EE392A

Adversarially Learned Transferable Features

Abhinav Agrawal(14011)

under the guidance of

Prof. Vinay P. Namboodiri, CSE
Prof. Tanaya Guha, EE

Indian Institute of Technology, Kanpur

Undergraduate Project Report

Contents

Abstract

The aim of this project was to explore the use of recent generative modelling approaches in learning transferable features. Specifically, we focus on the task of Domain Adaptation and explore different approaches which employee both variational autoencoders and generative adversarial networks.

1 Introduction

With availability of large datasets (ImageNet[1],COCO[2],PASCAL-VOC[3]) recent advances in deep learning have been able to learn representation that can be used for different tasks([4, 5]). However, due to dataset bias or domain shift, these models are not always able to generalize well to novel data. A prevalent approaches has been to fine-tune such models for the required domain(datasets), this however depends on availability of supervision. Collecting annotations is often expensive and in-feasible. A typical case is when training is done on synthetically created datasets, while the test set comprises of real world data-points which is unsurprisingly from a different distribution.

Recently, a line of work has explored the unsupervised domain adaptation methods which try to alleviate the above mentioned issues of fine-tuning and annotation in case of domain shifts. This is usually achieved by learning representations which are common to the different domains by minimizing a domain-shift distance metric([**?** 6, 7]). The advent of generative models for images has facilitated a reincarnation of such unsupervised models through the use of adversarial learning([8, 9, 10]).

2 Previous Work

As most of the recent unsupervised approaches involve either one of the VAE and GAN models as the main architecture we explain these two preliminaries and the present some recent works.

2.1 Generative Adversarial Network (GAN)

Generative Adversarial Networks(GANs)[11] are a type of Implicit Probability Density Estimation models in which two competing players of a min-max game are able to collectively learn how to generate data similar to that of training examples. This works by training two networks together, a generator network(G) and a discriminator network(D), the architecture of which is flexible and are usually deep neural networks. The generator is given a noise variable z as input, drawn from a prior noise distribution(usually zero mean unit Gaussian distribution), to be transformed into an image. This generated image is then classified/assigned a probability by the D as being real of fake(fake is when it is from the generator). The two networks are trained such that the Generator G is able to fool the discriminator into believing that the images generated by it are same as the real ones, while the Discriminator D tries separate this synthesized image from real training data. This thus becomes a mini-max game between the two networks and who are trying to attain an equilibrium state. The optimization objective is:

$$\underset{G}{min}\ \underset{D}{max}\ E_{x\sim p_{data}}[log\ D(x)] + E_{z\sim p_z}[log\ (1 - D(G(z)))]$$

2.2 Variational Autoencoder (VAE)

Variational autoencoder (VAE)[12] was introduced as an efficient method of variational inference which used a neat re-parametrization trick to facilitate the use of deep neural networks as density estimators. Through some careful manipulations of the terms we can rewrite the log-likelihood of data as :

$$\log p_\theta(x) = \mathbb{KL}[q_\phi(z|x)||p_\theta(z|x)] + \mathcal{L}(p_\theta, q_\phi)$$

Since the first term is a KL-divergence, which is always positive, it is interesting to focus on the other term. In variational inference, we maximize this second term called the evidence lower bound (ELBO)

$$\mathcal{L}(p_\theta, q_\phi) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)]$$

over the space of whole q_ϕ.

By rearranging the ELBO we get the following equation:

$$\log p(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathbb{KL}[q_\phi(z|x)||p(z)]$$

In the first term, log $p_\theta(x|z)$ is the conditional log-likelihood of the observed x given the z. This term is similar to a reconstruction loss for x given the encoding z. Hence, log $p_\theta(x|z)$ is the Decoder(D) The second term is the KL-divergence between $q_\phi(z|x)$ and the prior $p(z)$, which is usually fixed to be $\mathcal{N}(0, I)$. Here, $\mathrm{q}_\phi(z|x)$ works as an Encoder(E). For the re-parametrization process, we:

- First, we sample a noise variable ϵ from a simple distribution like the standard Normal $\epsilon \sim \mathcal{N}(0, I)$
- Then, we apply a deterministic transformation $h_\phi(\epsilon, x)$ that maps the random noise into a more complex distribution $z = h_\phi(\epsilon, x)$. Specifically, if μ and Σ are the mean and co-variance of $q_\phi(z|x)$, we can sample from $\mathcal{N}(\mu, \Sigma)$ by first sampling $\epsilon \sim \mathcal{N}(0, I)$ and then computing $z = \mu + \Sigma^{\frac{1}{2}}\epsilon$.

The biggest advantage of this approach is that we many now write the gradient of an expectation with respect to $q_\phi(z)$ (for any f) as:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{z\sim q_\phi(z|x)}[f(x,z)] &= \nabla_\phi \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[f(x, h_\phi(\epsilon, x))] \\ &= \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)}[\nabla_\phi f(x, h_\phi(\epsilon, x))]\end{aligned}$$

The exact form of p or q are not specified and only limited by the constraints on the best $q_\phi(z|x)$ that should be able to approximate the true posterior $p(z|x)$.

2.3 Domain Adaptation

The problem of domain adaptation has gathered interest from many research in different fields(Natural Language Processing and Vision). Classically the problem is defined when there are two distributions, source (P_S) and target (P_T) which are sufficiently different. The task is further complicated as the source samples are generally presented with some sort of annotation while the target datasets do not. The task is to learn a function that generalizes well to the target dataset for a particular task. Theoretically, it has been shown that a way to do it is to align the latent representations of these two distributions and use them for the needed task. Formally,

$$P(\mathbf{z}|\mathbf{x}_s) = P(\mathbf{z}|\mathbf{x}_t)$$

The recent adversarial approaches were instantiated after the work of Ganin et al.[8] and Ajakan et al. [13] who introduced the DomainAdversarial Neural Network (DANN): a model trained to extract domain-invariant features. They constraint the first few layers of their model to be shared by two the classifiers: first which predicts task-specific class labels from source labels while the second is trained to predict the domain of the inputs. DANNs achieve the minimization of the domain classification loss with respect to parameters specific to the domain classifier, while maximize it with respect to the parameters that are common to both the classifiers. This min-max optimization ensures that the which are common to both the domains are similar and hence the first classifier shall be able to generalize well to target samples. This is achieved in practise using a gradient reversal layer. Tzeng et al.[14] and Long et al. [15] proposed versions of DANNs where the maximization of the domain classification loss is replaced by the minimization of the Maximum Mean Discrepancy (MMD) metric[16], computed between features extracted from sets of samples from each domain. Ghifary et al.[17] propose an alternative model in which the task loss for the source domain is combined with a reconstruction loss for the target domain, which results in learning domain-invariant features. Bousmalis et al. [18] introduce a model that explicitly separates the components that are private to each domain from those that are common to both domains. They make use of a reconstruction loss for each domain, a similarity loss (eg. DANN, MMD) which encourages domain invariance, and a difference loss which encourages the common and private representation components to be complementary.

Other related techniques involve learning a mapping from one domain to the other at a feature level. In such a setup, the feature extraction pipeline is fixed during the domain adaptation optimization. This has been applied in various non-CNN based approaches [19, 20, 21] as well as the more recent CNN-based Correlation Alignment (CORAL) [22] algorithm.

Our approach to domain is most similar to the recently proposed by Bousmalis et al. [23] where first a task invariant domain adaptation process is carried out by conditioning the generation of target images with a source image. They are able to generate a transform their source domain images into target domain while preserving the characteristic features so as to be able to use the same labels for the generated image. This allows the authors to theoretically generate an infinite number of target domain like images with labels to train on the task of interest.

We propose a different architecture motivated by the idea of mapping the representation of the data-points in the two different domains directly. The details of our approach are explained in the following section.

3 Our Approach

We will explain our model for the said task of unsupervised domain adaptation. For now, we work with the classification task, while our model is not depended on any specific task. The general setting of domain adaptation classification task assumes that the labels are available for the source domain dataset and unavailable for the target domain dataset and the goal is to train a classifier on data from the source domain that is able to generalize well to the target domain. Our model decouples tries to decouple the process of domain adaptation and classification by employing a different network for domain adaptation while having another task-specific network for classification. This is similar to the idea of [23], but we differ from them in two major aspects:

- We try to map the representations of the two domains directly through an end-to-end approach by including a KL divergence loss between the representation of the target and source images.
- Ideally, our approach should be able to skip the problems of negative and positive modal modal alignment.

The approach makes an underlying assumption(which is similar to the ones made in [23])the differences between the domains are primarily low-level(due to noise, resolution, illumination, color) and not due to high-level changes(types of objects, geometric variations, etc).

Formally, let $\mathbf{X}_s = \{\mathbf{x}_i^s, \mathbf{y}_i^s\}_{i=0}^{N_s}$ represent a labeled dataset of N_s samples from the source domain and let $\mathbf{X}_t = \{\mathbf{x}_i^t\}_{i=0}^{N_t}$ represent an unlabeled dataset of N_t samples from the target domain. Our pixel adaptation model consists of an Encoder function $E(\mathbf{x}; \theta_E) \rightarrow \mathbf{z}$, parameterized by θ_E that maps an input image to a latent representation. This representation is then fed to a generator function $G(\mathbf{z}; \theta_G)\mathbf{x}_f$, parameterized by θ_G, which converts it into an image of the target domain. Given the generator function G, it is possible to create a new dataset $\mathbf{X}_f = \{G(\mathbf{z}_s), ys\}$ of any size. This and the source dataset, can thus be used to train the task specific module.

3.1 Learning and Loss functions

The details of the learning algorithm are given in the [1] and also explained in the 1. We use the standard DC-GAN[24] architecture for the purpose of both Generator (G) and Discriminator(D). The standard loss function of the GAN model introduced earlier is slightly modified to suit the task. Specifically, apart from feeding the source representations to the G as input, we also sample from a normal distribution. This

Algorithm 1 Training the Model

```
1: $\theta_G, \theta_D, \theta_E, \theta_T \leftarrow$ initialize network parameters
2: **repeat**
3:     $\mathbf{x}_s \leftarrow$ Random minibatch of source
4:     $\mathbf{z}_s \leftarrow E(\mathbf{x}_s)$
5:     $\mathbf{x}_f \leftarrow G(\mathbf{z}_s)$
6:     $\mathbf{z}_n \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
7:     $\mathbf{x}_t^{'} \leftarrow G(\mathbf{z}_n)$
8:     $\mathbf{x}_t \leftarrow$ Random minibatch of target images
9:     $\mathbf{z}_f \leftarrow E(\mathbf{x}_f)$
10:    $\mathcal{L}_{GAN} \leftarrow \mathbb{E}_{\mathbf{x}_t}[\log D(\mathbf{x}_t)] + \alpha\mathbb{E}_{\mathbf{x}_s}[\log(1 - D(\mathbf{x}_f)] + \beta\mathbb{E}_{\mathbf{z}_n}[\log(1 - D(G(\mathbf{z}_n))]$
11:    $\mathcal{L}_{prior} \leftarrow D_{KL}(q(\mathbf{z}_s|\mathbf{x}_s)||p(\mathbf{z}_s))$
12:    $\mathcal{L}_C \leftarrow D_{KL}(q(\mathbf{z}_s|\mathbf{x}_s)||q(\mathbf{z}_f|\mathbf{x}_f))$
13:    $\mathcal{L}_{Task} \leftarrow \mathbb{E}_{\mathbf{x}_s,\mathbf{y}_s,\mathbf{x}_f}[-\mathbf{y}_s^\top \log(\mathbf{z}_f) - \mathbf{y}_s^\top \log(T(\mathbf{z}_s))]$
14:    //Update parameters according to gradients
15:    $\theta_E \leftarrow -\nabla_{\theta_E}(\mathcal{L}_{prior} + \mathcal{L}_C + \mathcal{L}_{Task})$
16:    $\theta_D \leftarrow -\nabla_{\theta_D}(\mathcal{L}_{GAN})$
17:    $\theta_G \leftarrow -\nabla_{\theta_G}(\mathcal{L}_{GAN} + \mathcal{L}_C)$
18:    $\theta_T \leftarrow -\nabla_{\theta_T}(\mathcal{L}_{Task})$
19: **until** convergence
```

has shown to provide better results for such kind of models as stated by authors in [25]. The final loss function looks like :

$$\begin{aligned}\mathcal{L}_{GAN}(D,G) =& \mathbb{E}_{\mathbf{x}_t}[\log D(\mathbf{x}_t;\theta_D)] + \alpha\mathbb{E}_{\mathbf{x}_s}[\log(1 - D(G(E(\mathbf{x}_s;\theta_G);\theta_G);\theta_D)] \\ &+ \beta\mathbb{E}_{\mathbf{z}_n}[\log(1 - D(G(\mathbf{z}_n;\theta_G);\theta_D)]\end{aligned}$$

This loss function is depended on D and G networks and trains only them.

In addition to this GAN loss, we also have a task specific loss for the classification step which trains the Encoder(E) and Task(T) specific network. For the purpose of easing comparison we have kept the architecture of this combination to be same as that of the classifier modules used in [8]. The task specific loss function looks like this:

$$\begin{aligned}\mathcal{L}_{Task}(E,T) =& \mathbb{E}_{\mathbf{x}_s,\mathbf{y}_s,\mathbf{x}_f}[-\mathbf{y}_s^\top \log(T(E(\mathbf{x}_f;\theta_E);\theta_T)) \\ &- \mathbf{y}_s^\top \log(T(E(\mathbf{x}_s;\theta_E);\theta_T))]\end{aligned}$$

Another loss function that figures in the process is the prior loss of the VAE process which helps train the E and G(since G sends a fake target image to Encoder to be used for task specific process).

$$\mathcal{L}_{prior} = D_{KL}(q(\mathbf{z}_s|\mathbf{x}_s)||p(\mathbf{z}_s))$$

3.2 Consistent Representation Loss

Due to the nature of our model, we are able to obtain the representations of the fake target images generated from G. We try to constraint the distributions of these representations and that of the source images to be same by trying to minimize their KL-divergence. In the ideal case, this would elevate the problem of negative and positive alignment and search for common representations.

$$\mathcal{L}_C = D_{KL}(q(\mathbf{z}_s|\mathbf{x}_s)||q(\mathbf{z}_f|\mathbf{x}_f))$$

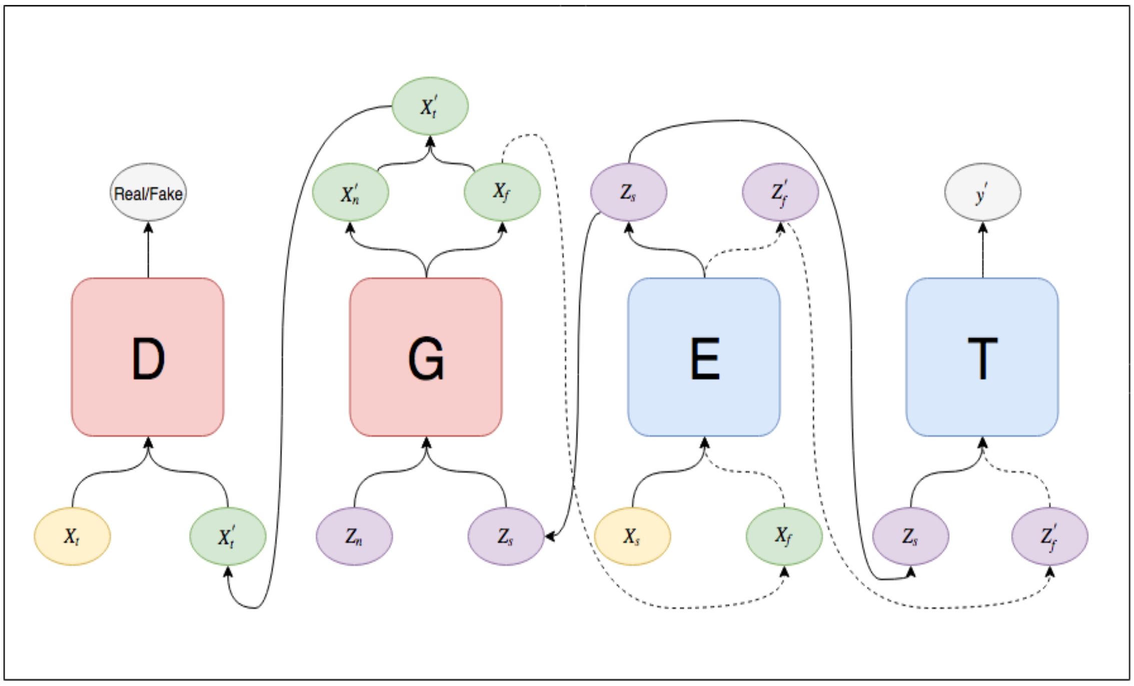

Figure 1: We present an overview of the model architecture. The different networks are represented by their initials(E,D,G and T). The input images from the datasets $\mathbf{X}_s$ and $\mathbf{X}_t$ are represented in yellow nodes. The learned representations are represented in purple nodes. The training process starts from a source image as input to E, which is converted into its latent representation $\mathbf{z}_s$. This is then fed to the G to generate a fake image $\mathbf{x}_f$ which belongs to target domain but encodes source features. and the image generated from sampled noise,$\mathbf{z}_n$, are then send to D as $\mathbf{x}_t^{'}$. is then also fed back to the encoder. This is shown in the dotted lines to avoid confusion, to generate $\mathbf{z}_f$. Both $\mathbf{z}_f$ and $\mathbf{z}_s$ are then fed to the T network. The parameters are updated as stated in [1].

4 Experiments and Results

We work with a restricted setting to first test out the efficacy of our approaches by working with MNIST and MNISTM datasets. MNISTM is a synthesized version of MNIST created specifically for the task of domain adaptation. The digits are $28 \times 28 \times 3$ RGB images with backgrounds added for further distinction from its standard version. We have tried with different variants of the above model but we failed to get a results we hoped for. The model was trained in two variants:

- End-to-end training : In this approach, although we were able to generate the target images but we failed to perform on the task.
- Sequential training: Here we first trained the encoder on the source images and the fixed it to then train the generator and discriminator. In this approach we suffered from the well known problem of modal collapse.

To better understand the reason for this collapse, we trained a classifier(basically the encoder and the task module combined) with source only, target only, and source and target combined images. We developed three different types of this classifier:

- Standard LeNet architecture[26] w/o sampling[1]: We used this as a baseline to compare with the other two models. This same as a standard convolutional neural network based classifier.

Table 1: Accuracy on MNISTM dataset(w/o VAE architecture)

Method	Accuracy
Target Only	96.3
Source Only	58.9
Target and Source Both	96.4

- Standard LeNet with sampled Z[2]: This model was trained only with $\mathcal{L}_{Task}$ and we ignored the $\mathcal{L}_{prior}$ for this. This allows the model to learn a more unconstrained posterior distribution. We were able to obtain the same performance as in the previous case.
- Standarad LeNet with sampled Z[3]: In this version we used both losses, $\mathcal{L}_{Task}$ and $\mathcal{L}_{prior}$ to train the model. Here we observed that the model was able to perform on the individual datasets, but fails to work when the datasets are combined.

Table 2: Accuracy on MNISTM dataset(w/ VAE architecture)

Method	Accuracy
Target Only	96.2
Source Only	58.7
Target and Source Both	93.8

Table 3: Accuracy on MNISTM dataset(w/ VAE architecture)

Method	Accuracy
Target Only	95.2
Source Only	55.4
Target and Source Both	11.8

5 Conclusion and Future work

From the observe results, we still can not reject the hypothesis we started with. Before refuting this model we would like to try some more variants of it which could include these possible modifications:

- Using stronger priors with the same model that zero-mean unit variance Gaussian distributions(Diagonal Gaussian distribution or a more probabilistic approach). This is motivated by the additional experiments conducted with sampling based classifiers. The one of the model was able to learn to classify images while still mapping the latent representations to some Gaussian distribution.

- Using transformations for latent representations(Models like Normalizing flows[27] and Real-NVP[28]). This again comes from the observation that stronger prior distributions of these methods have been able to model the representations better and hence should able to incorporate for the difference in source and target representations.

Another way to approach this same problem could with the use of a model inspired by Cycle-GAN [29]and Adversarially Learned Representations[30]. Exploration of this model is left as a future work.

6 Acknowledgement

I would like to thank Prof. Vinay P. Namboodiri for providing me with valuable support and guidance as and when needed. He graciously allowed me to use the computing resources, vital for this work. I am also equally grateful to Prof. Tanaya Guha for agreeing to be the co-advisor on this project. A special thanks to Mr. Vinod Kumar Kurmi for essential discussions and getting me introduced to the field of domain adaptation.

References

[1] 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.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[3] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[4] 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. In *International conference on machine learning*, pages 647–655, 2014.

[5] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[6] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European Conference on Computer Vision*, pages 443–450. Springer, 2016.

[7] 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.

[8] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[9] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 4068–4076. IEEE, 2015.

[10] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *Advances in neural information processing systems*, pages 469–477, 2016.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[13] Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014.

[14] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[16] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

[17] 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.

[18] 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.

[19] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2066–2073. IEEE, 2012.

[20] Rui Caseiro, Joao F Henriques, Pedro Martins, and Jorge Batista. Beyond the shortest path: Unsupervised domain adaptation by sampling subspaces along the spline flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3846–3854, 2015.

[21] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 999–1006. IEEE, 2011.

[22] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, volume 6, page 8, 2016.

[23] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017.

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[25] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

[26] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[27] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[28] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[29] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

[30] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.