Unsupervised Data Augmentation
(2019)

Qizhe Xie et al
Resume

May 7, 2019

1 Introduction

The authors propose to apply data augmentation to unsupervised data in a semi-supervised setting, the method named Unsupervised Data Augmentation (UDA), encourages the model predictions to be consistent between an unlabeled example and an augmented unlabeled example. Unlike previous methods that use random noise such as Gaussian noise or dropout noise, UDA makes use of harder and more realistic noise generated by state-of-the-art data augmentation method.

The most recent promising methods in semi supervised setting are: 1- using graph convolution for semi supervised learning for graph based propagation, 2- modeling prediction target as latent variables, 3- consistency / smoothness enforcing that regularize the models prediction to be less sensitive to small perturbations applied to examples (labeled or unlabeled). Given an observed example, smoothness enforcing methods first create a perturbed version of it (e.g., adding Gaussian noise or dropout), and enforce the model predictions on the two examples to be similar given that a good model should be invariant to any small perturbations that do not change the nature of an example, the methods in this category differ mostly in the perturbation function,

The proposed method shows that better augmentation methods lead to greater improvements and that they can be used on many other domains, UDA minimizes the KL divergence between model predictions on the original example and an example generated by data augmentation, UDA is applied on unsupervised data which is available at larger quantities and therefore has the potential to work much better than standard supervised data augmentation.

The main contributions of the paper are:

- A training technique called TSA that effectively prevents overfitting when much more unsupervised data is available than supervised data.
- Showing that targeted data augmentation methods (such as AutoAugment) give a significant improvements over other untargeted augmentations.
- Combine a set of data augmentations for NLPs, and show that the method works well there.
- Showing significant leaps in performance compared to previous methods in a range of vision and language tasks.
- Proposing a method so that UDA can be applied even if the class distributions of labeled and unlabeled data mismatch.

2 Unsupervised Data Augmentation (UDA)

2.1 Supervised Data Augmentation

Data augmentation aims at creating novel and realistic-looking training data by applying a transformation to the input of an example, without changing the label / nature of the example.

Formally, let $q(\hat{x}|x)$ be the augmentation transformation from which one can draw augmented examples $\hat{x}$ based on an original example x. For an augmentation transformation to be valid, it is required that any example $\hat{x} \sim q(\hat{x}|x)$ drawn from the distribution shares the same ground-truth label as x, $y(\hat{x}) = y(x)$. Given a valid augmentation transformation, we can simply use the following objective to minimize the negative log-likelihood on augmented examples:

$$\min_{\theta} \mathcal{J}_{\text{da}}(\theta) = \mathop{\mathbb{E}}_{x,y^* \in L \hat{x} \sim q(\hat{x}|x)} \left[-\log p_\theta \left(y^* | \hat{x} \right) \right]$$

So we construct an augmented set from the original set to provide additionnal supervised training signal.

2.2 Unsupervised Data Augmentation Methods

A recent line of work in semi-supervised learning has been utilizing unlabeled examples to enforce smoothness of the model. The general form of these works can be summarized as follows:

- Given an input x, compute the output distribution $p_\theta(y|x)$ given x and a perturbed version $p_\theta(y|x, \epsilon)$ by injecting a small noise ϵ. The noise can be applied to x or hidden states or be used to change the computation process.
- Minimize some divergence between the two predicted distributions $\mathcal{D}\left(p_\theta(y|x) \| p_\theta(y|x, \epsilon)\right)$.

This procedure enforces the model to be insensitive to the perturbation ϵ and hence smoother with respect to changes in the input (or hidden) space.

UDA uses state-of-the-art data augmentation (AutoAUgment) targeted at different tasks as a particular form of perturbation and optimize the same smoothness enforcing objective on unlabeled example:

$$\min_{\theta} \mathcal{J}_{\text{UDA}}(\theta) = \mathop{\mathbb{E}}_{x \in U} \mathop{\mathbb{E}}_{\hat{x} \sim q(\hat{x}|x)} \left[\mathcal{D}_{\text{KL}} \left(p_{\tilde{\theta}}(y|x) \| p_\theta(y|\hat{x}) \right) \right]$$

Where $q(\hat{x}|x)$ is a data augmentation transformation and $\tilde{\theta}$ is a fixed copy of the current parameters θ indicating that the gradient is not propagated through $\tilde{\theta}$ as suggested. The data augmentation transformation used here is the same as the augmentations used in the supervised data augmentation such as back translation for texts and random cropping for images.

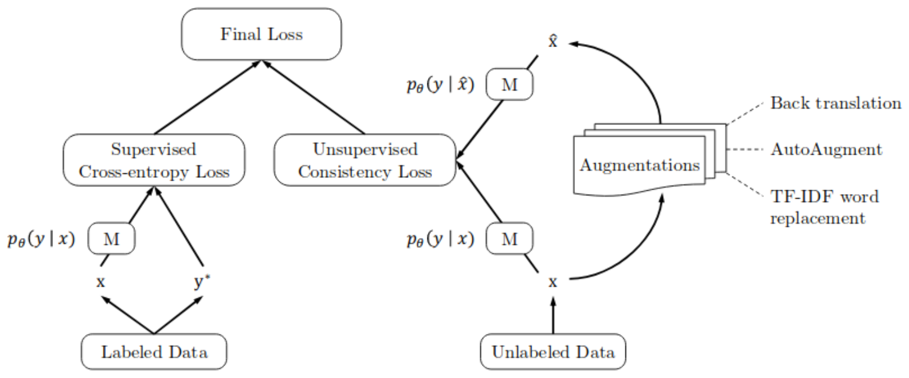


Figure 1: Training objective for UDA, where M is a model that predicts distribution $p_\theta(y \mid x)$ given x, and y^* is the ground-truth label.

Using targeted data augmentation as the perturbation function has several advantages:

- Valid / realistic perturbations: if the perturbation is generated byadding a large Gaussian noise to the input example, the input image might become indiscernible orthe correct label of the augmented example might be different from that of the original exampl
- Diverse perturbations: Data augmentation can generate a diverse set of samples since it can makelarge modifications to the input example without changing its label, while the perturbations such asGaussian or Bernoulli noise only make local changes to the input example
- Targeted inductive biases: As shown in AutoAugment, data augmentation policy can be directly optimized towards improving validation performance on each task. Such performance-oriented augmentation policy can learn to figure out the missing or most wantedtraining signal / inductive biases in an original labeled set

2.3 Training Signal Annealing

Given that is it much easier to obtain labeld data than labeled data, and to take adavantage of as much unlabled data as possible, we need a large model, but it can overfit on the supervised data very easily, to avoid this probel TSA gradually decreased the number of supervised examples and the model is trained on more and more unsupervised examples, for each training step t, we set a threshold $\eta_t \leq 1$. When the probability of the correct category $p_\theta\left(y^*|x\right)$ of a labeled example is higher than the threshold η_t, we remove this example from the loss function and only trainon other labeled examples in the minibatch. Formally, given a minibatch of labeled examples B, we replace the supervised objective with the following objective:

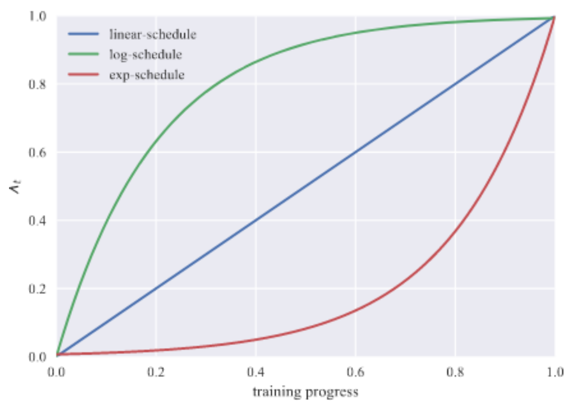
$$\min_\theta \frac{1}{Z} \sum_{x,y \in B} \left[-\log p_\theta(y|x) I\left(p_\theta(y|x) < \eta_t\right)\right]$$

where $I(.)$ is the indicator function and $Z = \sum_{x,y\in B} I\left(p_\theta(y|x) < \eta_t\right)$ is simply a re-normalization factor. Effectively, the threshold η_t serves as a ceiling to prevent the model from over-training on examples that the model are already confident about.

Hence, when we gradually anneal η_t from $\frac{1}{k}$ to 1 during training with k being the number of categories, the model can only slowly receive supervisions from the labeled examples, largely alleviating the overfitting problem.

There is three types of signal annealing schedulers of λ_t given that $\eta_t = \frac{1}{k} + \lambda_t * \left(1 - \frac{1}{k}\right)$, these are:

- log-scheduler (η_t increases rapidly at the start): $\eta_t = \left(1 - \exp\left(-\frac{t}{T} * 5\right)\right) * \left(1 - \frac{1}{k}\right) + \frac{1}{k}$
- linear-schedule (η_t increases linearly): $\eta_t = \frac{t}{T} * \left(1 - \frac{1}{k}\right) + \frac{1}{k}$
- exp-schedule (where η_t increase rapidly at the end of training): $\eta_t = \exp\left(\left(\frac{t}{T} - 1\right) * 5\right) * \left(1 - \frac{1}{k}\right) + \frac{1}{k}$



2.4 Additional training techniques

Using out-of-domain unlabeled data Ideally, we would like to make use of out-of-domain unlabeled data since it is usually much easier to collect a large amount of out-of-domain unlabeled data, but the class distributions of out-of-domain data are usually mismatched with those of in-domain data, due to the mismatched class distributions, using out-of-domain unlabeled data can hurt theperformance than not using it, one solution is the train the model on labled data and run it on the unlabled data and only take the examples where the model is confident.

Flat distributions With a large number of categories (like imagenet) and a limited number of training examples, the predicted distributions on unlabeled and augmented unlabeled data ($p_{\tilde{\theta}}(y|x)$ and $p_\theta(y|\hat{x})$) tend to be over-flat across categories, so the unsupervised traning signal from the KL divergence is relatively weak and get dominated by the supervised par, to sharpen the signal:

- Entropy minimization: adding an entropy term to the overall objective and regularize the predicted distribution on augmented examples $p_\theta(y|\hat{x})$ to have a low entropy,
- Softmax temperature control: $p_{\tilde{\theta}}(y|x)$ is computed as $\text{Softmax}(l(x)/\tau)$ where $l(x)$ denotes the logits and τ is the temperature. A lower temperature corresponds to a sharper distribution.
- Confidence-based masking: ask out examples that the current model is not confident about. Specifically, in each minibatch, the unsupervised loss term is computed only on examples whose highest probability $\max_y p_{\tilde{\theta}}(y|x)$ is greater than a threshold.

3 Experiments

Fully supervised baseline

Datasets (# Sup examples)	IMDb (25k)	Yelp-2 (560k)	Yelp-5 (650k)	Amazon-2 (3.6m)	Amazon-5 (3m)	DBpedia (560k)
Pre-BERT SOTA	*4.32*	2.16	29.98	3.32	34.81	0.70
$\text{BERT}_{\text{LARGE}}$	4.51	*1.89*	*29.32*	*2.63*	*34.17*	*0.64*

Semi-supervised setting

Initialization	**UDA**	IMDb (20)	Yelp-2 (20)	Yelp-5 (2.5k)	Amazon-2 (20)	Amazon-5 (2.5k)	DBpedia (140)
Random	✗	43.27	40.25	50.80	45.39	55.70	41.14
	✓	25.23	8.33	41.35	16.16	44.19	7.24
$\text{BERT}_{\text{BASE}}$	✗	27.56	13.60	41.00	26.75	44.09	2.58
	✓	5.45	2.61	33.80	3.96	38.40	1.33
$\text{BERT}_{\text{LARGE}}$	✗	11.72	10.55	38.90	15.54	42.30	1.68
	✓	4.78	2.50	33.54	3.93	37.80	1.09
$\text{BERT}_{\text{FINETUNE}}$	✗	6.50	2.94	32.39	12.17	37.32	-
	✓	**4.20**	**2.05**	**32.08**	**3.50**	**37.12**	-

Table 1: Error rates on text classification datasets. $\text{BERT}_{\text{FINETUNE}}$ denotes $\text{BERT}_{\text{LARGE}}$ fine-tuned on in-domain unlabeled data. We do not pursue further experiments for $\text{BERT}_{\text{FINETUNE}}$ on DBPedia since fine-tuning BERT on DBPedia does not result in better performance than $\text{BERT}_{\text{LARGE}}$ in our preliminary experiments. This is probably due to the fact that DBPedia is on the Wikipedia domain and BERT is already trained on the whole Wikipedia corpus. In the fully supervised settings, the pre-BERT SOTAs include ULMFiT [Howard and Ruder, 2018] for Yelp-2 and Yelp-5, DPCNN [Johnson and Zhang, 2017] for Amazon-2 and Amazon-5, Mixed VAT [Sachan et al., 2018] for IMDb and DBPedia.

Datasets	CIFAR-10 (4k)	SVHN (1k)
Supervised	20.26 ± .38	12.83 ± .47
AutoAugment*	14.1	8.2
Pseudo-Label	17.78 ± .57	7.62 ± .29
Π-Model	16.37 ± .63	7.19 ± .27
Mean Teacher	15.87 ± .28	5.65 ± .47
VAT	13.86 ± .27	5.63 ± .20
VAT + EntMin	13.13 ± .39	5.35 ± .19
LGA + VAT	12.06 ± .19	6.58 ± .36
mixmixup	10	-
ICT	7.66 ± .17	3.53 ± .07
UDA	**5.27 ± .11**	**2.46 ± .17**

Table 2: Comparison with existing methods on CIFAR-10 and SVHN with $4,000$ and $1,000$ examples respectively. All compared methods use a common architecture WRN-28-2 with 1.4M parameters except AutoAugment* which uses a larger architecture WRN-28-10. The results for Pseudo-Label [Lee, 2013], Π-Model [Laine and Aila, 2016], Mean Teacher [Tarvainen and Valpola, 2017] and VAT (VAT) [Miyato et al., 2018] are reproduced by Oliver et al. [2018]. We also compare with recently proposed models ICT [Verma et al., 2019], LGA+VAT [Jackson and Schulman, 2019] and mixmixup [Hataya and Nakayama, 2019].