Implementation of Unsupervised Domain Adaptation by Backpropagation

Chih-Hui Ho Yuan Qi Xingyu Gu
Department of Computer and Science Engineering
University of California San Diego
chh279@eng.ucsd.edu yuq083@eng.ucsd.edu xig090@eng.ucsd.edu

Abstract

Deep learning models have been shown to be impressively powerful in tasks such as image classification and object recognition given massive amounts of labeled data. However, it is still an open challenge on the domain where only few labeled data are available. Inspired by the previous work Unsupervised Domain Adaptation by Backpropagation [7], aiming at minimizing the gap between deep features from target domain and source domain, we decide to implement the content mentioned in this paper. The project will contains experiments evaluating the performance of the classifiers trained with the proposed architecture in [7]. Several domain adaptation tasks will be conducted using dataset including MNIST [16], MNIST-M, Synthetic numbers, SVHN [22], Synthetic Signs, GTSRB [28], Office. Moreover, we will extend the paper in the follow three directions (i) conducting experiments on dataset that are not listed in [7] (ii) examining and comparing the adaptation difficulties between different datasets.(iii) investigating the possibility of solving the domain adaptation problem with generative adversarial network [11, 1, 2]. All the code can be found on `https://github.com/neoqy/DDA`

1. Introduction

During the last few years, the large convolutional neural networks (CNNs), such as Alexnet [15], VGG [27], GoogleNet [29], ResNet [12], etc. achieve previously unmatched image classification performance. The successfulness of these deep models are established on the large scale labeled image dataset ImageNet [5].

For most of the applications which use deep learning models as basic architecture, the models are pretrained on ImageNet and transferred to the specific task by fine-tuning. However, fine-tunning is only useful when the labeled data for the specific task is available to the developer. For the cases where only limited amount of labeled data or no labeled data are available, insufficiently-tunned deep model will fail in solving the interested task. Such scenarios are quite common in the real world application. For example, medical images are really rare and hard to collect, which makes fine-tunning techniques no longer useful in medical images classification task.

Although lacking of large-scale labeled dataset for the targeted task, there are lots of relevant datasets that might be useful for the targeted task. Take multiview image classification task for example. There is no existing large-scale labeled dataset containing multiview of the real images, but the synthetic graphical 3D image dataset, such as ModelNet [33] and ShapeNet [3], are available to the public. As a result, the question becomes whether it is possible to utilize those labeled data from different domains and apply the trained classifier to the target task.

In this situation, domain adaptation often provides an attractive solution, given that labeled data of similar nature but from a different domain (e.g. synthetic images) are available. Among all the related work of domain adaptation, the paper Unsupervised Domain Adaptation by Backpropagation [7] is one of the earliest papers in this field, which is cited by lots of papers. In this project, we want to implement the proposed solution in this paper, and test it by the experiments talked in this paper. The implemented algorithm is tested on adaptation between several datasets. In addition, we modify the domain adaptation loss similar to GAN, as described in section 3.3 and follow the training scheme in [11]. We experiment a new training methods to gap the difference between domains.

2. Related work

There has been a great amount of research in the field of domain adaptation and transfer learning in the past few decades. Most of the previous work is established on the shallow learning methods, including [26, 23, 8]. The previous shallow learning domain adaptation methods aim to solve the domain shift between the source and target domains and those methods can be mainly separated into two categories. The first category learns the shared feature space between target and source domain [23, 8], while the other category trains the classifier on the weighted source

data [26].

As the emergence of deep learning, many deep features are extracted using deep learning methods. The deep features are the high-level abstract representation of the input and is shown to effectively encode the important information of the input. An emerging problem in deep learning is that it is prone to be overfitting to the training data or source domain and such a problem restricts the generalization of the trained classifier. As a result, recent works have intensively addressed this problem by combining the domain adaptation task with deep learning.

The literature of domain adaptation can also be separated into two categories. The first category uses part of the target data, either unsupervised or semi-supervised, to bridge the discrepancy between source and target domain. Previous works include [13, 19, 18, 24, 32]. The second category uses a discriminator to distinguish between domains, so that the feature extractor can learn a domain invariant feature regardless the input domain to fool the domain discriminator. [17, 14, 30, 31] and the paper *Unsupervised Domain Adaptation by Backpropagation* that we are implementing falls into this category.

The nature of Deep Domain Adaptation (DDA) [7] is a minimax game, very like that of a Generative Adversarial Network (GAN). However, traditional GAN [10] is very hard to train because the loss cannot indicate the network's performance. Furthermore, the training of generator and discriminator has to be balanced well, which is just like what we do to the label classifier and domain classifier in DDA and is very tricky. However, Wasserstein GAN [1] fixed the problem by using Wasserstein distance as the optimization goal instead of KL divergence or JS divergence. This makes the loss able to indicate training process and makes the discriminator can be trained to converge. Taken into account Wasserstein distance's success in both theory and practice, and its similar nature to DDA, it is highly possible that it can be applied to DDA to improve the performance.

3. Deep Domain Adaptation

3.1. Notation

Domain adaptation task aims at bridging the gap between source and target domains. The input x from source domain S is denoted as x_s, while the input x from target domain T is denoted as x_t. The ground truth label for source data x_s is denoted as y_s and ground truth label for target data is not provided.

Identical to [7], we define label classifier G_y has parameter θ_y, domain classifier G_d has parameter θ_d and feature extractor G_f has parameter θ_f. The loss function is defined as

$$E\left(\theta_f, \theta_y, \theta_d\right) = \sum_{\substack{i=1\ldots N \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=i\ldots N} L_d^i(\theta_f, \theta_d), \tag{1}$$

where $L_y(\cdot,\cdot)$ and $L_d(\cdot,\cdot)$ are the loss functions for label and domain classifications respectively, and λ is an adaptation factor that controls the ratio of the two loss values.

3.2. Baseline Architecture

We follow the proposed architecture in [7], as shown in Figure 1. The architecture can be separated into three parts.

1. **Feature extractor** extracts the feature from the input, regardless of the domain of the input.
2. **Domain classifier** aims at classify which domain the extracted feature comes from.
3. **Label classifier** classifies the extracted feature from source domain, where the ground truth is given as y_s

The feature extraction part of the network is typical convolutional network, while the classification part of the network involves a specially designed layer, named gradient reversal layer (GRL). This layer will multiply the gradient with a negative scalar and pass the reversed gradient to the previous layer. Its forward- and backpropagation behaviors are $R_\lambda\left(\mathbf{x}\right) = \mathbf{x}$ and $\frac{dR_\lambda}{d\mathbf{x}} = -\lambda\mathbf{I}$. More details can be found the original paper[7].

3.3. GAN architecture

Given the inspiration from [11], we modified the proposed method with different loss function similar to generative adversarial network. Following the notation define in 3.1, the loss function L is reformulated as

$$log(G_d(G_f(x_t))) + log(1 - G_d(G_f(x_s))) + G_y(G_f(x_s), y_s) \tag{2}$$

$$\underset{\theta_d}{\operatorname{argmax}} \underset{\theta_f,\theta_y}{\operatorname{argmin}} L \tag{3}$$

,where G_d will output 1 if the extracted feature is from target, otherwise it will output 0. By designing the new architecture, we examine the possibility to further extend the original paper.

3.4. WGAN architecture

It is known that Wasserstein GAN [1] is more stable and easier to converge during the training process. We redesign equation 2 as follow by modifying the loss function L similar to WGAN.

$$G_d(G_f(x_t)) - G_d(G_f(x_s)) + G_y(G_f(x_s), y_s) \tag{4}$$

,where G_d will output 1 if the extracted feature is from target, otherwise it will output 0.

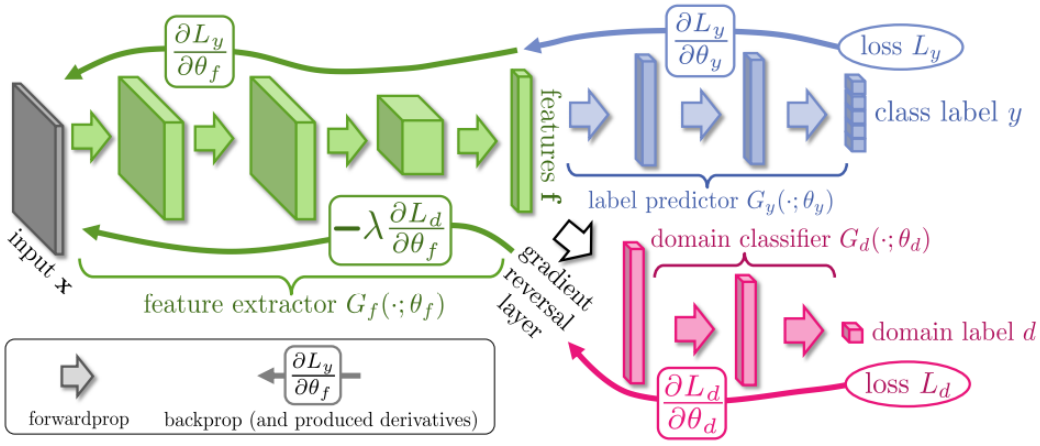


Figure 1. Proposed architecture

4. Experiments

To test the proposed approach, we carry out five experiments of domain adaptation between different datasets: MNIST [16], MNIST-M [7], Synthetic numbers [7], SVHN [22], Synthetic Signs[21], GTSRB [28], Office Dataset [25].

Table 1. Source and target domain for each experiment

Experiment	Source Domain	Target Domain
1	MNIST	MNIST-M
2	SYN NUMBERS	SVHN
3	SVHN	MNIST
4	SYN SIGNS	GTSRB
5	AMAZON	WEBCAM
5	DSLR	WEBCAM
5	WEBCAM	DSLR

We choose PyTorch to implement our approach, because it is of great flexibility and can be extended easily with custom deep learning network functions and layers.

Baselines. For each of the experiments, we present three baselines.

First, we train the model without the domain classifier on source domain dataset, and test it on target domain dataset. This **source only** baseline provides a lower bound of performance, since the architecture of the model is the same as the proposed approach, and no domain adaptation is done.

Second, we both train and test the model without the domain classifier on target domain dataset. This **train on target** baseline provides an upper bound of performance. Because all the information about the labels of target domain is utilized in this baseline, any domain adaptation method of the same architecture without utilizing target domain's label information cannot has better performance than it.

Third, our approach is compared with the subspace alignment (**SA**) [6] baseline, which is referred to as a "shallow" DA method. We directly use its performance reported in [7], in which SA's most important hyper-parameter (the number of principal components) is chosen in a grid search manner from range $\{2, \ldots, 60\}$ to ensure its test performance.

In the experiment on the office dataset, we compare the performance of our approach with the source only baseline and the results reported in other published papers.

Network architectures. For different datasets, we use different network architectures, i.e. the architectures of the feature extractor, label classifier and domain classifier. The loss functions of label classifier and domain classifier are cross entropy loss and binary cross entropy loss respectively.

The networks for all experiments except office dataset are trained from scratch. For office dataset, we use pre-trained AlexNet [15] provided by PyTorch package as a start point for the feature extractor, then train it together with our custom label classifier and domain classifier. The specific architecture for each experiment is shown in Section 5.

Network training procedure. We preprocess the input image by resizing them to fixed size for each architecture. The original range of images' pixel value is $[0, 1]$, we normalize it to $[-1, 1]$ by subtracting mean of 0.5 and dividing by std of 0.5.

We use SGD with momentum to train our net work on mini-batches. Each mini-batch consists of images half from source domain and half from target domain. But for office dataset, we set the ratio of images from source and target dataset according to the sizes of the datasets, because the sizes of the three datasets are highly imbalanced.

The momentum of the tth epoch is defined as $v_t = \eta \cdot v_{t-1} - \mu \cdot dx$, where μ is learning rate, dx is the gradient and η is a hyper-parameter set to be 0.9 in all our experiments.

We also use learning rate annealing, the learning rate is adjusted by

$$\mu_p = \frac{\mu_0}{(1 + \alpha \cdot p)^\beta}, \tag{5}$$

where p is the training process in range $[0, 1]$, μ_0 is the initial learning rate, α and β are parameters that control how the learning rate decreases. In all our experiments, we set $\mu_0 = 0.01$, $\alpha = 10$ and $\beta = 0.75$.

To avoid the noise of the domain classifier at the beginning of the training, we adjust the λ in Equation 1 using

$$\lambda_p = \frac{2}{1 + \exp(-\gamma \cdot p)} - 1, \tag{6}$$

where γ is set to be 10 in all our experiments. This makes λ change from 0 to 1 gradually according to the training process p.

We initialize the weights of convolutional layer using a Gaussian distribution with 0 mean and standard deviation sqrt $(2/n)$, where n is $kernel\_size^2 \times kernel\_number$. And the fully connected layer's weights are initialized using a Gaussian distribution with 0 mean and standard deviation sqrt $(2/(fan\_in + fan\_out))$, where fan_in and fan_out are respectively the input and output connection numbers of the layer. However, in the office dataset experiment, the feature extractor and part of classifier (`fc6` and `fc7`) of AlexNet uses pretrained parameters as initialization.

Visualization. We use t-SNE [20] to visualize high-dimensional data, e.g. the extracted feature vector, the hidden layer's activations, etc. We noticed that when the distributions of source and target domains are similar in the visualization, the domain adaptation's performance is better.

Choosing hyper-parameters The hyper-parameters are mainly the learning rate μ and adaptation factor λ in our model.

We choose the learning rate μ by observing the loss - if the loss decreases too slow, we increase the learning rate; if the loss diverges eventually, we decrease the learning rate.

The adaptation factor λ should be chosen in an unsupervised way - without referring to labeled data in the target domain. Since that the domain classifier error reflects how close the features from two domains are, and the test error on the source domain reflects how good the label classifier is and how good the features are for classification, we can adjust λ by observing them.

5. Discussion

5.1. MNIST to MNIST-M

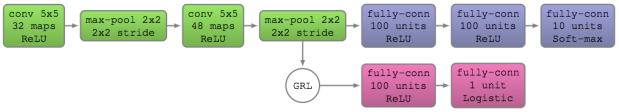


Figure 2. MNIST to MNIST-M architecture

Our first experiment uses the MNIST dataset (LeCun et al., 1998) as source domain, and instead of generating the dataset MNIST-M, we used the one generated in [7]. The experiment we conduct successfully use Domain Adaptation to improve the validating rate in the dataset of MNIST-M. And the detailed result is shown in the Table 2, and Table 3 compares the feature distribution before and after Domain Adaptation.

5.2. SYN Numbers to SVHN

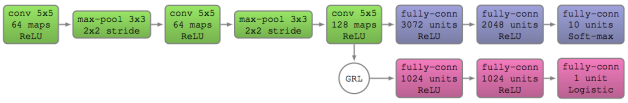


Figure 3. SVHN to MNIST architecture

This experiment uses the dataset from Street-View House Number dataset SVHN (Netzeret al., 2011) as the target domain, and Synthetic Numbers as the source domain. The dataset of Synthetic Numbers we use is also from [7]. From [7] we learn about the fact that the degrees of variation in Synthetic Numbers were chosen manually to simulate SVHN. The architecture in the paper is replicated, as shown in Figure 3. The detailed result of this experiment is also shown in Table 2, and Table 2 compares the feature distribution before and after Domain Adaptation. Our result is different from the one achieve in [7]: For the source-only case, we achieve a higher validating rate: 88.0%, while in the domain adaptation case, we achieve a lower validating rate: 89.4%.

5.3. SVHN to MNIST

This experiment uses the dataset from SVHN [22] as source domain and MNIST as target domain. The architecture in the paper is also shown in Figure 3. According to Table 2, the accuracy of testing on target data with classifier trained on source data only is 0.604, while the accuracy of classifier trained on target data only is 0.992. The proposed method achieves 0.698, which is within the upper and lower bound.

Table 2. Accuracy of different training methods

Method / Source / Target	MNIST / MNIST-M	SYN NUM / SVHN	SVHN / MNIST	SYN SIGN / GTSRB
Source Only (lower bound)	0.551	0.880	0.673	0.557
SA [6]	0.569	0.864	0.593	0.817
Proposed method	0.827	0.894	0.719	—
Train on Target (upper bound)	0.963	0.927	0.992	0.5789

Table 3. Visualization of **MNIST-M** feature for domain adaptation from **MNIST** dataset. It can be observed that when the feature from the output of G_f is mixed, the domain classifier cannot distinguish between source and target domain. It also indicates that after domain adaptation, source and target's distributions of last hidden layer of G_y become similar.

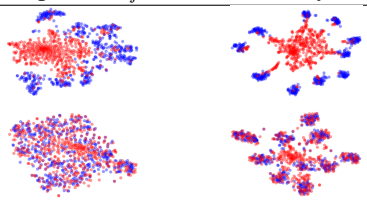
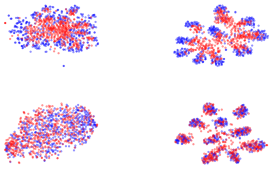
Method/feature	output of G_f	last hidden layer of G_y
Source Only		
DA		

Table 4. Visualization of **SYN** feature for domain adaptation from **SVHN** dataset.

Method/feature	output of G_f	output of G_y
Source Only		
DA		

5.4. SYN SIGN to GTSRB

This experiment uses the dataset from synthetic [1] and German Traffic Sign Benchmarks [28]. There are 43 classes in both dataset. While SYN SIGN dataset have relatively equal number of training data per class, GTSRB has imbalanced number of training sample per class. Assume the minimum training sample of class is N in GTSRB. N random samples per class are selected from the original GTSRB training set, where $N = 210$ in our case.

The proposed architecture, as shown in Figure 4, is implemented. Since the input size of the network is not specified in paper, we set the input image size as 40x40.

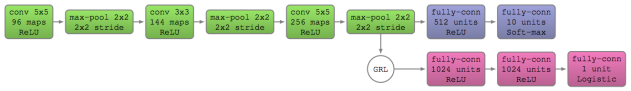


Figure 4. SYN SIGN to GTSRB architecture. Instead of 10 output, the network should have 43 output

[1] https://graphics.cs.msu.ru/en/research/projects/imagerecognition/trafficsign

With our setting, we fail to replicate the experiment in the paper. The accuracy of testing on target domain with classifier trained on source data only is 0.557, while the accuracy of the classifier trained on target data only is 0.5789. It is interesting that the latter is only slightly higher than the former.

5.5. Office dataset

For the three domain adaptation tasks in this joint dataset, we use pretrained AlexNet [15] provided by PyTorch package as a start point for the feature extractor and the common part (`fc6` and `fc7`) of the two classifiers. Then we stack a bottleneck layer of 256 neurons on `fc7`, then the label classifier ($256 \rightarrow 31$) and the domain classifier ($256 \rightarrow 1024 \rightarrow 1024 \rightarrow 2$).

Since the images in the datasets are sorted by classes, we first shuffle the dataset, otherwise a mini-batch may consist of images from only one or two classes, which makes the training process hard to converge.

We first train the label classifier on the source domain for a few epochs to create a better start point for our domain adaptation. Note that in this step, the weights of the feature extractor are fixed to prevent over-fitting features on the source domain. After that, we optimize the whole network using our approach. The results are shown in Table 5.

Table 5. Accuracy of different training methods on Office dataset

Method / Source / Target	AMAZON / WEBCAM	DSLR / WEBCAM	WEBCAM / DSLR
GFK [9]	0.214	0.691	0.650
SA [6]	0.450	0.648	0.699
DLID [4]	0.519	0.782	0.899
DDC [32]	0.605	0.948	0.985
DAN [19]	0.645	0.952	0.986
Source Only	0.506	0.914	0.967
Proposed method	0.565	0.934	0.989

AMAZON to WEBCAM In this adaptation task, both the accuracies of source only baseline (0.506) and proposed approach (0.542) are lower than the original paper [7] (0.642 and 0.730). Since we use the same dataset and network architecture as the original paper does, the difference

may come from the preprocessing of the dataset, or the initialization of pretrained AlexNet. However, the accuracy still increases by 6 percentiles.

DSLR to WEBCAM The source only baseline of the original paper (0.961) is higher than ours (0.914), but the improvement brought by our implementation (2 percentiles) is more significant than the original paper (0.3 percentile). So, we can claim that our implementation functions well.

WEBCAM to DSLR The proposed approach outperforms all previous methods in this task. This is because the source only baseline is relatively higher than the two previous tasks hence the network has a good initialization. Note that the proposed approach can increase the accuracy by 2 to 6 percentiles in this dataset, if we use the identical way to fine-tune the AlexNet so that it has a better initialization, it's possible that our implementation can outperform all other methods in the two previous tasks.

5.6. GAN and WGAN

To our interest, we investigate the possibility to replace the gradient reversal layer with the concept of generative adversarial network. Both the GAN loss function and the WGAN loss function are experimented, as shown in Algorithm 1 and 2. The MNIST to MNIST-M task are experimented. It is found that the training is very unstable and the discriminator tends to dominate the performance. The result we got is similar to the result that only trained on source only. However, the improvement of the accuracy on target domain test data is observed, which increases from **0.43** to **0.55**. We think this architecture can be further improved by carefully chosen hyperparameters.

Algorithm 1 GAN based domain adaptation algorithm

Randomly initialize the network
Define $L1 = log(G_d(G_f(x_t)))$
Define $L2 = log(1 - G_d(G_f(x_s)))$
Define $L3 = L(G_f(x_s), y_s)$

for e $<$ MaxEpoch **do**
  Fix θ_f and θ_y
  $\theta_d \leftarrow \theta_d - \mu \frac{\partial(-L1-L2)}{\partial \theta_d}$
  Fix θ_f and θ_d
  $\theta_y \leftarrow \theta_y - \mu \frac{\partial L3}{\partial \theta_y}$
  Fix θ_y and θ_d
  $\theta_f \leftarrow \theta_f - \mu \frac{\partial(L1+L2+\lambda L3)}{\partial \theta_f}$
end for

Algorithm 2 WGAN based domain adaptation algorithm

Randomly initialize the network
Define $L1 = G_d(G_f(x_t))$
Define $L2 = -G_d(G_f(x_s))$
Define $L3 = L(G_f(x_s), y_s)$

for e $<$ MaxEpoch **do**
  Fix θ_f and θ_y
  $\theta_d \leftarrow \theta_d - \mu \frac{\partial(-L1-L2)}{\partial \theta_d}$
  Fix θ_f and θ_d
  $\theta_y \leftarrow \theta_y - \mu \frac{\partial L3}{\partial \theta_y}$
  Fix θ_y and θ_d
  $\theta_f \leftarrow \theta_f - \mu \frac{\partial(L1+L2+\lambda L3)}{\partial \theta_f}$
end for

6. Conclusion

In this project, we implement the the paper unsupervised domain adaptation by backpropagation. From our knowledge, we are the first group to replicate the paper with most of the listed experiments in the original paper. The dataset used in the paper are organized that can be used with ease for future research. Moreover, we also investigate the possibility to modify the original loss function by incorporating the concept of generative adversarial network. Our code can be found on `https://github.com/neoqy/DDA`.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. *CoRR*, abs/1608.06019, 2016.

[3] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015.

[4] S. Chopra, S. Balakrishnan, and R. Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains.

[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] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *2013 IEEE International Conference on Computer Vision*, pages 2960–2967, Dec 2013.

[7] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1180–1189. JMLR.org, 2015.

[8] M. Gheisari and M. S. Baghshah. Unsupervised domain adaptation via representation learning and adaptive classifier learning. *Neurocomput.*, 165(C):300–311, Oct. 2015.

[9] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, June 2012.

[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*, pages 2672–2680, 2014.

[11] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition.

[13] J. Hu, J. Lu, Y.-P. Tan, and J. Zhou. Deep transfer metric learning. *Trans. Img. Proc.*, 25(12):5576–5588, Dec. 2016.

[14] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[16] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[17] M. Liu and O. Tuzel. Coupled generative adversarial networks. *CoRR*, abs/1606.07536, 2016.

[18] M. Long, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. *CoRR*, abs/1605.06636, 2016.

[19] M. Long, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. *CoRR*, abs/1602.04433, 2016.

[20] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[21] B. Moiseev, A. Konev, A. Chigorin, and A. Konushin. Evaluation of traffic sign recognition methods trained on synthetically generated data. In J. Blanc-Talon, A. Kasinski, W. Philips, D. Popescu, and P. Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, pages 576–583, Cham, 2013. Springer International Publishing.

[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[23] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1187–1192, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[24] X. Peng and K. Saenko. Synthetic to real adaptation with deep generative correlation alignment networks. *CoRR*, abs/1701.05524, 2017.

[25] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010.

[26] W. sheng Chu Fern. Selective transfer machine for personalized facial action unit detection.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, (0):–, 2012.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions.

[30] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. *CoRR*, abs/1510.02192, 2015.

[31] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017.

[32] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[33] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *CoRR*, abs/1406.5670, 2014.