful for extracting semantically related objects.

Classes	Our method			Baseline		
	Cluster 0	Cluster 1	Cluster 2	Cluster 0	Cluster 1	Cluster 2
Person	151	4315	17	4482	1	0
Rider	258	551	7	816	0	0
Car	5195	950	180	6312	13	0
Truck	89	39	5	131	2	0
Bus	127	20	5	152	0	0
Train	25	9	1	35	0	0
Motorcycle	127	76	4	207	0	0
Bicycle	1128	541	450	2119	0	0

Table 3: Unsupervised clustering on the Cityscapes dataset with 3 clusters. The table shows the number of examples assigned to each learned cluster. Our method (left) and baseline (right). Our method results in 69.98% accuracy.

Figure 1: Visualization of clusters learned by our method (for K=3). From the figure, the green class is responsible for retrieving cars, the blue one persons, and the red one bicycles. We can see that both cars and persons are discovered well but bicycles, a rarer class, can be confused with a person or with a partially visible car in the background.

References

Agrawal, P.; Carreira, J.; and Malik, J. 2015. Learning to see by moving. *CVPR*.

Bach, F. R., and Jordan, M. I. 2005. Learning spectral clustering. *NIPS*.

Barlow, H. 1989. Unsupervised learning. *Neural computation*.

Bengio, Y., et al. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning* 2(1):1–127.

Bengio, Y.; Courville, A. C.; and Vincent, P. 2012. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*.

Bengio, Y. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 17–36.

Bottou, L., and Bengio, Y. 1995. Convergence properties of the k-means algorithms. In *Advances in neural information processing systems*, 585–592.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3213–3223.

Doersch, C., and Zisserman, A. 2017. Multi-task self-supervised visual learning. *arXiv preprint arXiv:1708.07860*.

Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; and Bengio, S. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11(Feb):625–660.

Faktor, A., and Irani, M. 2014. Video segmentation by non-local consensus voting. *BMVC*.

Grossberg, S. 1994. 3-d vision and figure-ground separation by visual cortex. *Perception and Psychophysics*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. *CVPR*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images.

Kwak, S.; Cho, M.; Laptev, I.; Ponce2, J.; and Schmid, C. 2015. Unsupervised object discovery and tracking in video collections. *ICCV*.

Lin, M.; Chen, Q.; and Yan, S. 2013. Network in network. *arXiv preprint arXiv:1312.4400*.

Pathak, D.; Girshick, R.; Dollar, P.; Darrell, T.; and Hariharan, B. 2017. Learning features by watching objects move. *CVPR*.

Radford, A.; Jozefowicz, R.; and Sutskever, I. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

Ramachandran, P.; Liu, P. J.; and Le, Q. V. 2016. Unsupervised pretraining for sequence to sequence learning. *arXiv preprint arXiv:1611.02683*.

Russell, B. C.; Efros, A. A.; Sivic, J.; Freeman, W. T.; and Zisserman, A. 2006. Using multiple segmentations to discover objects and their extent in image collections. *CVPR*.

Singh, S.; Gupta, A.; and Efros, A. A. 2012. Unsupervised discovery of mid-level discriminative patches. *ECCV*.

Sivic, J.; Russell, B. C.; Efros, A. A.; Zisserman, A.; and Freeman, W. T. 2005. Discovering objects and their location in images. *ICCV*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. *CVPR*.

Vijayanarasimhan, S.; Ricco, S.; Schmid, C.; Sukthankar, R.; and Fragkiadaki, K. 2017. Sfm-net: Learning of structure and motion from video.

Adversarial Localization Network

Lijie Fan
Tsinghua University
flj14@mails.tsinghua.edu.cn

Shengjia Zhao
Stanford University
sjzhao@stanford.edu

Stefano Ermon
Stanford University
ermon@stanford.edu

Abstract

We propose the *Adversarial Localization Network*, a novel weakly supervised approach to generate object masks in an image. We train a corruption net to mask out regions of the image to reduce prediction confidence of the classifier. To avoid generating adversarial artifacts due to vulnerability of deep networks to adversarial perturbation, we also co-train the classifier to correctly predict the corrupted images. Our approach could efficiently produce reasonable object masks with a simple architecture and few hyper-parameters. We achieve competitive results on the ILSVRC2012 dataset for object localization with only image level labels and no bounding boxes for training.

1 Introduction

Supervised convolutional neural networks have achieved near human performance on several computer vision tasks, such as object localization [12, 15, 9, 26, 14] and semantic segmentation [16]. However, localization and segmentation annotations are often expensive and hard to obtain (compared to image-level labels such as object classes). To resolve this problem, weakly-supervised approaches are developed using image level labels only. These weakly supervised methods have reached competitive performance on various computer vision tasks, such as object detection [17, 27, 21] and semantic segmentation [18, 13].

A traditional approach [2, 3, 5] to weakly supervised localization or segmentation is to perturb (e.g., by occlusion) regions of the image to maximally decreases a classifier's prediction confidence. The assumption is that regions containing the object are the most important one for the decision, and as a result, they will significantly decrease a classifier's prediction confidence if perturbed. However, it has been shown that classifiers are very easily "fooled" [4] as a very small or even imperceptible amount of adversarial perturbation can completely alter a classifier's prediction. This effect, which we shall refer to as vulnerability to adversarial noise, is attributed to the fact that in high dimensions, there likely exists a direction where a small input change leads to a significant change in the output [6]. Therefore previous methods partition the space into very coarse grids, and apply the same perturbation to *all* pixels in each grid. This limits the dimensionality of the adversarial noise and alleviates this problem. However this usually means very coarse grained and inaccurate localizations or segmentations.

This paper presents two contributions that improve on the shortcomings of previous models. We draw inspiration from Virtual Adversarial Training (VAT)[8], which uses adversarial training to make a classifier more robust against adversarial noise. We perform adversarial training [24] between corruption network and classifier, so that a classifier's decisions will only be altered if the image is truly corrupted beyond recognition, which in our experiments, often leads to occlusion of the correct object. This allows us to learn much finer grained object masks compared to previous work [2, 3, 5]. In addition, we use super-pixels which incorporate structure about object boundary, rather than equally sized grids. The resulting mask follows the object contour better given the same resolution. In our experiments on the ILSVRC 2012 dataset, our method obtains superior or comparable results

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

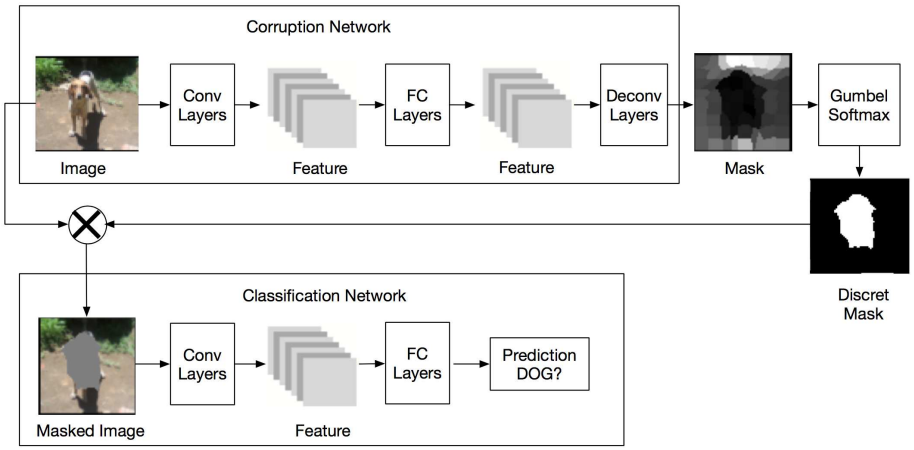


Figure 1: Adversarial Localization Network: The overall architecture of our proposed approach. The corruption network takes original images as inputs and produces corresponding saliency masks. The classification takes the masked images as input and produces classification scores. These two networks are trained in an adversarial way.

with previous state-of-the-art with a fairly small model architecture with few hyper-parameters or post processing.

2 Proposed Approach

In this section, we give detailed descriptions of our approach and design choices. Our model is shown in Figure 1. We train two components, a "corruption process" m_ϕ and a "classification network" c_θ. The corruption process consistent of three parts.

1) A deep network parameterized by ϕ takes the image x as input, and outputs a probability of corruption for each pixel. Details about the corruption network are given in Section 2.1.

2) Segment the image into super-pixels. For each super-pixel, we merge (by averaging) the occlusion probability of all its included pixels, and sample a binary decision to occlude or not occlude based on the occlusion probability. This procedure is described in Section 2.2.

3) We apply the occlusion mask to the image by setting occluded pixels to value 0. We denote the resulting corrupted image as $\tilde{x} = m_\phi(x)$.

On the other hand, the classifier network c_θ takes as input the corrupted image $\tilde{x}$, and outputs the resulting logits scores $c_\theta(\tilde{x})$ where $c_\theta(\tilde{x})_j$ is the classifier's prediction for the j-th class. Let y be the ground truth label. The classifier net is trained to increase the softmax cross entropy with the true label $\mathrm{Softmax\_CE}(c_\theta(\tilde{x}), y)$, while the corruption network is trained to reduce prediction confidence. The training procedure and regularizations used are described in Section 2.3.

2.1 Corruption Network

Because the extent of an object depends both on local information (such as position of edges) and global informations (such as object category), we design the corruption network to capture both global and local information. To capture global information, we first use a convolutional network with fully connected layers to map the input into high level features, then we use the inverted architecture (deconvolution) to decompress high level features into occlusion probabilities for each pixel. This process with fully connected layers ensures that we are able to model global information about objects (such as its category). However this also implies that details such as exact edge positions will get lost in the convolution-pooling process. Empirically we observe that such architecture produces inaccurate object boundaries. Therefore we also add a large number of residual connections in the intermediate layers. This ensures that local information can be more easily forwarded and utilized.

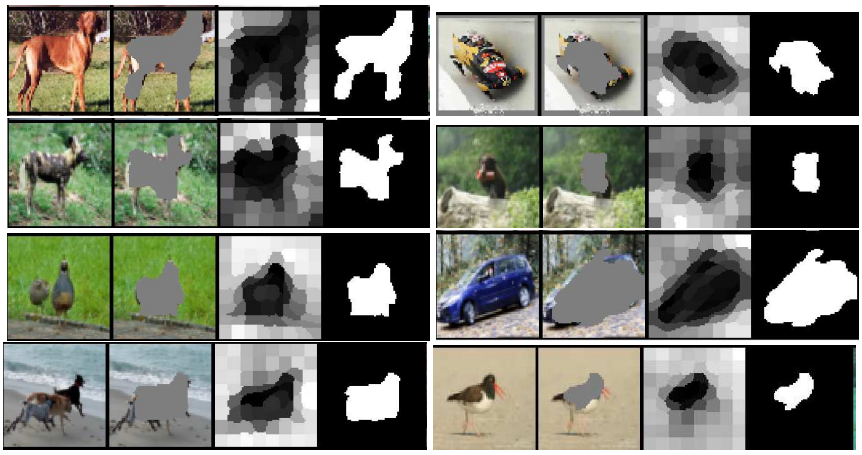


Figure 2: Examples of the generated results of our Adversarial Localization Network: First column: original input images; Second column: the masked images; Third column: mask intensity maps; Fourth column: the generated object segmentation mask.

2.2 Pixel Level Probabilities to Masks

We design a procedure to convert pixel level probabilities produced by the corruption network into (discrete) masks that are consistent with the input image, and allow for efficient training with back-propagation.

One observation is that object boundaries almost always align with image edges (points with sharp transitions of color or intensity). Therefore the edges of the image naturally partition the image into super-pixels(group of pixels). We use the super-pixel computation method proposed in [11], and each super-pixel can be treated as a whole when deciding whether it belongs to the object. Therefore we propose to average the masking probabilities for each super-pixel, and produce a masking decision for each super-pixel as a whole. This reduces the dimensionality of possible perturbations which alleviates the vulnerability to adversarial perturbations.

Next, we discretize the masking probability for each super-pixel to a binary masking decision (mask or not mask) using the Gumbel trick [10] with the straight-through estimator [7]. This is described by the following procedure. Let $\boldsymbol{p} = (p, 1-p)$ be a probability vector of a Bernoulli variable. Sample.

$$u_1, u_2 \sim \text{Uniform}\,(0,1)\,, \qquad z_i = -ln\,(-ln\,(-u_i))\,, i = 1,2$$

Then we draw a sample x from the Bernoulli distribution $\boldsymbol{p}$ by

$$x = \mathbb{I}(p + z_1 > 1 - p + z_2)$$

where $\mathbb{I}$ is the indicator variable. Gradient through this discretization indicator operation can be approximated by the straight-through estimator [7].

2.3 Adversarial Training Procedure

As described above, we train the networks with an adversarial procedure. For the corruption network, for each data point, label pair x, y we minimize

$$\min_{\phi} \mathcal{L}_{\text{Corruption}} = \sum_{j=1}^{K} \mathbb{I}_j c_\theta(m_\phi(x))_j - \frac{1}{K-1}(1-\mathbb{I}_j)c_\theta(m_\phi(x)) \tag{1}$$

where K is the number of object classes, and $\mathbb{I}_j$ is the indicator variable of whether class j is the top prediction made by the network: $\mathbb{I}_j = 1$ if $j = \arg\max_{1 \le i \le K} c_\theta(m_\phi(x))_j$. Intuitively this objective encourages the corruption network to corrupt the image such that the classifier is in a maximum state of confusion: it produces the same logit score for each class. For the classifier network, we simply train to maximize the softmax cross entropy with the true label

$$\max_{\theta} \mathcal{L}_{\text{classifier}} = \text{Softmax\_CE}(c_\theta(m_\phi(x)), y)$$

One trivial solution is for the mask to occlude the entire image, so that classifier has no information at all to make any prediction, minimizing the corruption network loss. Therefore add a regularization term that penalizes the total area of the generated masks so that the corruption network will only generate the minimum mask necessary to confuse the classifier.

$$\min_{\phi} \mathcal{L}_{\text{Corruption}} + \lambda_{\text{Reg}} \sum_{i}^{\text{all pixels}} \mathbb{I}(i \text{ is occluded})$$

where $\mathbb{I}$ is the indicator variable of events. This results in well-interpretable image masks that often occludes the correct object.

2.4 Balancing Adversarial Training

A practical concern of our method, is convergence to undesired local optimum. For example, the corruption net may occlude nothing, which is a local optima because adding a small amount of occlusion will lead to very little increase in classifier confusion, but incurs a penalty because we penalize the area of the occlusion mask. On the other hand, the corruption net may occlude too much if the penalty weighting λ_{Reg} is too small. We implemented an adaptive strategy to fix this weighting. We observe that the best results are usually obtained when the classifier accuracy on the corrupted image is around 10%, so we decrease (so occlude more) or increase (so occlude less) the weighting λ_{Reg} depending on the classifier accuracy.

Another concern of adversarial training is the delicate equilibrium that is often broken if either component trains significantly faster. In our implementation we observe that the classifier usually trains significantly faster than the corruption network. This causes instability and forces the occlusion network to generate a larger mask. We found the above adaptive strategy to work well for this issue as well: if the classifier accuracy is above 10% we only train the corruption net. If accuracy is below 10% we perform joint training. We have found that this simple strategy significantly improves training stability and quality of generated masks.

3 Experiments

We conduct our experiments on ILSVRC 2012 localization dataset [19] with 1000 classes and 1.2 million images. We train our model on the training set with only image level class labels, and evaluate on localization labels provided in the validation set.

3.1 Evaluation metric

We use the following two types of evaluation metrics:

Metric 1: Top-1 accuracy This is the localization evaluation metric officially provided by ILSVRC. For each test image, provide a classification prediction and one object bounding box prediction. Compute the percentage of test samples which are classified correctly and have more than 50% intersection over union (IoU) between the predicted bounding box and one of the ground-truth bounding boxes.

Metric 2: Top-1 accuracy with state-of-the-art (SOTA) classifier We train our network with a fairly simple classifier for stability and speed of the adversarial training procedure. Therefore performance can be improved when we replace the class prediction made by our smaller network with that made by a state-of-the-art classifier, while keeping the same bounding box prediction. We use a InceptionV4 network [22] to produce the class prediction instead.

3.2 Bounding Box Generation

We convert masking probabilities to a bounding boxes in our localization experiment. To do this we first binarize the masking probability map with an adaptive threshold.

$$\text{Threshold} = 0.7 \times \text{Mean}(\text{map}) + 0.3 \times \text{Max}(\text{map})$$

Then we take the bounding box that covers the largest connected component as our prediction.

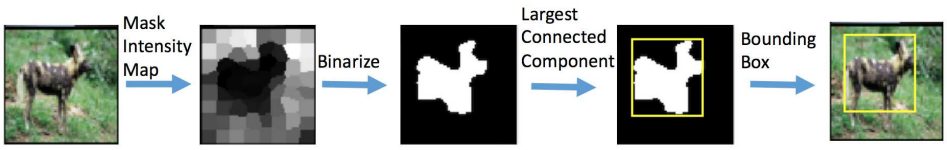


Figure 3: Bounding box generation pipeline: 1.Generate masking probability map with the corruption network; 2.Binarize the map with adaptive thresholds; 3.Compute the Largest Connected Component in the binarized mask; 4.Generate its bounding box.

3.3 Comparison with the baseline model

We compare with other state-of-the-art weakly supervised localization models [20, 23, 27]. The results are shown in Table 1. Our Adversarial Localization Network performs 3.1% and 1.9% better than the baseline model on Top-1 accuracy with SOTA classifier and Top-1 accuracy respectively. Importantly. Note that we train on a much smaller architecture compared to baseline models: the input is a 64×64 image and we use a simple 4-layer convolutional network. Each evaluation of our model only takes 0.093 seconds while the fastest baseline model takes 0.14 seconds. This indicates significant speed up with similar or even better performance.

Methods	Metric 1	Metric 2
Max box	41.0%	34.3%
Backprop [20]	–	38.7%
Layer-wise Relevance Propagation [23]	42.2%	–
Global Average Pooling(Baseline)[27]	51.9%	43.6%
Adversarial Localization Networks	$\mathbf{56.5\%}$	$\mathbf{45.5\%}$

Table 1: Performance comparison between Adversarial Localization Network(ALN) and state-of-the-art models

3.4 Necessity of Adversarial Training

To demonstrate the necessity of our adversarial training procedure, we only train the corruption network with the same loss as Eq.(1). We show that it is very easy for the corruption net to generate adversarial artifacts to "fool" the classification network, even when we are using super-pixel representation for the input image, or use increased penalty for the size of the generated mask. If the classification network is also trained, almost no adversarial artifact are observed.

Figure 4: Adversarial Examples: To show the necessity of adversarial learning, if we freeze the classification network and only train the corruption network, it could be trained to generate adversarial examples to "fool" the classifier easily, however, those adversarial examples make no sense to humans.

4 Conclusion

In this work, we presented a novel weakly supervised approach for object localization/segmentation. In particular, we make two major contributions: applying adversarial training to avoid vulnerability of deep networks to adversarial perturbation and make classifier more robust, and using super-pixels, which reduces the dimensionality of possible perturbations, alleviates the vulnerability to adversarial perturbations and leads to better detection boundary. We out-perform state-of-the-art models in performance and have significantly faster training and evaluation time.

References

[1] M. Van den Bergh, and X. Boix, and G. Roig and B. de Capitani and L. Van Gool, "SEEDS: Superpixels extracted via energy-driven sampling," *European conference on computer vision*, 2012

[2] P. Dabkowski, and Y. Gal, "Real Time Image Saliency for Black Box Classifier," *arXiv preprint arXiv:1705.07857*, 2017.

[3] R. Fong, and A. Vedaldi , "Interpretable Explanations of Black Boxes by Meaningful Perturbation," *arXiv preprint arXiv:1704.03296*, 2017.

[4] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint arXiv:1610.08401*, 2016.

[5] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan, "Object region mining with adversarial erasing: A simple classification to semantic segmentation approach," *arXiv preprint arXiv:1703.08448*, 2017.

[6] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2016.

[7] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[8] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," *arXiv preprint arXiv:1507.00677*, 2015.

[9] K. Duan, D. Parikh, D. Crandall, and K. Grauman. Discovering localized attributes for fine-grained recognition. In *CVPR*, 2012.

[10] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*. US Govt. Print. Office, 1954, no. 33.

[11] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[12] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[13] A. Khoreva, R. Benenson, M. Omran, M. Hein, and B. Schiele. Weakly supervised object boundaries. In *CVPR*, 2016.

[14] M. Kiapour, K. Yamaguchi, A. C. Berg, and T. L. Berg. Hipster wars: Discovering elements of fashion styles. In *ECCV*, 2014.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[16] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[17] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? – weakly-supervised learning with convolutional neural networks. In *CVPR*, 2015.

[18] D. Pathak, P. Krähenbühl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *ICCV*, 2015.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[20] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR Workshop*, 2014.

[21] K. K. Singh, F. Xiao, and Y. J. Lee. Track and transfer: Watching videos to simulate strong human supervision for weakly-supervised object detection. In *CVPR*, 2016.

[22] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, 2017, pp. 4278–4284.

[23] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[25] P. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *International Conference on Machine Learning*, 2014, pp. 82–90.

[26] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev. PANDA: Pose Aligned Networks for Deep Attribute Modeling. In *CVPR*, 2014.

[27] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.

Learning by Generating Mental Imagery from Limited Labeled Dataset

Esube T. Bekle[1], **Wallace Lawson**[2]
[1]NRC Postdoctoral Fellow
[2]Navy Center for Applied Research in Artificial Intelligence
US Naval Research Laboratory
Washington, DC 20375
esube.bekele.ctr@nrl.navy.mil

Abstract

Most real-world scenarios such as surveillance and life-long learning on mobile robotic platforms are characterized by a scarcity of labeled training datasets for deep learning. Training deeper networks on limited labeled datasets is non-trivial. Weaker supervision has the potential to alleviate this problem in part. We propose deep convolutional generative adversarial networks (DCGAN) that learn to produce a "canonical image" from input images using an internal representation of the expected distributions of the input data. This canonical image is what the DCGAN "imagines" that the input image category might look like under ideal conditions, i.e. iconic representation. A DCGAN learns this association by training an encoder to capture salient features from the original image and a decoder to convert salient features into its associated canonical image representation. Our new approach, which we refer to as a Mental Image DCGAN (MIDCGAN), learns features that are useful for classification, and that this in turn has the benefit of helping single or few shot recognition from limited labeled datasets. We demonstrate our approach on object instance recognition and handwritten digit recognition tasks.

1 Introduction

Despite the early success of deep convolutional neural networks in computer vision, training deeper networks with limited labeled datasets is a challenge. In such situations, leveraging strategies that are similar to how humans learn would be beneficial. Consider the way that children interact with objects when they are very young (6; 15): during their interaction, children look at objects from different perspectives. Eventually, they build up a preference for certain viewpoints after examining objects for a long period of time. They use that preferred viewpoint of the object as their preferred 'canonical image' atlas or internal representation of the object. When a similar object is encountered from a different viewpoint, they try to map it to the canonical preferred viewpoint they formed. This process is mostly with little to no supervision. In this paper, we consider the question of what would happen if we were to train a deep convolutional generative adversarial network in the same manner (see Fig. 1). For this, we provide the canonical image (i.e., an ideal representation for a category), to map the image to its canonical representation. The form of this supervision is a high-level and weak. Rather than using labels, instead we use the image of the object at different viewpoints. The Mental Image DCGAN (MIDCGAN) is trained to associate each of these samples in a specific input distribution back to the canonical image. This association is learned using a GAN architecture (see Fig. 2) with a generator composed of an encoder and decoder. MIDCGAN trains the encoder to learn salient bottleneck features for each class while the decoder learns to generate a canonical image from bottleneck features. We will show that MIDCGAN bottleneck features are better suited for

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

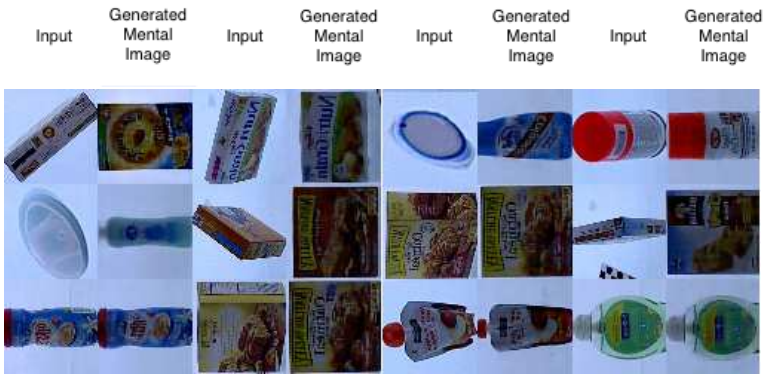


Figure 1: MIDCGAN input (first column and every other column) and generated canonical image (second column and every other column). The images are selected from the BigBird database (12).

learning than those features that are generated without the benefit of using a canonical image (or regular DCGAN).

Stated formally, a typical learning task seeks to learn the data distribution, $p(x)$ mapping to a class or category label y or $p(y|x)$. The diversity of samples in the training data are limiting in the way the learner represents the category class internally. The MIDCGAN approach on the other hand provides a canonical image $\tilde{x}$ as target to be learned and stored as representation of a category expected target distribution. Hence, the diversity of the input samples will no longer be a problem (as the ideal canonical or preferred representation is supplied in the form of the canonical image) and the network can be trained with limited training samples with decreased performance degradation. The learner maps the input data distribution, $p(x)$, to this canonical representation of the category, $\tilde{x}$, i.e the conditional $p(\tilde{x}|x)$. During this mapping process, the MIDCGAN creates an internal bottleneck feature vector that is better representative of the input distribution mapping to the canonical image. In summary, the canonical image forces the network to focus on the iconic parts and features of the object.

Generative adversarial networks (GAN) (2) and their variants such as improved GAN (11), categorical GAN (13) have been shown to learn features for classification. MIDCGAN is inspired partly by the work of image-to-image mapping (5) and image editing (14) and inpainting using GANs (9). DCGANs have been employed to learn expressive features for classification in unsupervised (10) and semi-supervised manner (11; 13).

We demonstrate the effectiveness of mental image DCGANs (MIDCGAN) on three different problems. First, we demonstrate this for handwritten digits as proof of concept. In this case, we assume that a helpful tutor has provided an ideal representation of the digit, so the canonical image is a stencil. Next, we demonstrate the performance of MIDCGAN on instance based object recognition, in which, the helpful tutor provides the system with an iconic view of the object. Finally, we evaluate performance on a dataset that was not used to learn salient features. In all cases, we show that MIDCGAN substantially outperforms its DCGAN equivalent.

2 Methodology

DCGAN learns generative models using the joint adversarial training of a generator network and discriminator network. The generator maps a noise vector z to generated image using $\hat{x} = G_{\theta^G}(z)$ that increasingly represents the underlying target distribution $p(x)$ during training. The discriminator predicts whether a sample is from the real target data distribution, $p(x)$, or the generated data distribution, $p(\hat{x})$, and is represented by $D_{\theta^D}(x, \hat{x})$ (11). For the sake of simplicity, we refer to the generator as $G(z)$ and the discriminator as $D(x)$. We train the generator network using both l_2 loss and the adversarial discriminative loss so that it learns features useful for classification. Unlike DCGAN, the generator network is not generating samples from a random noise vector, z, rather they are derived from the encoder of an autoencoder (9). This enables MIDCGAN to encode important features from the input distribution, $p(x)$, into a bottleneck feature vector denoted as $\bar{z}$.

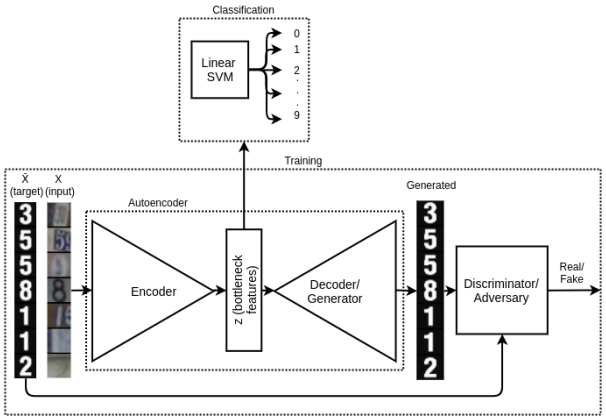


Figure 2: The MIDCGAN architecture built with either feedforward convolutional or ResNet blocks.

2.1 Architecture

The MIDCGAN architecture is based on DCGAN with the generator replaced by a full autoencoder (i.e., encoder-decoder). Given an input selected from a distribution $p(x)$, the encoder network produces a representation of the object in the form of a bottleneck feature vector, $\bar{z}_n$, of length n. The decoder network then takes the output of the encoder, $\bar{z}_n$, and produces the a sample from the generated distribution, $p(\hat{\tilde{x}})$. Here, $\tilde{x}$ is the canonical image of x.

Fig. 2 shows an overview of the MIDCGAN architecture. The convolutional blocks in all the three components of MIDCGAN (encoder, decoder and discriminator) take two forms: either simple convolutional blocks that contain a sequence of convolution-batch normalization-activation (we call this simple network) or n residual blocks. We use $n = 5$ double convolutional BN-activation-convolution residual units (3) in each of the convolution blocks. The discriminator is of two types: an AlexNet style discriminator (similar to most DCGANs) and a discriminator that has the same architecture as the encoder. We refer to these as the AlexNet and Separate discriminators.

2.2 The Joint Autoencoder and Adversarial Loss

MIDCGAN is trained using joint adversarial and generator l_2 losses. In a regular autoencoder, the input and the target are the same and since there is no guidance for the encoder to select a specific mode of the multimodal underlying data distribution, $p(x)$. Therefore, the autoencoder tends to average modes resulting in a blurry average image (9). By adding the canonical image as a target in the l_2 loss component, it forces the network to produce a specific mode of the target distribution, $p(\tilde{x})$, instead (2; 9). Therefore, MIDCGAN generates images that are sharper and closer to the target earlier in the training than an equivalent regular DCGAN while at the same time learning useful classification features via its mapping of the input sample to the canonical image.

2.2.1 Autoencoder Reconstruction Loss

Here we used a normalized l_2 loss similar to (9) without the masking. The l_2 loss is given by the squared difference between the target canonical image, $\tilde{x}$, and the generated canonical image, $G(z)$. Substituting encoded features from input data distribution, $p(x)$, for z using the encoder, $E(x)$ results in Eq. 1

$$L_{l_2}(x, \tilde{x}) = \mathop{\mathbb{E}}_{x \in p(x), \tilde{x} \in p(\tilde{x})} ||\tilde{x} - G(E(x))||_2^2 \tag{1}$$

2.2.2 Adversarial Loss

According to (2), the regular adversarial two-player minmax game between the discriminator, $D(x)$, and the generator, $G(z)$ is given by Eq. 2.

Table 1: MNIST test error (%) with n labeled examples for different architectures at nBn=256. Note: SS is simple feedforward CNN encoder/decoder/discriminator and separate discriminator and RS is residual CNN encoder/decoder/discriminator and separate discriminator

Method/ Arch	Number of Examples					All
	10	20	50	100	200	
DCGAN Alexnet	53.39 $\pm$ 4.22	47.1 $\pm$ 3.63	34.44 $\pm$ 2.13	24.55 $\pm$ 1.56	19.54 $\pm$ 1.1	8.17
Springenber et. al. (13)	-	-	-	1.39 $\pm$ 0.28	-	**0.48**
Makhzani et. al. (8)	-	-	1.90 $\pm$ 0.1	-	-	0.85
Salimans et. al. (11)	-	16.77 $\pm$ 4.52	2.21 $\pm$ 1.36	**0.93 $\pm$ 0.07**	**0.9 $\pm$ 0.04**	-
MIDCGAN SS	**1.51 $\pm$ 0.27**	**1.22 $\pm$ 0.08**	**1.13 $\pm$ 0.09**	1.07 $\pm$ 0.08	0.99 $\pm$ 0.11	0.82
MIDCGAN RS	4.58 $\pm$ 2.65	1.72 $\pm$ 0.69	1.48 $\pm$ 0.43	1.17 $\pm$ 0.13	1.03 $\pm$ 0.07	0.68

$$\min_{G}\max_{D} \mathop{\mathbb{E}}_{x\in p(x)}[log(D(x))] + \mathop{\mathbb{E}}_{z\in p(z)}[log(1 - D(G(z)))] \tag{2}$$

In Eq. 3, $p(x)$ represents the actual data distribution of the dataset input to the encoder while $p(\tilde{x})$ represents the target canonical canonical image distribution.

$$L_{adv}(x, \tilde{x}) = \mathop{\mathbb{E}}_{\tilde{x}\in p(\tilde{x})}[log(D(\tilde{x}))] + \mathop{\mathbb{E}}_{x\in p(x)}[log(1 - D(G(E(x))))] \tag{3}$$

In practice (2; 9), this loss is implemented by training the discriminator and the encoder-generator pair using alternating SGD. The total joint loss used to train the encoder-decoder generator pair is the weighted sum of the Eq. 1 and Eq. 3 and is given as Eq. 4.

$$L_{total} = \lambda_{l_2} L_{l_2} + \lambda_{adv} L_{adv} \tag{4}$$

The training of MIDCGAN is shown in Algorithm 1. The convergence of MIDCGAN happens often early due to the dual mode selection because of the canonical image and the adversarial loss and hence the discriminator could tell a real target sample quickly.

Algorithm 1 Training MIDCGAN

$\theta^E, \theta^G, \theta^D \leftarrow HeNormal$ (3) ▷ initialize encoder, decoder/generator and discriminator params
repeat
  $X, \tilde{X} \leftarrow$ shuffled mini-batch ▷ X is the input image while $\tilde{X}$ is the canonical image
  $Z \leftarrow Encoder(X)$
  $\hat{\tilde{X}} \leftarrow Decoder(Z)$
  $L_{adv} \leftarrow log(D(\tilde{X})) + log(1 - D(\hat{\tilde{X}}))$ ▷ Compute adversarial loss for real and fake
  $\theta^D \leftarrow \theta^D - \nabla_{\theta D}(L_{adv})$ ▷ Update Discriminator params with the adversarial loss gradients
  $l_2 \leftarrow ||\tilde{X} - \hat{\tilde{X}}||_2^2$
  $L_T \leftarrow \lambda_{l_2} l_2 + \lambda_{adv} L_{adv}$ ▷ total loss as fraction of adversarial and l_2 losses
  $\theta^E \leftarrow \theta^E - \nabla_{\theta E}(L_T)$ ▷ Update Encoder params with the total loss gradients
  $\theta^G \leftarrow \theta^G - \nabla_{\theta G}(L_T)$ ▷ Update Decoder/Generator params with total loss gradients
until number of epochs

3 Results and Discussion

The bottleneck features z represent important aspects of the input distribution, $p(x)$. During test time, we need only the encoder to generate the bottleneck features $\bar{z}$. These features are then given to an l_2 SVM to perform classification. In this section, we evaluate classification accuracy on handwritten digit recognition (MNIST and SVHN) and object instance recognition (BigBIRD and RGBD).

3.1 MNIST with Stencils 'Canonical Images'

In this experiment, the encoder, decoder and the separate discriminator were all ResNet architectures with each convolution block replaced by 5 dual-convolutional residual units. Therefore, the separate discriminator is much deeper and more expressive than the more commonly used AlexNet discriminator. In Table 1, we compare different architectures trained with a bottleneck size of 256 against the

Table 2: SVHN test error (%) with n labeled examples for different architectures at nBn=2048.

Method/ Arch	Number of Examples		
	10	**100**	**1000**
Makhzani et. al. (8)	-	-	17.70± 0.3
Salimans et. al. (11)	-	-	8.11± 1.3
MIDCGAN SA	**17.8± 4.11**	**8.4± 0.68**	**7.15± 0.26**

Table 3: Object recognition accuracy using different databases and approaches.

Dataset	Method/ Arch	Pre-training	Examples per Class				
			50	**25**	**10**	**5**	**1**
BigBIRD	MIDCGAN	Self-supervised	**98.19 ± 0.14**	**95.45 ± 0.39**	**87.4 ± 0.59**	**79.8 ± 0.89**	**52.24 ± 0.64**
BigBIRD	DCGAN	Unsupervised	65.55 ± 0.42	54.7 ± 0.45	43.4 ± 0.54	32.6 ± 0.73	14.1 ± 0.89
RGBD	MIDCGAN	Transfer	**61.7 ± 0.25**	**59.7 ± 0.61**	**55.99 ± 0.54**	**51.4 ± 0.36**	37.6 ± 0.8
RGBD	DCGAN	Transfer	43.33 ± 0.5	40.34 ± 0.51	37.5 ± 0.46	32.1 ± 0.35	20 ± 0.8
RGBD	CNN (4)	Supervised	-	-	-	-	**63.9**

state-of-the-art (13; 8; 11) with a different number of labeled training samples to highlight limited supervision performance.

3.2 SVHN with Stencils 'Canonical Images'

MIDCGAN with simple feedforward blocks and AlexNet style discriminator outperforms all other architectures we experimented with for the SVHN dataset. As shown in Table 2, MIDCGAN produced the best performance among all the methods with a wide range of number of limited training samples.

3.3 Object Instance Recognition

In object instance recognition, rather than identifying a general class of objects (eg., a bottle), we instead identify the specific instance of the object (e.g., a coke bottle). We evaluate this on the BigBIRD (12) and the RGBD (7). The former dataset has 125 object instances, whereas the later has over 300 objects instances. The RGBD dataset has a depth component, but in our experiments we only used RGB data. We evaluate this in two different ways. In the first, we permitted the MIDCGAN to see the objects in the training set, with the first image chosen arbitrarily as the canonical image. In the second, MIDCGAN did not see any training image from RGB-D, but rather to use features that it had already learned in the BigBIRD dataset. We present both results in Table 3. For the first case, we have results with varying numbers of examples per class. With only a single instance of each class (i.e., *single shot recognition*), MIDCGAN recognizes objects 52.2% of the time, compared to 14% of the time for DCGAN. MIDCGAN still significantly outperforms the DCGAN in the second case of transfer learning.

4 Conclusion

In our experiments, we demonstrated the ability to use canonical images to improve recognition with deeper networks when a limited amount of training is available. Although MIDCGAN was demonstrated on representative viewpoints, the selection of the mental target image could be arbitrary. In cases when MIDCGAN was not sure about the object, the generated image did not resemble the actual representative image, or important details about the representative image were omitted. In essence, this could potentially provide another way to determine the confidence in the prediction of the object class. Several improvements could be made to MIDCGAN in the future including autonomous selection of the canonical images (including a set of mixture of canonical images with different views of the same object), and incorporating limited labels into the MIDCGAN network loss function. With such approach and an autonomous mobile robot, for instance, a wide range of objects could be learned with minimal supervision by incorporating only the available labels as another to the loss function and just by generative and adversarial components of the loss for those which do not have labels.

Acknowledgements

Wallace Lawson was supported by the Office of Naval Research, Esube Bekele was supported by the National Research Council.

References

[1] B. Browatzki, V. Tikhanoff, G. Metta, H. H. Bülthoff, and C. Wallraven. Active object recognition on a humanoid robot. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2021–2028. IEEE, 2012.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[4] D. Held, S. Savarese, and S. Thrun. Deep lerning for single-view instance recognition. *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[5] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

[6] K. H. James, S. S. Jones, L. B. Smith, and S. N. Swain. Young children's self-generated object views and object recognition. *Journal of Cognition and Development*, 15(3):393–401, 2014.

[7] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[8] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[9] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[10] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[11] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[12] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 509–516. IEEE, 2014.

[13] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[14] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision*, pages 318–335. Springer, 2016.

[15] C. Yu, L. B. Smith, H. Shen, A. F. Pereira, and T. Smith. Active information selection: Visual attention through the hands. *IEEE Transactions on Autonomous Mental Development*, 1(2):141–151, 2009.

Active Learning of Classification Models from Soft-Labeled Groups

Zhipeng Luo and Milos Hauskrecht
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
{zpluo, milos}@cs.pitt.edu

Abstract

Learning of classification models in practice often relies on nontrivial human annotation effort in which humans assign class labels to data instances. As this process can be very time consuming and costly, finding effective ways to reduce the annotation cost becomes critical for building such models. To solve this problem we develop a new approach that actively learns instance-based classification model from subpopulations (groups) of instances and their soft labels. A soft label represents a human estimate of the proportion of instances with one of the class labels in the subpopulation, or equivalently, the probability with which an instance with that class label is drawn from the subpopulation. To form the groups to be annotated, we develop a hierarchical active learning framework that divides the whole population into smaller subpopulations, which allows us to gradually learn a more refined model from the subpopulations and their soft labels. Our comprehensive experiments show that our method is competitive and outperforms existing approaches on reducing the human annotation cost.

1 Introduction

Learning of classification models from real-world data often requires heavy supervision on labeling data instances. The problem is that *instance-based* supervision is often time-consuming, and thus it may be unrealistic to assume that an arbitrarily large number of instances can be feasibly labeled. The key challenge then is to find ways to build accurate models from limited human supervision.

One popular approach to reduce the labeling effort is active learning which sequentially selects a subset examples to be labeled. However, instance-based active learning may still be insufficient, as the assumption that instances are easy for humans to label may not hold. For example when data instances are high-dimensional or very complex objects, e.g. Electronic Health Records, each instance labeling requires annotators with additional expertise and time.

To save annotation cost, another promising direction is to move from instance-based labeling and learning to *group-based* supervised learning. That is, instead of providing instance-based feedback a human gives labels on subpopulations or groups of instances. The essential advantage is that people can provide weak supervision on *multiple* instances at a time. In this work, we combine active learning and group learning to propose a new framework that seeks to build an instance-based binary classification model from human feedback on groups of instances. Our approach assumes the human feedback is provided in terms of a *soft range* label (e.g. *"70% to 90%" Positive*) that represents a weak estimate of the proportion of classes in the subpopulation (group), or equivalently, the probability with which examples that belong to one of the classes can be drawn from the subpopulation.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Our *Learning from Soft-Labeled Groups* framework, summarized in Algorithm 1, can actively learn a binary probabilistic model from soft-labeled groups. As initially there are no apparent groups in the data, our algorithm starts with a hierarchical clustering on all the unlabeled data (line 1), followed by a proper adjustment (line 2) to generate groups. Then we always maintain a fringe of groups which is a complete partition of the all the data (line 3), and perform standard active learning cycles (Line 4-9). We use maximum expected model change strategy to select the most promising group along the fringe to split, assess its child groups by human and replace the group with its children in the fringe.

Algorithm 1: The LSLG (Learning from Soft-Labeled Groups) Framework

Input: An unlabeled data pool $\mathcal{U}$; A labeling budget; Human annotator(s)
Output: An instance-based binary probabilistic model $P(y|\boldsymbol{x};\boldsymbol{\theta})$

1: $T \leftarrow$ Perform hierarchical clustering on $\mathcal{U}$
2: $T_G \leftarrow$ Adjust T to form a new tree of groups and learn description for each group (Section 2)
3: Iteration $t \leftarrow 0$; Current fringe $F^{(0)} \leftarrow \{(T_G)'s\ root\}$; $\boldsymbol{\theta}^{(0)} \leftarrow$ random initialization
4: **repeat**
5: Actively select a group G_* in $F^{(t)}$ based on $P(y|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$ (Section 3)
6: Obtain the labels of G_*'s children from annotator(s)
7: $F^{(t+\Delta t)} \leftarrow \{F^{(t)} - G_*\} \cup \{G_*\text{'s children}\}$; $t \leftarrow t + \Delta t$; ($\Delta t =$ # of G_*'s children)
8: Retrain the model $P(y|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$ based on current labeled groups $F^{(t)}$ (Section 4)
9: **until** the labeling budget runs out
10: **return** $P(y|\boldsymbol{x};\boldsymbol{\theta}^{(t)})$

An illustration example Suppose we want to learn a binary classification model for predicting hospital admissions (the target label y) for all the patients encountered in the Emergency Room (ER) based on the a set of measurements such as heart rate, blood pressure and temperature (the predictors $\boldsymbol{x}$). Initially, an ER clinician may estimate the chance of admission to be 20% $\sim$ 30% for the entire ER population, but this assessment may go up significantly to say 60% $\sim$ 70% for the subpopulation with a high heart rate (120-140), and may further rise to 70% $\sim$ 90% for a subpopulation of patients with both a high heart rate (140-160) and a low blood pressure (Diastolic$<$60, Systolic$<$100), which are the signs of a significant blood loss. Our approach aims to take advantage of such subpopulations and their soft assessments to obtain an instance-based classifier for each patient.

2 The Concept of Groups

Given a pool of unlabeled data instances, represented as a real number matrix $\mathcal{U}_{n\times m}$ consisting of n m-dimensional instances, we first perform a standard hierarchical clustering (using ward linkage [13]) on $\mathcal{U}$ to output a tree T.

The Group Description The initial clusters (the potential groups) consist of sets of instances. To describe each group efficiently to humans, we use conjunctive patterns over the input space features, which matches precisely the hypercube definition of a typical decision tree algorithm. To convert the groups to hypercube representation we employ a C4.5 [8] classifier to automatically learn the group descriptions [9]. More specifically, if we want to learn the description of a group G_i indexed at i, we mark all instances in G_i as **1** and the rest of data ($\mathcal{U} - G_i$) as **0**. Then a C4.5 classifier will output a set of hypercubes $\mathcal{C}(G_i)$ that could potentially describe G_i. The fitness of each hypercube $c \in \mathcal{C}(G_i)$ can be assessed by $F1score = \frac{2\times precision\times recall}{precision+recall}$. That is, the description of G_i is a hypercube which is $\arg\max_{c\in\mathcal{C}(G_i)} F1score(c)$.

The Group Formation When C4.5 is directly performed on the clusters of the original hierarchical tree T, there may exist some clusters that are not hypercube-shaped. As a consequence, their best description hypercubes still have intolerably low $F1scores$. To mitigate this issue, we need to adjust the original tree structure to form a new tree T_G such that only hypercube-like clusters are conserved in T_G. Formally, we define that a cluster is *hypercube-like* if its description hypercube satisfies a minimum $precision(\geq 0.5)$ and $recall(\geq 0.5)$. Then we prune the original hierarchical tree T by preserving only hypercube-like clusters. As a result, the binary tree T may become a multi-nary tree T_G in which the nodes are all hypercube-like, and we will use T_G as a tree of unlabeled groups for succeeding learning process.

Point-Based Group Soft Label The human assessment of each group is made via a *soft* label, which is an estimate of the proportion of one of the classes in the group. For example, people could say that *90%* of instances in a certain group are *positive*. As the classifier model we want to build is instance-based, it is important to link the individual data instances to group labels. Suppose a group G_i consists of n_i instances $\{(\boldsymbol{x}_{ij}, y_{ij})\}_{j=1}^{n_i}$ and we can express under a probabilistic model that each class variable y_{ij} follows a Bernoulli distribution with parameter μ_{ij}. Then we can estimate the mean parameter of the group from instances observed in the group as $\tilde{\mu}_i = \frac{1}{n_i}\sum_j \mu_{ij}$. This averaged $\tilde{\mu}_i$ for a large sample from the group should converge to the true proportion of classes μ_i in the group. This true proportion is estimated by a human and it is the feedback that we seek to label the group.

Range-Based Group Soft Label One caveat of using the exact point estimate of the class proportions μ_i in the group G_i is that the estimate is often hard for a human to make and it can be subject to various bias and noise. In our work, we relax this requirement by considering a *range-based* soft-label feedback that allows weaker supervision on groups, such as *60% to 80% positive*. Suppose the group G_i is given a range label as $[lb_i, ub_i]$ ($0 \leq lb_i \leq ub_i \leq 1$), how can we interpret it and build it into our learning process? As humans often tend to give a symmetric interval estimation and place most of his/her confidence in the interval, our solution is to assume that the range label defines a symmetric interval and that the span of this interval reflects uncertainty of the annotator in the proportion estimate. To incorporate this idea into the model, we rely on a hierarchical Bayesian approach and assume μ_i is a random variable that follows Beta distribution $Beta(a_i, b_i)$, where a_i, b_i are the two shape parameters. We treat $[lb_i, ub_i]$ as a confidence interval to determine the two hyper-parameters as follows. First, the midpoint of the upper and lower bound of the interval is approximately equal to the mean of the Beta distribution. Then it leads to: $\mathbb{E}(\mu_i) = a_i/(a_i + b_i) = (lb_i + ub_i)/2$. Second, we assume that $1 - \alpha$ of the density of μ_i should fall into $[lb_i, ub_i]$, that is, $I_{ub_i}(a_i, b_i) - I_{lb_i}(a_i, b_i) = 1 - \alpha$, where $I_x(a_i, b_i)$ is the cumulative distribution function (or the regularized incomplete beta function). In this way, the range label of each group G_i can be interpreted as a $Beta(a_i, b_i)$ of μ_i.

3 Active Refinement of Groups

Given the group hierarchy T_G and the initial fringe $F^{(0)} = \{(T_G)'s\ root\}$ at $t = 0$, in each active learning cycle, we adopt *maximum expected model change* criterion [12, 3] to select a group G_* in $F^{(t)}$ to split, query its children and then replace G_* with its children in the fringe, renewed as $F^{(t+\Delta t)}$. The key idea is to select the group G_* which would lead to the greatest change to current model if we *were* to split it. To achieve this goal, we need to: (1) estimate the label distribution of each group's children; and (2) calculate the expected model change properly.

For each group G_i in the current fringe $F^{(t)}$, we model each of its child's soft-label distribution again using a Beta distribution, aka a posterior Beta. Please recall the classic Binomial-Beta model for Bayesian estimation, that the empirical counts from a Binomial likelihood plus a Beta prior yields a posterior Beta. This model precisely applies to our situation where the empirical counts of positives/negatives in a child group can be predicted by current model $\boldsymbol{\theta}^{(t)}$, and the prior Beta comes directly from its parent. One difference is we use *expected* rather than *hard* counts to preserve more precision. Formally, suppose $Beta(a_i, b_i)$ is G_i's label distribution (known), serving as a prior; for each child G_{ij}, the number of expected positive instances is $\hat{n}_{ij}^{+} = \sum_j \mathbb{E}(\mathbf{1}_{\hat{y}_{ij}=1}) = \sum_j \hat{\mu}_{ij}$, where $\hat{\mu}_{ij} = P(y_{ij}|\boldsymbol{x}_{ij}; \boldsymbol{\theta}^{(t)})$. $\hat{n}_{ij}^{-}$ can be calculated similarly. Based on the two statistics, the label distribution of G_{ij} follows a posterior $Beta(a_{ij}, b_{ij})$ where $a_{ij} = a_i + \hat{n}_{ij}^{+}$ and $b_{ij} = b_i + \hat{n}_{ij}^{-}$. After all child groups $\{G_{ij}\}$'s label distribution are estimated, we retrain the model (to be $\boldsymbol{\theta}_{[i]}^{(t)}$) based on $F_{[i]}^{(t)} = (F^{(t)} - G_i) \cup \{G_{ij}\}$, and compare it to $\boldsymbol{\theta}^{(t)}$. So $\boldsymbol{\theta}_{[i]}^{(t)}$ represents the new model parameter that we have *guessed* should be, had the group G_i be split.

To calculate the model change $MC(\boldsymbol{\theta}, \boldsymbol{\theta}')$, similar to [3, 10], we use all instances in $\mathcal{U}$ as a validation set, and calculate the soft label changes for all instances. Formally, $MC(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sum_{\boldsymbol{x}\in\mathcal{U}} |\mu(\boldsymbol{x}; \boldsymbol{\theta}) - \mu(\boldsymbol{x}; \boldsymbol{\theta}')|$, where $\mu(\boldsymbol{x}; \boldsymbol{\theta}) = P(y = 1|\boldsymbol{x}; \boldsymbol{\theta})$.

Finally, we select $G_* = \arg\max_{G_i \in F^{(t)}} MC(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}_{[i]}^{(t)})$ to split. After that, G_*'s children $\{G_{*j}\}$ are sent for querying and a new fringe is updated as $F^{(t+\Delta t)} = (F^{(t)} - G_*) \cup \{G_{*j}\}$ for learning a new model. As each group label consumes one query, Δt is equal to the number of G_*'s children.

4 Learning a Model from Labeled Groups

The Loss Function Suppose at time t, there are N labeled groups $\{G_1, ..., G_N\}$ in fringe $F^{(t)}$ and let $\{\boldsymbol{x}_{i1}, ..., \boldsymbol{x}_{in_i}\}$ be a set of n_i instances covered by a group G_i. Our goal is to learn an instance probabilistic model $P(y|\boldsymbol{x}; \boldsymbol{\theta})$. Here $\boldsymbol{x} \in \mathbb{R}^m$ and $y \in \{0, 1\}$.

Let us first consider the case where each group G_i is given an *exact* label $\mu_i \in [0, 1]$ by an annotator. Inspired by previous work [7, 5, 6], this estimate μ_i is an approximation to the empirical average of all the instance μ_{ij}s in that group. Therefore, we formulate the learning problem as a least square regression that tries to minimize the squared error over each group label μ_i and the empirical average of all μ_{ij}s given by the model. That is, the error term for group G_i is $(\mu_i - \frac{1}{n_i}\sum_j \mu_{ij})^2$, where μ_i is given by the annotators and $\mu_{ij} = P(y_{ij} = 1|\boldsymbol{x}_{ij}; \boldsymbol{\theta})$ is given by the model. Further after weighting each term by group size n_i, the overall loss function is defined as:

$$L(\boldsymbol{\theta}) = \sum_i \{\frac{n_i}{n}(\frac{\sum_j \mu_{ij}}{n_i} - \mu_i)\}^2 + \lambda R(\boldsymbol{\theta})$$

Here $n = \sum_i n_i = |\mathcal{U}|$ and $R(\boldsymbol{\theta})$ is the regularization term weighted by a constant $\lambda > 0$.

However, providing an exact μ_i may be hard for a human. Instead we query a range label which has been interpreted as a Beta distribution $Beta(a_i, b_i)$ in Section (2). Then we take the expectation of u_i on the loss function to integrate out μ_i. Therefore the final loss function can be modified as:

$$\begin{aligned} L(\boldsymbol{\theta}) &= \sum_i \underline{\mathbb{E}_{\mu_i}}\{\frac{n_i}{n}(\frac{\sum_j \mu_{ij}}{n_i} - \mu_i)\}^2 + \lambda R(\boldsymbol{\theta}) \\ &= \frac{1}{n^2}\sum_i \{(\sum_j \mu_{ij})^2 - 2n_i \sum_j \mu_{ij}\underline{\mathbb{E}(\mu_i)} + n_i^2 \underline{\mathbb{E}(\mu_i^2)}\} + \lambda R(\boldsymbol{\theta}) \end{aligned}$$

As the above equation indicates, the main difference of using range labels is to use the first and second moments of μ_i to approximate the exact μ_i. $\mathbb{E}(\mu_i)$ preserves the major information of the point μ_i, while $\mathbb{E}(\mu_i^2)$ permits more uncertainty of μ_i.

Learning We want to find the best parameters $\boldsymbol{\theta}^*$ that minimize $L(\boldsymbol{\theta})$. This loss function can generalize to any classifier that outputs instance-level probabilities used with differentiable objective functions. If the second order derivative is readily available, one can apply Newton's optimization to minimize the loss. For more complicated models, one can learn parameters using a gradient-based optimization method, such as BFGS [4].

5 Experiments

We conduct an empirical study to evaluate our proposed approach on 3 real binary classification data sets collected from UCI machine learning repository [1]. Please see Table 1. *Wine* has been used widely in previous work [9, 14]; *Music* is high-dimensional and thus instance-based labeling can be time-consuming; *Seismic* has an unbalanced class distribution, and instance-based labeling may not cover the minor class information properly.

Methods Tested We compare our approach (LSLG) to four related methods: (1) Density-Weighted Uncertainty Sampling (DWUS) [11] which combines uncertainty sampling and data density on deciding which instance to be queried next; (2) RIQY, is a state-of-the-art group-based active learning approach that also queries groups. (But they did not deal with group learning problem, as they directly propagate group labels to instances.) (3) Hierarchical Sampling (HS) [2] which uses a hierarchical tree to guide instance sampling; (4) Multi-Instance Active Learning (MIAL), which queries instances in positive bags in Multiple-Instance Learning framework.

Soft Group Label Simulation For group queries required by our approach, we simulate the human feedback as RIQY did, by counting the class proportion based on instance labels within each group's description region represented by conjunctive pattern. To simulate a variety of range labels in our framework, we experiment with four separate levels of precisions: 5%, 10%, 20% and 30%, where each one is a fixed width for all the range labels in that experiment. For example, "LSLG(10%)" means that we run our method with all the group range labels simulated at a fixed width=10% (e.g.

Table 1: 3 UCI data sets

Name	Description	# Data	# Features	Positive Class	Feature Type*
Seismic	Seismic bumps	2584	18	7%	Num-Ord-Cat
Music	Music origin	1059	68	53%	Num
Wine	Wine quality	4898	11	67%	Num

*'Num', 'Ord' and 'Cat' stand for Numerical, Ordinal and Categorical respectively.

60%∼70% positive). To work out the exact range label provided for each group query, we perform the following steps: (1) find all instances in $\mathcal{U}$ that fall into the group's description (i.e. a hypercube); (2) count and calculate the positive portion among the included instances, marked as p; (3) Add a uniform noise to p: $p_\epsilon = p + \epsilon$ where ϵ follows $Uniform(-w/2, w/2)$, and w is the fixed range label width; (4) A candidate range label appears as $[p_\epsilon - w/2, p_\epsilon + w/2]$; (5) Fix the candidate label to be within [0,1] by truncating the out portion.

Model Setting We employ Logistic Regression as the base model for all methods. Also there are two hyper-parameters in our learning phase. The first is α that is used to interpret a range label into a Beta distribution; and the other is λ that serves as the penalty in regularization. We have experimented many times by cross-validation and find out our model is not sensitive to their choices. α can be picked $\in [0.01, 0.1]$ and $\lambda \in [10^{-5}, 10^{-2}]$. By default, we set $\alpha = 0.05$ and $\lambda = 0.001$.

Evaluation Metrics We adopt Area Under the Receiver Operating Characteristic curve (AUC) to evaluate the generalized classification quality of Logistic Regression on the test data. Our graphs will plot the AUC scores iteratively after each $k \leq 200$ queries are posed.

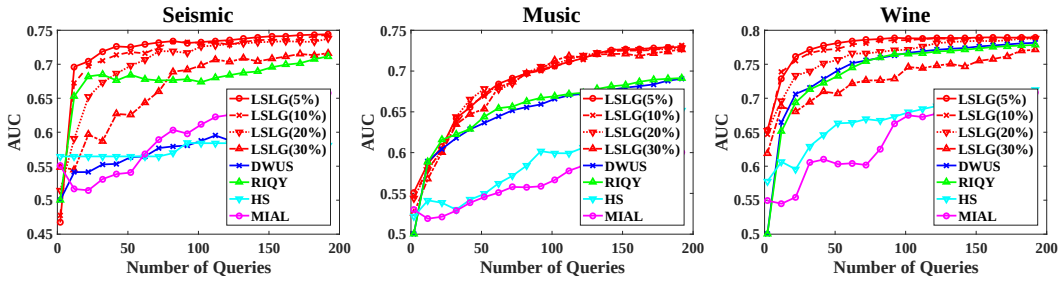


Figure 1: Performances of different methods on 3 UCI data sets.

Experiment Results The results are shown in Figure 1. Overall, when range labels are fairly precise (i.e. range width $\leq$20%), our LSLG (in red lines) outperforms other methods. There are two primary strengths: firstly, our group queries are more informative than the same number of instances queries and our group learning algorithm benefits from the richer class information provided by groups. Secondly, our active refinement of groups can select the group that can potentially lead to maximum model change and thus accelerates the whole learning process.

Effects of Range Label Precision We have experimented our framework with four levels of precisions. According to the results (shown in red lines with different styles) , we observe that for all datasets, the performance of our method gradually drops as the range label uncertainty increases from 5% to 30%, as expected. We recommend a robust range width as 20%, based on our results.

6 Summary

We propose a group-based active learning framework that can efficiently learn instance-based classifiers from soft-labeled groups. We have dealt with the problems of group generation, representation and label collection when labeled groups are not readily available. Therefore, our framework is able to generalize to conventional binary classification tasks and is best suited when providing group labels is more feasible and less costly than instance labeling. In future work, real user studies are necessary to further evaluate the feasibility and efficiency of our proposed approach.

Acknowledgement

The work presented in this paper was supported by NIH grants R01GM088224 and R01LM010019. The content of the paper is solely the responsibility of the authors and does not necessarily represent the official views of NIH.

References

[1] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[2] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, pages 208–215. ACM, 2008.

[3] Alexander Freytag, Erik Rodner, and Joachim Denzler. Selecting influential examples: Active learning with expected model output changes. In *European Conference on Computer Vision*, pages 562–577. Springer, 2014.

[4] Geof H Givens and Jennifer A Hoeting. *Computational statistics*, volume 710. John Wiley & Sons, 2012.

[5] Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM, 2015.

[6] Hendrik Kück and Nando de Freitas. Learning about individuals from group statistics. *CoRR*, abs/1207.1393, 2012.

[7] Novi Quadrianto, Alex J Smola, Tiberio S Caetano, and Quoc V Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, 10(Oct):2349–2374, 2009.

[8] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[9] Parisa Rashidi and Diane J Cook. Ask me better questions: active learning queries based on rule induction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 904–912. ACM, 2011.

[10] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, pages 441–448, 2001.

[11] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[12] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Advances in neural information processing systems*, pages 1289–1296, 2008.

[13] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[14] Yanbing Xue and Milos Hauskrecht. Active learning of classification models with likert-scale feedback. In *SIAM Data Mining Conference, 2017*. SIAM, 2017.