Universidad de Concepción
Dirección de Postgrado
Facultad de Ingeniería - Programa de Magíster en Ciencias de la Computación

ADVERSARIAL VARIATIONAL DOMAIN ADAPTATION FOR SEMI-SUPERVISED IMAGE CLASSIFICATION

Tesis para optar al grado de
MAGÍSTER EN CIENCIAS DE LA COMPUTACIÓN

POR
Manuel Ignacio Pérez Carrasco
CONCEPCIÓN, CHILE
Septiembre, 2019

Profesor guía: Guillermo Felipe Cabrera Vives
Departamento de Ingeniería Informática y Ciencias de la Computación
Facultad de Ingeniería
Universidad de Concepción

ACKNOWLEDGMENTS

Abstract

For success fully training deep neural networks, we usually need a large amount of annotated data in order to avoid the overfitting and being able to generalize to new data. In most of real cases, getting labels is difficult and time consuming. In this work we address the problem of transferring knowledge obtained from a vast annotated source domain to a low labeled or unlabeled target domain, reducing the efforts to get labels on the target. We propose Adversarial Variational Domain Adaptation (AVDA), a semi-supervised domain adaptation method based on deep variational embedded representations. The idea of AVDA is to use a mixture of Gaussian distribution as a prior for the latent space, mapping samples that belong to the same class into the same Gaussian component, independently of the domain membership, using approximate inference. We use adversarial methods to align source and target distributions in latent space for each class independently. We tested our model using the digits dataset, which contains images of handwritten digits and images of number of houses. We empirically show that on a semi-supervised scenario, our approach improved the state of the art for digits dataset from 0.3 to 1.5% of accuracy using only 1 and 5 labels per class. Also, we tested out model using images of galaxies from the Cosmic Assembly Near-infrared Deep Extragalactic Legacy Survey (CANDELS, [23]) as source and the Cluster Lensing and Supernova Survey with Hubble (CLASH, [62]) as target. We empirically show that using few labels our model presents a significant speed-up in terms of the increase in accuracy, and the model keeps improving as more labels we add.

Chapter 1

INTRODUCTION

Deep neural networks have become the state of the art for a lot of machine learning problems. However, these methods usually imply the need for a large amount of labeled data in order to avoid overfitting and be able to generalize. Furthermore, it is assumed that train and test data come from the same distribution and feature space. This becomes a huge problem in cases when labeling is costly and/or time-consuming. One way to address this challenge is to use a source domain which contains a vast amount of annotated data and reduce the differences in distribution (domain shift) between this domain and a different, but similar, target domain in which we have few or even no annotations.

Domain adaptation (DA) methods aim at reducing the domain shift between datasets [58], allowing to generalize a model trained on source to perform similarly on the target domain by finding a common shared space between them. Deep DA uses deep neural networks to achieve this task. Previous works in deep DA have addressed the problem of domain shift by using statistical measures [48, 46, 83, 87, 94, 78, 60] or introducing class-based loss functions [81, 15, 51, 52] in order to diminish the distance between domain distributions. Since the appearance of Generative Adversarial Networks [21] new approaches have been developed focused on using adversarial domain adaptation (ADA) techniques [14, 13]. The goal of adversarial domain adaptation [82] is to learn from the source data distribution a model to predict on the target distribution by finding a common representation for the data by using an adversarial objective with respect to a domain discriminator. This way, a domain-invariant feature space can be used to solve a classification task on both the source and the target.

Despite ADA methods being good at aligning distributions even in an unsupervised domain adaptation (UDA) scenario (i.e. with no labels from the target), they have

problems when facing some domain adaptation challenges. First, since most of these methods were made to tackle UDA problems, they usually fail when there is a significant covariate shift between domains [95]. Second, these methods are not able to take advantage of the semi-supervised scenario in order to produce more accurate models when a few amount of labels are available from the target, generating poor decision boundaries near annotated target data [71]. This behavior has been studied in different works, which tried to adapt domain-invariant features from different classes independently [72, 37].

In this work, we propose Adversarial Variational Domain Adaptation (AVDA), a domain adaptation model which works on unsupervised and semi-supervised scenarios by exploiting target labels when they are available by using variational deep embedding (VaDE, [34]) and adversarial methods [21]. The idea behind AVDA is to correct the domain shift of each class independently by using an embedded space composed by a mixture of Gaussians, in which each class correspond to a Gaussian mixture component. Specifically, for the source domain we map samples that belong to the same class into the same gaussian mixture component in latent space optimizing the Kullback-Leibler (KL) divergence, while for target domain, we use the discriminator and KL divergence to force samples to map into the Gaussian mixture component associated to the objects classes. By doing this, we aim find a common shared feature space for each class independently of the domain membership.

The performance of AVDA was validated on benchmark digit recognition tasks using MNIST [43], USPS [9], and SVHN [55] datasets and on a real case consisting in galaxy images using the Cosmic Assembly Near-infrared Deep Extragalactic Legacy Survey (CANDELS, [23]) as source and the Cluster Lensing and Supernova Survey with Hubble (CLASH, [62]) as target. We demonstrate competitive results among other methods in the state-of-the-art for the digits task and then show the potential of our model obtaining speed-up in performance when few target labels are available even in a high domain shift scenario.

Chapter 2

BACKGROUND AND RELATED WORK

2.1 Deep Neural Networks

2.1.1 Perceptron

The idea of deep neural networks [66] is to mathematically model the human neuron function. In the particular case of feed-forward neural networks, the modeling is made through a unit base called perceptron [66]. The perceptron is defined as

$$\hat{y} = f(\sum_{i=1}^{n} w_i x_i + b), \tag{2.1}$$

where w_i represents the parameters of the perceptron, x_i represents the input and b is an scale factor called bias. The activation function f is a non-linear function which determines the output of the perceptron. The scheme of a perceptron is displayed in Figure 2.1

Using the perceptron we are able to obtain differentiable non-linear functions by using the activation function. The parameters w are fitted during training. At the same time, this non-linear functions allow us to generate new representations of the data in order to perform machine learning tasks.

2.1.2 Feed-forward Neural Network

The connection of many perceptrons generates a feed-forward neural network (FNN). FNN are modeled by layers in which the data sequentially pass through it, generating a final output. Figure 2.2 shows a feed-forward neural network model. The perceptron units are grouped by layers which receive as input the output of the previous layer, generating a new activation. The last layer, called output layer, generates the final

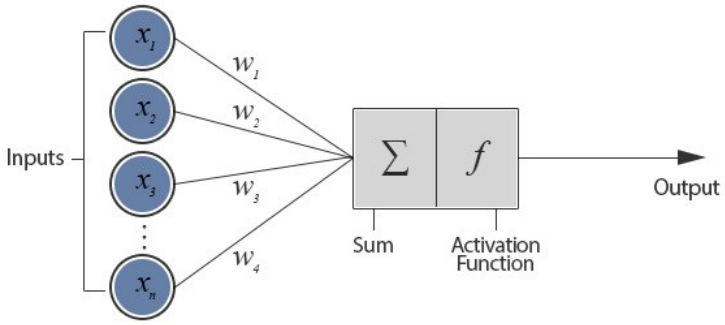

Figure 2.1: Perceptron model, the base unit of a feed-forward neural network. x_i is the input of the model, which is weighted by the parameters w_i of the model and summed with a bias b. The results is passed through an activation function which determines the output of the perceptron.
Source: Harsh Pokharna, For Dummies: The Introduction to Neural Networks we all need. `https://medium.com/technologymadeeasy/for-dummies-the-introduction-to-neural-networks-we-all-need-c50f6012d5eb`

output $\hat{y}$ of the model.

The parameters w_i of the model are iteratively learned by minimizing a loss function between the desired output and the output given by the model. In the classification case, the categorical cross entropy loss function is commonly used, this function is defined by the Equation 2.2:

$$J(w) = \sum_{i=1}^{n} y_i \log \hat{y}_i(x), \tag{2.2}$$

where $\hat{y}_i$ correspond to the output of the model for data i, y_i correspond to the desired real output for data i and n is the total number of data used to compute the loss function.

The parameters are updated using the backpropagation algorithm [44], which computes the partial derivatives of the loss function with respect to the parameters of each layer of the model, iteratively modifying the parameters using an iterative optimization algorithm such as gradient descent. For example, the Stochastic Gradient Descent algorithm (SGD, [65]) updates the parameters in the opposite direction of the gradients, controled by a learning rate. SGD is defined as

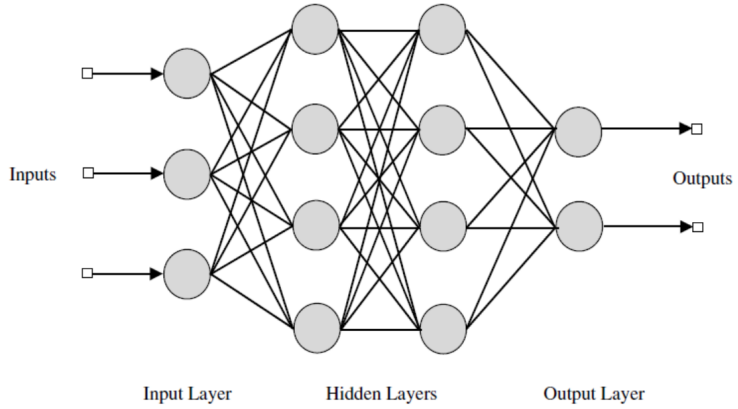

Figure 2.2: Feed-forward neural network model. In this model, perceptron units are grouped by layers called hidden layers, which takes as input the output of the previous layers, generating a new activation. Then, a final layer generates the output $\hat{y}$ of the model.
Source: Virtual Labs, Multilayer feedforward neural networks.
`http://cse22-iiith.vlabs.ac.in/exp4/index.html`

$$w_{t+1} = w_t - \eta \nabla_w J(w_t), \tag{2.3}$$

where w_t are the parameters at iteration t, η correspond to the learning rate, and $\nabla_w J(w_t)$ represents the gradient of the loss function with respect to the parameters w.

2.1.3 Convolutional Neural Networks

Convolutional neural networks [12, 44] are deep learning models commonly used when data exhibit a topological structure to preserve (e.g. pixels in an image). The objective is to model the data using different abstraction levels through layers using convolution and statistical operations (e.g. maxpooling [53]).

The convolutions are operations realized by a kernel which slides through the entire input and computes the dot product between the input and the parameters of the kernel. The result of this operation is a tensor that represents an abstraction of the original data. This representation will give us important features in order to perform the machine learning task using a predictive model, as for example, a feed-forward neural

network [92] .

Despite convnets being invented in 1987 [12], the lack of computational power, enough amount of data and efficient regularization algorithms to avoid overfitting, did not allow successful implementations until a few time ago. In 2012, thanks to the ReLU [54] activation functions, dropout regularization [26, 76] and graphic processing units (GPU's), Krizhevsky et al. 2012 [42] obtained an important advantage over the rest of competitors in the *Large Scale Visual Recognition Challenge [68]*[1] (ILSVRC) of ImageNet, one of the most important challenges in the computer vision field, positioning convolutional neural networks as the best alternative for processing images in machine learning tasks. Since then, the most succesfull algorithms have been based on this kind of architectures [75, 79, 25] using some variations and increasing the number of the convolutional layers, thanks to algorithms that allow a better backpropagation of the errors (e.g. batch-normalization [31])

2.2 Autoencoders

2.2.1 Vanilla Autoencoder

Autoencoders [67, 85] are unsupervised models which learn a function $f_w(x) \approx x$. In other words, they generate an output $\hat{x}$ that looks similar to the original input x. To do that, autoencoders use a compressed representation of the data in a latent space. Autoencoders can be decomposed in two parts that look as mirorred connected networks. The first part (a.k.a. encoder) maps the data into the latent space, while the second part (a.k.a. decoder) reconstructs the original data using the information contained in the latent space. By doing that, the latent space, captures important features that characterize an underlying representation of the data. We can optimize the parameters w of the model by minimizing the average reconstruction error

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} L(x_i, f_w(x_i)), \tag{2.4}$$

[1] http://www.image-net.org/challenges/LSVRC/

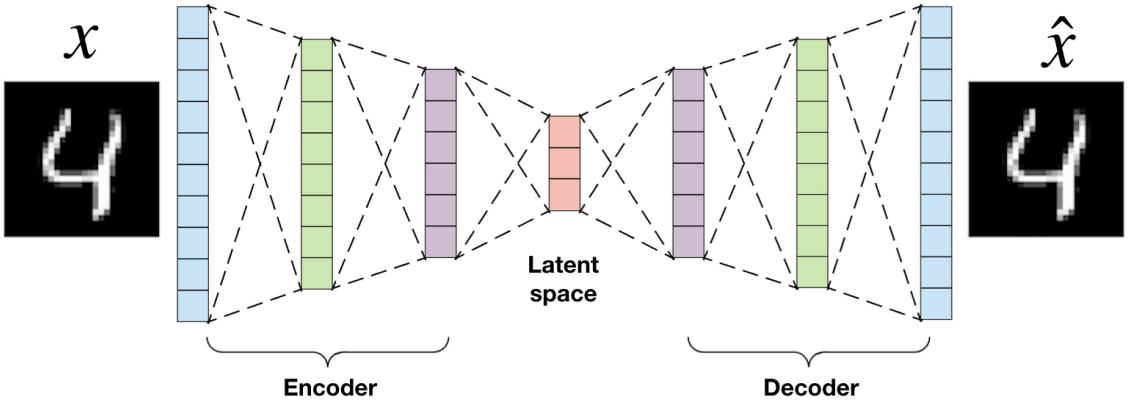

Figure 2.3: Representation of a vanilla autoencoder. The idea is to generate an output $\hat{x}$ that looks similar to the input x using a compressed representation of the data in a latent space. The encoder tries to maps the data x into the latent space, while the decoder generates a reconstruction $\hat{x}$ from this latent space.

where L is a loss function such as the mean squared error if the input x has a continuous representation or binary crossentropy if x is a binary vector. The scheme of a vanilla autoencoder is shown in Figure 2.3.

2.2.2 Variational Autoencoder

Variational Autoencoders (VAE, [41, 35]) are deep generative models. Similar as vanilla autoencoders, VAEs learn a function $f_w(x) \approx x$, generating data that looks like the input data using a latent representation. The idea of variational autoencoders is to encode data into a latent space that follows a known distribution from which we can sample new data.

Formally, given data x and a probability distribution of the data $p(x)$, we encode the data into latent variables z that follow a probability distribution $p(z)$ and generate new data following a given distribution $p(x|z)$. If we knew $p(x|z)$ and $p(z)$ we could find the probability distribution of the data $p(x)$ that generates new data x, by marginalizing over z

$$p(x) = \int_z p(x|z)p(z)dz. \tag{2.5}$$

Given that the calculation of Equation 2.5 is intractable due the difficulty of approximating $p(z)$ in high dimensionensions through sampling, the idea is to infer $p(z)$ through $p(z|x)$, which is likely to produce values of $p(x|z)$ for which x is non-zero, using variational inference. In variational inference we approximate the posterior distribution $p(z|x)$ with a family of simpler distributions $q(z|x)$ minimizing the difference of both distributions using Kullback-Liebler (KL) divergence, which measure the lost information when using $q(z|x)$ instead of $p(z|x)$. The KL divergence can be defined as follows:

$$\begin{aligned} D_{KL}[q(z|x)||q(z|x)] &= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q(z|x)}{p(z|x)}\right], \\ &= \mathbb{E}_{q_\phi(z|x)}[\log q(z|x) - \log p(z|x)]. \end{aligned} \tag{2.6}$$

Applying Bayes rule

$$D_{KL}[q(z|x)||p(z|x)] = \mathbb{E}_{q_\phi(z|x)}[\log q(z|x) - \log p(x|z) - \log p(z)] + \log p(x), \tag{2.7}$$

using Equation 2.7 we can approximate the probability distribution of $p(x)$ via maximizing the variational lower bound as follows:

$$\begin{aligned} \log p(x) &\geq \mathbb{E}_{q_\phi(z|x)}[\log p(x|z) - (\log q(z|x) - \log p(z))], \\ &= \mathbb{E}_{q_\phi(z|x)}[\log p(x|z)] - D_{KL}[q(z|x)||p(z)]. \end{aligned} \tag{2.8}$$

In this Equation, $q_\phi(z|x)[\log p(x|z)]$ correspond to a maximum likelihood estimation, which can be easily optimized by minimizing a statistical measure like the mean squared error between the input x and the output $\hat{x}$ of the network. For $D_{KL}[q(z|x)||p(z)]$, we have to use a prior probability distribution $p(z)$ in order to compute the KL divergence and optimize it. Assuming $p(z) = \mathcal{N}(0, I)$ as a prior probability distribution, there is an analytical solution given two function approximators $\mu(x)$ and $\sigma(x)$, which can be

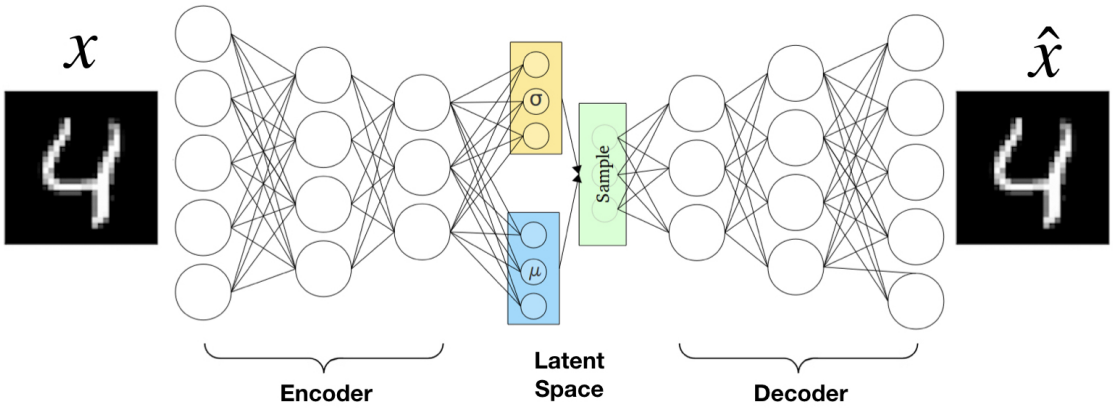

Figure 2.4: Representation of a variational autoencoder. Similar to a vanilla autoencoder, is composed by an encoder and a decoder. The data is mapped into a latent space that follows a known distribution. We are able to sample from this know distribution, generating new data through the decoder.

computed as

$$D_{KL}[\mathcal{N}(\mu(x), \sigma(x))||\mathcal{N}(0, I)] = \frac{1}{2}\sum_{i}^{J}(\sigma(x_i) + \mu^2(x_i) - \log\sigma(x_i) - 1), \tag{2.9}$$

where J is the dimension of the latent space. Using Equation 2.9, we can use the outputs μ and σ of the encoder as the function approximator to minimize the KL divergence. The scheme of variational autoencoders is shown in Figure 2.4.

2.3 Semi-supervised Learning

In most of real cases, unlabeled data is abundant and easy to obtain, while labeling data is expensive and time-consuming. On the other hand, deep neural networks models need a huge amount of labeled data in order to avoid the overfitting and be able to generalize to test data

The idea of semi-supervised learning is to take advantage of the unlabeled data on top of labeled samples [4]. Intuitively, the key ideas for the success of a semi-supervised classification models is to have an internal representation of the data that meets the following characteristics: 1) have samples from the same class nearby each

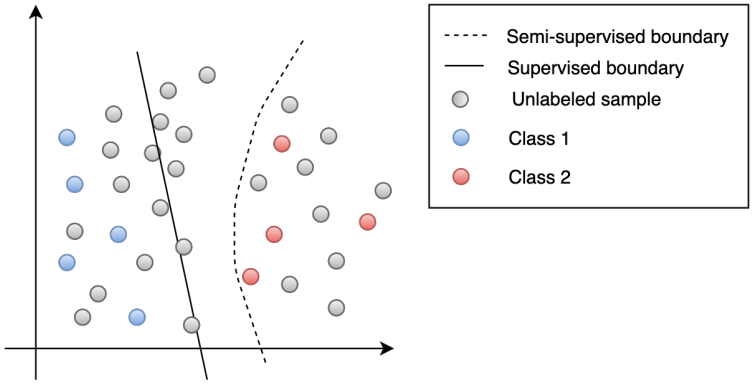

Figure 2.5: In a semi-supervised learning scenario we aim to find low density zones that divide samples from different classes, performing accurate decision boundaries around them.

other, 2) have samples from different classes in different regions, 3) create decision boundaries around the zones in which the density of points is lower. By doing this, we aim to find boundaries that better separate the points that belong to different classes. As we can see in Figure 2.5, when we take into account unlabeled samples and we learn the decision boundaries around low density zones, we are able to create better decision boundaries as compared to using only the limited labeled data.

2.4 Semi-supervised Variational Autoencoders

A promising approach to deal with labeled and unlabeled data during training are semi-supervised variational autoencoders [40, 64, 49]. These models aim at learning a latent space which depends on the labeled data. As the latent space is shared between labeled and unlabeled data, points from the same class are expected to be closer in the latent space. The generative semi-supervised model described in [40] defines a generative process as follows:

$$p(y) = Cat(y|\pi), \qquad p(z) = \mathcal{N}(z|0, I), \qquad p_\theta(x|z, y) = p(x; z, y, \theta), \tag{2.10}$$

where $Cat(y|\pi)$ is modeled by a categorical distribution parametrized by π, the prior probability for class y, $\pi \in \mathcal{R}_+^K$, $\sum_{i=1}^K \pi_i = 1$, where K is the number of classes. z and

y are treated as latent variables if no class is available. If labels are available, only z is treated as a latent variable. $p(x; z, y, \theta)$ is the likelihood of x, defined as a non-linear transformation of the variable z and y with parameters θ. In this work, we use deep neural networks as a non-linear function approximation for $f(x; z, y, \theta)$.

Similar as in variational autoencoders, we approximate the posterior distribution $p(z, y|x)$ by another function $q_\phi(z, y|x)$, as an inference model. The inference model is defined as follows:

$$q_\phi(y|x) = Cat(y|\pi_\phi(x)), \qquad q_\phi(z|y, x) = \mathcal{N}(z|\mu_\phi(y, x), \sigma^2_\phi(x)I) \tag{2.11}$$

where $\mu_\phi(x)$ and $\sigma^2_\phi(x)$ are the mean and variance outputed by the neural network with parameters ϕ. $Cat(y|\pi_\phi(x))$ is a categorical distribution with prior π which is given by a probability vector that depends on x. For this model we can derive two variational lower bounds, one for the labeled data and other for unlabeled data. For a single labeled datapoint the variational lower bound is given by:

$$\begin{aligned} \log p(x, y) &\geq \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x|y, z) + \log p_\theta(y) + \log p(z) - \log q_\phi(z|x, y)], \\ &= \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x|y, z)] + C + D_{KL}[q_\phi(z|x, y)||p(z)], \\ &= -\mathcal{L}(x, y). \end{aligned} \tag{2.12}$$

For a single unlabeled datapoint, we treat both z and y as latent variables. The variational lower bound is given by:

$$
\begin{aligned}
\log p(x) &\geq \mathbb{E}_{q_\phi(z,y|x)}[\log p_\theta(x|y,z) + \log p_\theta(y) + \log p(z) - \log q_\phi(y,z|x)], \\
&= \mathbb{E}_{q_\phi(z,y|x)}[\log p_\theta(x|y,z) + C + \log p(z) - \log q_\phi(y|x) - \log q_\phi(z|x)], \\
&= \mathbb{E}_{q_\phi(y|x)}[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|y,z) + C + \log p(z) - \log q_\phi(y|x) - \log q_\phi(z|x)]], \\
&= \mathbb{E}_{q_\phi(y|x)}[\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|y,z)] + C + D_{KL}[q_\phi(z|x)||p(z)] - \log q_\phi(y|x)], \\
&= -\mathcal{U}(x),
\end{aligned}
\tag{2.13}
$$

where $\log p_\theta(y) = C$ is a constant which represent class proportions. The final objective function be defined as the summation of the lower bounds for labeled and unlabeled data as

$$
\mathcal{J} = \sum_{(x,y)\sim\tilde{p}_l(x,y)} \mathcal{L}(x,y) + \sum_{x\sim\tilde{p}_u(x)} \mathcal{U}(x). \tag{2.14}
$$

Notice that a distribution $q_\phi(y|x)$ was defined in Equation 2.11. This distribution will help the labeling process when labels are missing. In order to improve the labeling process for unlabeled data, we can take advantage of the labeled data points adding an extra discriminative term to the objective. Also, this discriminative term will be encharged of enforce the separation of the gaussians into the latent space. The overall objective can be defined as:

$$
\mathcal{J} = \sum_{(x,y)\sim\tilde{p}_l(x,y)} \mathcal{L}(x,y) + \alpha\, \mathbb{E}_{\tilde{p}_l(x,y)}[-\log q_\phi(y|x)] + \sum_{x\sim\tilde{p}_u(x)} \mathcal{U}(x), \tag{2.15}
$$

where $\tilde{p}_l(x,y)$ and $\tilde{p}_u(x)$ are the empirical distributions over labeled and unlabeled data respectively, and α is an hyperparameter that controls the relative importance between the discriminative and generative process of the model.

In recent works, the latent space is often extended to an embedding in which the latent space corresponds to an embedding representantion [34] and each class is mapped into an embedding component. Our proposed Adversarial Variational Domain Adaptation framework is based on semi-supervised variational autoencoders using a

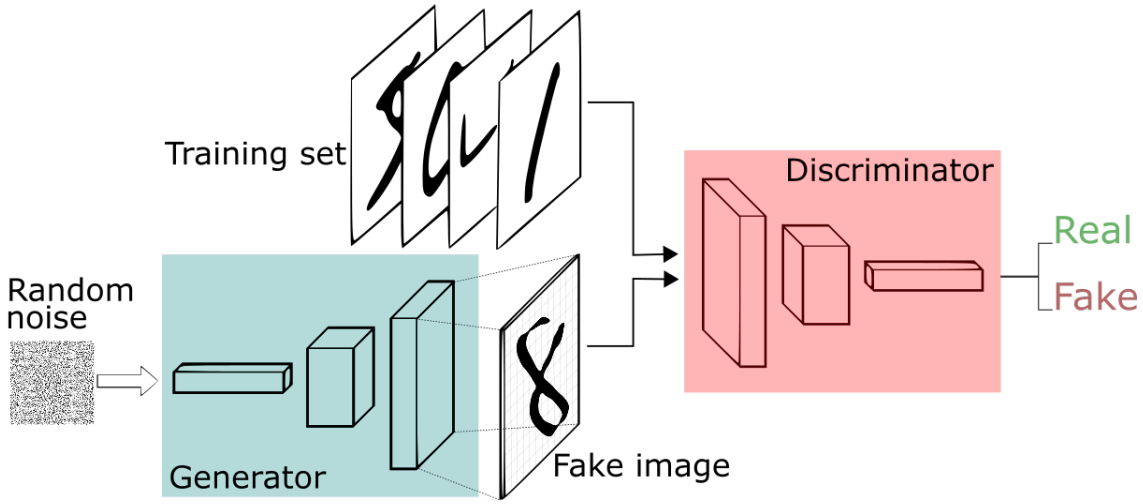

Figure 2.6: Overall process of GANs. The generator generates data that look like the real data using noise as input. On the other hand, the discriminator tries to distinguish if the data is real or was generated. Doing that, we aim to capture the underlying data distribution.
Source: Thalles Silva, An Intuitive Introduction to Generative Adversarial Networks.
`https://www.freecodecamp.org/news/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394/`

variational deep embedding representation.

2.5 Adversarial Learning

Generative adversarial networks (GAN, [21]) are models composed of two networks: a generator and a discriminator. The idea of GANs is to find the underlying distribution behind the data by using these two networks under a two-player game. The generator tries to generate data that looks like the real one, while the the discriminator tries to distinguish if the data is real or if it was generated. The network is trained in a minimax fashion, forcing the generator to fool the discriminator, while the discriminator learns how to not be fooled. The overall process is shown in Figure 2.6.

The optimization of a GAN process is shown in Equation 2.16:

$$\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log 1 - D(G(z))] \quad (2.16)$$

where $D(\cdot)$ and $G(\cdot)$ repreents the discriminator and generator respectively and z represents the noise from which the input of the generator is sampled.

2.6 Transfer Learning

Consider a problem in which we have a large amount of data in one domain (source) and a limited amount of data on another domain (target) and they do not have the same feature space or data distribution. The goal of transfer learning is to improve the performance on the target domain by exploiting similarities between the source and target domains.

Formally, consider a domain $\mathcal{D}$ composed of a d-dimensional feature space $X \subseteq \mathbb{R}^d$ and a marginal probability distribution $p(x)$. For this domain, a task $\mathcal{T}$ is defined by a label space $\mathcal{Y}$. A conditional probability $p(y|x)$ is learned from the pairs $\{x_i, y_i\}$ (where $x_i \in X$ and $y_i \in \mathcal{Y}$). Given a source domain $\mathcal{D}^s$ and a target domain $\mathcal{D}^t$, the goal of transfer learning is to learn $p(y^t, x^t)$ exploiting previous learned probability $p(x^s, y^s)$, given that $p(x^s, y^s) \neq p(x^t, y^t)$ [59].

Several approaches have been developed in classical machine learning in order to learn the conditional probability $p(y^t|x^t)$ [7]. For example the *instance re-weighting methods* propose to estimate the likelihood of being source or target in order to decrease the importance of the missclassified source examples [11] [5] [90]. *Parameter adaptation methods* try to directly modify the classifier trained on source domain in order to perform better on the target domain [36] [88] [6].

Due to the capability of deep networks to learn transferable [2, 91, 57] and invariant [20] representations of the data, deep transfer learning techniques have become in a wide and successful area of research [80]. These methods use deep neural architectures designed to exploit similarities betweeen domains. Deep transfer learning can be grouped in three categories: *Shallow transfer learning* considers a deep architecture for feature extraction and these features can be used in any of the methods described before (e.g. [10], [8], [77]). *Fine-tuning deep CNN architectures* is used to retrain some layers of a deep architecture trained on source, with target samples [91, 57]. Better

results in deep transfer learning have been achieved by *Deep domain adaptation architectures*, which are methods designed for aligning source and target distribution into a shared internal representation.

2.6.1 Domain Adaptation

Domain adaptation methods deal with the challenge of transfer learning by reducing the domain shift between the source and target domains [58]. This is achieved by aligning the data in a common intern representation for them.

Some statistical metrics have been proposed in order to align source and target distributions, such as maximum mean discrepancy (MMD) [48, 46, 83, 87], Kullback Leibler (KL) divergence [94] or correlation alignment (CORAL) [78, 60]. Since the appearance of Generative Adversarial Networks [21] significant work has been developed around adversarial domain adaptation (ADA) techniques [14, 13, 16]. The idea of ADA methods is to use a domain classifier which discriminates if a sample belongs to the source or target domain, while a generator learns how to create indistinguishable representations of data in order to fool the domain classifier. By doing this, a domain-invariant representation of the data distribution is produced in a latent space.

Despite ADA models achieving good results either by matching distributions in a feature representation (i.e. feature-level) [13, 45, 47, 74, 69] or generating target images that look as if they were part of the source dataset (i.e. pixel-level) [32, 93, 28, 30, 29], when they are used in a UDA scenario, they have difficulties dealing with big covariate shifts between domains [95]. Furthermore, when a few number of annotated target samples are included, these models often do not improve performance relative to just **training** with labeled target samples [71]. In order to deal with few labels, few-shot domain adaptation methods have been created [51, 52], which are not meant to work with unlabeled data, often producing overfitted representations and having problems to generalize on the target domain [86].

Semi-supervised domain adaptation (SSDA) deal with these challenges using both labeled and unlabeled samples during training [19, 22, 18, 73, 1, 72]. Usually for SSDA

we are interested on finding a space in which labeled and unlabeled target samples belonging to the same class have a similar internal representation [10, 89, 95, 71].

Our proposed AVDA framework uses a variational deep embedding [34] representation, in which both source and target samples that belong to the same class are mapped into an embedding component, allowing the model to obtain a significant speed-up in performance as more labels are used from the target domain.

Chapter 3

THE METHOD

In this work we propose Adversarial Variational Domain Adaptation (AVDA), a model based on semi-supervised variational deep embedding (SSVaDE) and adversarial methods. We use a Gaussian mixture model as a prior for the embedded space and align samples from source and target domains that belong to the same class into the same Gaussian component.

3.1 Problem Definition

In a semi-supervised domain adaptation scenario, we are given a source domain $\mathcal{D}^s = \{x_i^s, y_i^s\}_{i=1}^{n^s}$ with n^s number of labeled samples and a target domain $\mathcal{D}^t = \{(x_i^t, y_i^t)\}_{i=1}^{n^t}$ with n^t number of labeled samples. Also, for the target domain we have a subset $\mathcal{D}^u = \{(x_i^u)\}_{i=1}^{n^u}$ of n^u unlabeled samples. For both domains we have the same K classes, i.e. $y_i^s \in \{1, ..., K\}$, $y_i^t \in \{1, ..., K\}$. Source and target data are drawn from unknown joint distributions $p^s(x^s, y^s)$ and $p^t(x^t, y^t)$ respectively, where $p^s(x^s, y^s) \neq p^t(x^t, y^t)$.

The goal of this work is to build a model that provides an embedding space z in which source and target data have the same representation for each of the K classes. We propose the use of a Semi-supervised Variational Deep Embedding [20]. This model is composed by the inference models $q_\phi^s(z^s|x^s)$ and $q_\psi^t(z^t|x^t)$ that encodes source and target data into this latent representation, which we set to be a mixture of Gaussian distribution depending on the labels y and they are parametrized by ϕ and ψ, for source and target respectively. Also, the generative model $p_\theta^s(x^s|z^s)$ describes the data as if they were generated from a latent variable z^s and is parametrized by θ. A discriminative process $q_\phi^s(y^s|x^s)$ is included to enforce the separability between the Gaussian mixture components. The overall model is displayed in Figure 3.1.

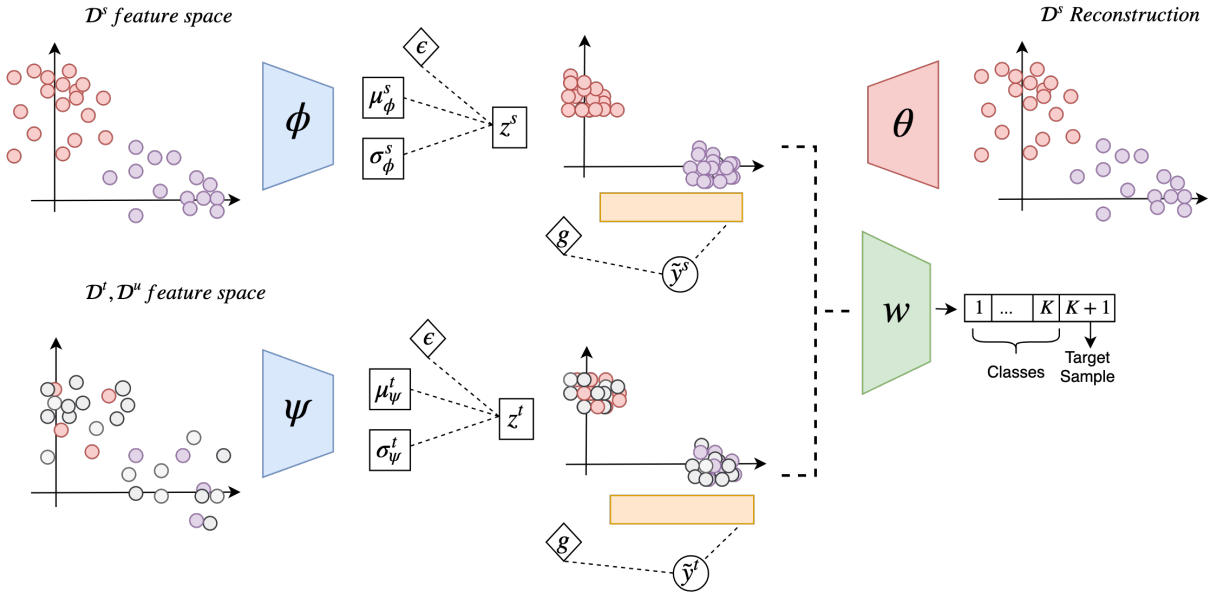

Figure 3.1: Overall architecture for Adversarial Variational Domain Adaptation. The model works in three steps. First, source data $\mathcal{D}^s$ is encoded into an embedded space z^s using the inference model with parameters ϕ and decoded using the generative model with parameters θ. Each data point is mapped to an embedded component associated to its class. Second, a classifier with parameters w is trained to discriminate if a sample comes from the source domain and its class, or comes from the target domain. Third, the target inference model is trained in an adversarial fashion using target data $\mathcal{D}^t$ and $\mathcal{D}^u$, generating an aligned embedding representation for source and target domains.

3.2 Adversarial Variational Domain Adaptation Model

For the source domain, we define a generative process as follows:

$$p(y) = Cat(y|\pi) \qquad p(z|y) = \mathcal{N}(z|\mu(y), \sigma^2(y)) \qquad p_\theta(x^s|z) = p(x^s; z, \theta) \tag{3.1}$$

where $Cat(y|\pi)$ is a multinomial distribution parametrized by π, the prior probability for class y, $\pi \in \mathcal{R}^K_+$, $\sum_{y=1}^K \pi_i = 1$. At the same time, $\mu(y)$ and $\sigma^2(y)$ are the mean and variance of the embedded normal distribution corresponding to class labels y. $p(x^s; z, \theta)$ is a likelihood function whose parameters are formed by non-linear transformations of the variable z using a neural network with parameters θ. In this work, we use deep neural networks as a non-linear function approximation.

For source and target domains, we define two inference models. We use variational

inference to find an approximation for the true posterior distribution $p(y, z|x)$ using the approximated posterior distributions $q_\phi(y, z|x)$ and $q_\psi(y, z|x)$ which are parametrized by a deep neural networks with parameters ϕ for the source domain and ψ for the target domain. We assume the approximate posterior can be factorized as $q(y, z|x) = q(z|x)q(y|z)$, and model it by using normal and categorical distributions as follows:

$$q_\phi(z^s|x^s) = \mathcal{N}(z|\mu_\phi(x^s), \sigma^2_\phi(x^s)) \qquad q_\phi(y^s|z^s) = Cat(y^s|\pi_\phi(z^s)) \tag{3.2}$$

$$q_\psi(z^t|x^t) = \mathcal{N}(z|\mu_\psi(x^t), \sigma^2_\psi(x^t)) \qquad q_\psi(y^t|z^t) = Cat(y^t|\pi_\psi(z^t)) \tag{3.3}$$

where $(\mu_\phi(x^s), \sigma_\phi(x^s))$ and $(\mu_\psi(x^t), \sigma_\psi(x^t))$ are the outputs of the source and target deep neural networks with parameters ϕ and ψ respectively, and are then used to sample z from a Gaussian mixture distribution by using the reparametrization trick defined in [41]. $q_\phi(y^s|z^s)$ and $q_\psi(y^t|z^t)$ represent the source and target processes modeled through independent neural networks. These networks take the latent variables and return the parameters $\pi_\phi(z^s)$ and $\pi_\psi(z^t)$ to sample a categorical variable by using a Gumbel-Softmax distribution [34]. With this estimator, we can generate labels y and backpropagate through this sampled categorical variable by using the continuous relaxation defined by Jiang et al. 2016 [34], avoiding the marginalization over all the categorical labels introduced in [40, 34], significantly reducing computational costs.

3.3 Variational Objectives

The supervised variational lower bound on the marginal likelihood for a single source data point can be derived similarly to a deep generative model [41, 40] as follows:

$$\log p_\theta(x^s) \geq \mathbb{E}_{q_\phi(z^s|x^s)}\left[\log p_\theta(x^s|z^s)\right] - D_{KL}\big(q_\phi(z^s|x^s)||p_\theta(z^s)\big) \tag{3.4}$$

where the labels y are used to map each sample to their correspondent embedding in latent space. At the same time, a predictive function $q_\phi(y^s|x^s)$ is included in order to enforce the separability between the embedding components. The lower bound for

source domain can be optimized by minimizing the following objective:

$$\mathcal{L}^s_{sup} = D_{KL}\big[q_\phi(z^s|x^s)||p_\theta(z^s)\big] - \mathbb{E}_{q_\phi(z^s|x^s)}\left[\log p_\theta(x^s|z^s)\right] - \alpha^s \log q_\phi(y^s|x^s) \tag{3.5}$$

where α^s is the hyper-parameter that controls the relative importance between the generative and discriminative processes of the model.

On the other hand, for the target domain, we would like that the inference model will be able to map the samples into the same embedding obtained by the source generative model. Taking this into account, the objective can be decomposed in two parts. One for labeled data and other for unlabeled data. For a single target labeled data point we can obtain the supervised objective as follows:

$$\mathcal{L}^t_{sup} = D_{KL}\big(q_\psi(z^t|x^t)||p^*_\theta(z^t)\big) - \alpha^t \log q_\psi(y^t|x^t) \tag{3.6}$$

where $p^*_\theta(z)$ is the optimized prior distribution for the source. α^t is the hyper-parameter that controls the relative importance of the discriminative process in the model. For a single target unlabeled data point we can derive the unsupervised objective as follows:

$$\begin{aligned}\mathcal{L}^t_{unsup} &= D_{KL}(q_\psi\big(z^t, y^t|x^t)||p^*_\theta(z^t, y^t)\big) \\ &= \mathbb{E}_{q_\psi(z|x)}\left[D_{KL}(q_\psi(y^t|z^t)||p^*_\theta(y^t))\right] + \mathbb{E}_{q_\psi(y|z)}\left[D_{KL}(q_\psi(z^t|x^t)||p^*_\theta(z^t|y^t))\right]\end{aligned} \tag{3.7}$$

The minimization of this term helps generating the mapping of each unlabeled target sample component into the correspondent embedding space using a predicted categorical variable.

3.4 Adversarial Objective

We would like the agglomerative distribution over the approximated posterior distribution $q(z) = \mathbb{E}_{p^*(x)}\left[q(z|x)\right]$ to be the same for source and target domains (i.e $q_\psi(z^s) = q_\phi(z^t)$). These distributions are embedded spaces which depend on the labels, hence we use the semi-supervised generative adversarial model proposed by Odena A. 2016

[56] in order to encourage the alignment between these distributions. In particular, we use a discriminator $D_w(\cdot)$ with parameters w which takes the form of a classifier distinguishing between $K+1$ classes, where the first K classes correspond to the source classes and the class $K+1$ correspond to a class representing the data was generated by the target inference model. By doing this, we encourage the discriminator to learn the underlying distribution of each class independently. This discriminator, can be optimized using the following objective:

$$\mathcal{L}^D = \sum_{(x_i,y_i)\in\mathcal{D}^s} H(D(q^s_\phi(z^s_i)), y^s_i) + \sum_{(x_i)\in\mathcal{D}^u\cup\mathcal{D}^t} H(D(q^t_\psi(z^t_i)), y' = K+1) \tag{3.8}$$

where $H(\cdot,\cdot)$ is the cross entropy loss. On the the other hand, we try to confuse D via an adversarial loss which forces the inference model of the target to learn a mapping from the samples to their correspondent embedding component. We can optimize the parameters of the target inference model using the following objective:

$$\mathcal{L}^A = \sum_{(x_i)\in\mathcal{D}^u} H(D(q^t_\psi(z^t_i)), \tilde{y_i}^t)) + \sum_{(x_i,y_i)\in\mathcal{D}^t} H(D(q^t_\psi(z^t_i)), y^t_i)) \tag{3.9}$$

where $\tilde{y}^t$ are the predicted categorical variables sampled from the Gumbel-Softmax distribution for unlabeled target samples and y^t the real class for labeled target samples.

3.5 Overall Objectives and Optimization

In this section we describe the overall objective of the model and how this objective can be optimized. The training process is done in three steps: train the source model, train the discriminator, and train the target model.

Source Step: The first step consists in optimizing the source model. By doing this, we can obtain an embedding space which will be used later to map target samples into the same embedding components. The overall objective for the source domain is defined in Equation 3.5 and it is composed of three terms. The first term can be

computed analytically by following the proof of [34] as follows:

$$D_{KL}\big[q_\phi(z|x)||p_\theta(z)\big] = \sum_{i=1}^{n^s}\Big(-\frac{J}{2}\log(2\pi) + \frac{1}{2}\sum_{j=1}^{J}\Big(\log(2\pi\sigma_j^2(y)) + \frac{\sigma_\phi^2(x_{ij})}{\sigma_j^2(y)} + \frac{(\mu_\phi(x_{ij}) - \mu_j(y))^2}{\sigma_j^2(y)} - 1 - \log\sigma_\phi^2(x_{ij})\Big)\Big) \tag{3.10}$$

where $\mu_\phi(x)$ and $\sigma_\phi^2(x)$ are the approximated mean and variance given by the neural network. $\mu(y)$ and $\sigma^2(y)$ are the mean and variance of each embedding component. J is the dimensionality of $\mu_\phi(x)$ and $\sigma_\phi^2(x)$. The second term can be optimized by computing the expectation of gradients using the reparametrization trick defined in [41, 35] as follows:

$$\nabla_{\{\theta,\phi\}}\,\mathbb{E}_{q_\phi(z|x)}\big[\log p_\theta(x|z)\big] = \mathbb{E}_{\mathcal{N}(0,I)}\big[\nabla_{\{\theta,\phi\}}\log p_\theta(x|\mu_\phi(x) + \sigma_\phi^2(x)\odot\epsilon)\big] \tag{3.11}$$

where ∇ denotes simple gradients over the parameters ϕ and θ. $\odot$ is the element-wise product. The third discriminative term can be trivially optimized by minimizing the cross entropy loss between real labels and predicted labels as estimated by the predictive function.

Discriminative Step: The discriminative step is done by minimizing Equation 3.8 with respect to the parameters w of the discriminator $D_w(\cdot)$. The goal of this step is to encourage the discriminator to learn the embedding representation generated by using the source domain. Then, we minimize the target inference model in order to fool the discriminator mapping samples into the same embedding. For this purpose, the discriminative step and later introduced target step are performed alternately.

Target Step: The overall objective for the target domain can be written as follows:

$$\mathcal{L}^t = \gamma\mathcal{L}^t_{sup} + (1-\gamma)\mathcal{L}^t_{unsup} + \mathcal{L}^A \tag{3.12}$$

where γ is a hyperparameter that controls the relative importance between labeled and unlabeled samples. In this Equation, $\mathcal{L}^t_{sup}$ can be optimized using Equation 3.10. $\mathcal{L}^t_{unsup}$

can be decomposed into two terms as in Equation 7. Following the derivation of [1], we can compute the derivatives of the second term using expectation of gradients and the reparametrization trick defined in [33] to derive a Monte Carlo estimator as follows:

$$\nabla_\phi \mathbb{E}_{q_\phi(y|z)} \left[D_{KL}(q_\phi(z|x)||p_\theta(z|y)) \right] = \mathbb{E}_{Gumbel(g|0,1)}[\nabla_\phi D_{KL}(q_\phi(z|x)||p_\theta(z|\tilde{y}))] \tag{3.13}$$

where $\tilde{y}$ is a predicted categorical variable sampled using the Gumbel-softmax relaxation defined as:

$$\tilde{y} = argmax(f(\phi, g, \tau)), \qquad f(\phi, g, \tau)_{1 \leq k \leq K} = \frac{\exp((\tilde{\alpha}_k + g_k)/\tau)}{\sum_{i=1}^{K} \exp((\tilde{\alpha}_i + g_i)/\tau)} \tag{3.14}$$

where $\tilde{\alpha}_k$ are parameters outputed by the neural network $\pi_\phi(z)$, g is a random variable sampled from $g \sim Gumbel(0, 1)$ and τ is a hyperparameter that regulates the entropy of the sampling. This reparametrization trick allows us to discretize y during the forward pass, while we can use a continuous approximation in the backward pass. The KL divergence of Equation 3.13 is similar to the one introduced for source optimization, hence we can use the analytical solution introduced in Equation 3.10. The derivatives of the second term of Equation 3.12 can be computed as:

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)} \left[D_{KL}(q_\phi(y|z)||p_\theta(y)) \right] = \mathbb{E}_{\mathcal{N}(\epsilon|0,1)}[\nabla_\phi D_{KL}(q_\phi(y|z)||p_\theta(y))] \tag{3.15}$$

The third term of Equation 3.12 can be optimized by minimizing Equation 3.9 with respect to the parameters ψ. The overall training process is sumarized in Algorithm 1.

Algorithm 1 Training procedure for Adversarial variational Transfer

Pretrain the model finding $\{\theta, \phi\}$ by minimizing $\mathcal{L}^s$
while not converge **do**:
    $\mathcal{D}^s \leftarrow$ getSupervisedMiniBatchSource()
    $\mathcal{D}^t \leftarrow$ getSupervisedMiniBatchTarget()
    $\mathcal{D}^u \leftarrow$ getUnsupervisedMiniBatchTarget()
    $\mathcal{L}^D \leftarrow$ Equation 3.8, $\quad \forall\{x_i, y_i\} \in \mathcal{D}^s \wedge \forall\{x_i\} \in \mathcal{D}^u$
    $w_{t+1} \leftarrow w_t - \eta\nabla_{w_t}$
    $\mathcal{L}^t \leftarrow$ Equation 3.12, $\quad \forall\{x_i, y_i\} \in \mathcal{D}^t \wedge \forall\{x_i\} \in \mathcal{D}^u$
    $\psi_{t+1} \leftarrow \psi_t - \eta\nabla_{\psi_t}$
end while

Chapter 4

EXPERIMENTS

We evaluate our framework on the digits dataset composed of MNIST [43], USPS [9], and Street View House Numbers (SVHN) [55]. We then apply it to a real case scenario using galaxy images from the Cosmic Assembly Near-infrared Deep Extragalactic Legacy Survey (CANDELS, [23]) as source and the Cluster Lensing and Supernova Survey with Hubble (CLASH, [62]) as target.

4.1 Digits

We use three benchmark digits datasets: MNIST (**M**), USPS (**U**), and Street View House Numbers (SVHN) (**S**). Theses datasets contain images of digits from 0 to 9 in different environments: **M** and **U** contain handwritten digits while, **S** contain natural scene images. Each dataset is composed of its own train and test set: $60,000$ train samples and $10,000$ test samples for MNIST; $7,291$ train samples and $2,007$ test samples for USPS; and $73,257$ train samples and $26,032$ test samples samples for SVHN. For adaptation purposes we used, the evaluation protocol proposed in CyCADA [28], i.e. three domain adaptation task are performed: from MNIST to USPS (**M**$\rightarrow$ **U**), from USPS to MNIST (**U** $\rightarrow$ **M**), and from SVHN to MNIST (**S** $\rightarrow$ **M**).

4.1.1 Implementation Details

For all the tasks, images were rescaled to $[-1.0, 1.0]$ and resized to 32x32. The SVHN dataset contains RGB images, so for **S** $\rightarrow$ **M**, the MNIST images were repeated in each of the three filters in order to use the same input tensor size to the network. The hyperparameters were empirically selected by measuring the performance on 100 randomly selected samples from the training set, performing 5 cross-validated experiments. Specifically, for all scenarios α^s was set as 1000 and α^t as 10. γ was set as

1.0 and linearly decreased to 0.90 after 25 unsupervised batches, τ was set as 3. Our embedding is created on a 20-dimensional space, where the means μ_ϕ and standard deviations σ_ϕ of each Gaussian component are learnt via backpropagation. The means are initially set along different axes so that $||\mu_\phi|| = 10$ and $\sigma_\phi = \vec{1}$ (all-ones vector). Our training was performed using the Adam optimizer [39] with parameters $\beta_1 = \beta_2 = 0.5$ and a learning rate of 0.001 using mini-batches of 128 samples. For fair comparison, we used similar network architectures as the ones proposed in [45, 30].

4.1.2 Results

We compare our results with state-of-the-art methods in UDA and SSDA scenarios. In the UDA scenario, we use a fully labeled source and a fully unlabeled target to perform the adaptation. In Table 4.1, we show our results and compare them to previous approaches as obtained from their papers. We report the mean accuracy and its standard deviation across ten random experiments and the accuracy obtained by our best run. Even though our method is not designed for an unsupervised scenario, it is competitive on **M** $\rightarrow$ **U** and **U** $\rightarrow$ **M**. Our method performs poorly on **S** $\rightarrow$ **M**, which presents a higher domain shift, and ACAL performs the best by matching features at a pixe-level adding a relaxation on the cycle consistency, replacing it by a task-specific loss.

For the SSDA scenario, we perform experiments using one label per class on the target (denoted as 1-shot) and five labels per class on the target (denoted as 5-shot). The rest of the training samples are used in an unsupervised fashion. We compare our results against other methods that utilize the same number of labels per class. Notice that CCSA [52] and FADA [51] do not utilize unlabeled samples during training while we do. The results are computed using the same procedure as before and are displayed in Table 4.2. Here, we outperform all previous approaches. Our approach has a higher speed of adaptation in the sense that by using a small number of labels in the target domain we are able to obtain competitive results.

Method	Unsupervised		
	M → U	U → M	S → M
UNIT [45]	95.97	93.58	90.53
SBADA-GAN [69]	97.6	95.0	76.1
CyCADA [28]	95.6 ± 0.2	96.5 ± 0.1	90.4 ± 0.4
CDAN [47]	95.6	98.0	89.2
DupGAN [30]	96.01	**98.75**	92.46
ACAL [29]	**98.31**	97.16	**96.51**
CADA [95]	96.4 ± 0.1	97.0 ± 0.1	90.9 ± 0.2
AVDA (ours) best	97.63	97.68	76.80
AVDA (ours) random	96.92 ±0.92	96.59 ±0.80	76.98 ±1.36

Table 4.1: Results on *Digits* dataset for unsupervised task. All the results are reported using accuracy by performing 10 random experiments.

Method	1-shot			5-shot		
	M → U	U → M	S → M	M → U	U → M	S → M
CCSA [52]	85.0	78.4	-	92.4	88.8	-
FADA [51]	89.1	81.1	72.8	93.4	91.1	86.1
F-CADA [95]	97.2	97.5	94.8	98.3	98.6	95.6
AVDA (ours) best	**98.63**	**98.41**	**95.80**	**98.68**	**98.62**	**97.19**
AVDA (ours) random	98.36 ±0.14	98.25 ±0.27	95.13 ±0.47	98.5 ±0.19	98.59 ±0.028	96.93 ±0.18

Table 4.2: Results on *Digits* dataset for semi-supervised task. All the results are reported using accuracy by performing 10 random experiments.

4.1.3 Ablation Study

We examined the performance of AVDA adopting three different training strategies, in which we change critical components of our framework. First, we investigate the use of fixed priors during training (i.e they are not updated via backpropagation). We denoted this experiment as AVDA_{FP}. Second, we investigate the model in a classical adversarial domain adaptation scenario (i.e the discriminator tries to distinguish between samples generated by source or target). We denote this experiment as AVDA_{ADA}. Third, we investigate the model when a target generative model is included. We denote this experiment as AVDA_{GT}. The experiments were performed in the most difficult digit scenario **S**→ **M** using five labels per class. For the first experiment, AVDA_{FP} obtained

an accuracy of 92.17 ± 1.88, hence lowering the accuracy of our model. For the second experiment, AVDA$_{\text{ADA}}$ obtained an accuracy of 95.13 ± 1.03, increasing the **variance** of the model performance. For the third experiments, AVDA$_{\text{GT}}$ obtained an accuracy of 91.76 ± 0.39, decreasing the discriminative capability of our model. In consequence, AVDA components obtain the best performance as compared to these three slightly variant frameworks.

4.1.4 Visualization

In order to visualize the alignment between source and target domains, we visualize the embedding space by using t-distributed stochastic neighbor embedding (t-SNE, [84]) for the task **S**→ **M** considering a 5-shot scenario. Figure 4.1 shows this visualization. On the left, we show each class in a different color, demonstrating the classifying capability of the model. On the right, we show the source and target in different colors, demonstrating the ability of the model to generate good alignments between the data labels and the embedded components for both domains. In figure 4.2 we visualize some predictions of the model over target considering the same 5-shot scenario. For this experiment we achieve 96.44% of accuracy.

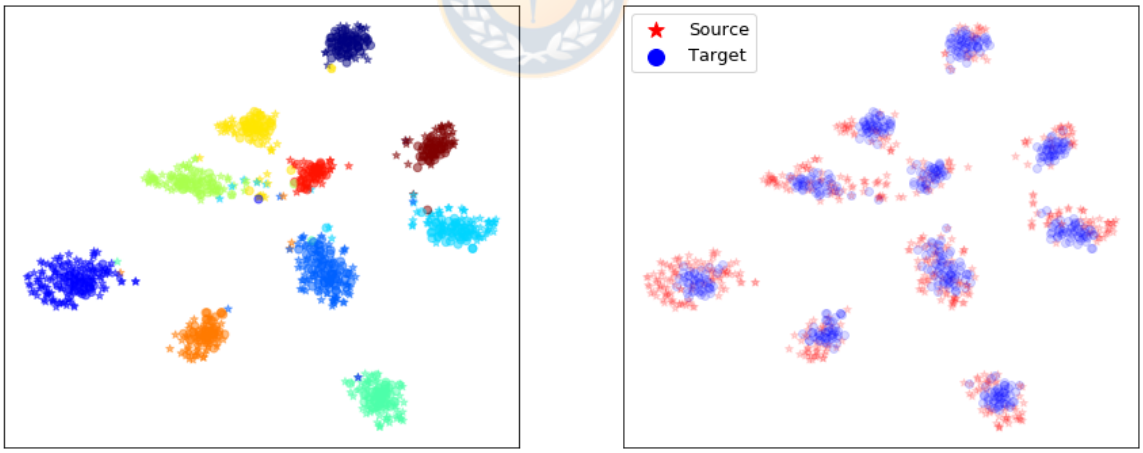

Figure 4.1: Visualization of the embedding space using t-SNE for the **S**→ **M** task considering a 5-shot scenario. Colors on the right panel represent the data labels, showing the capability of the model to generate good discriminative boundaries. Colors on the right panel represent source and target data showing the capability of the model to align source and target domains into the same embedding components.

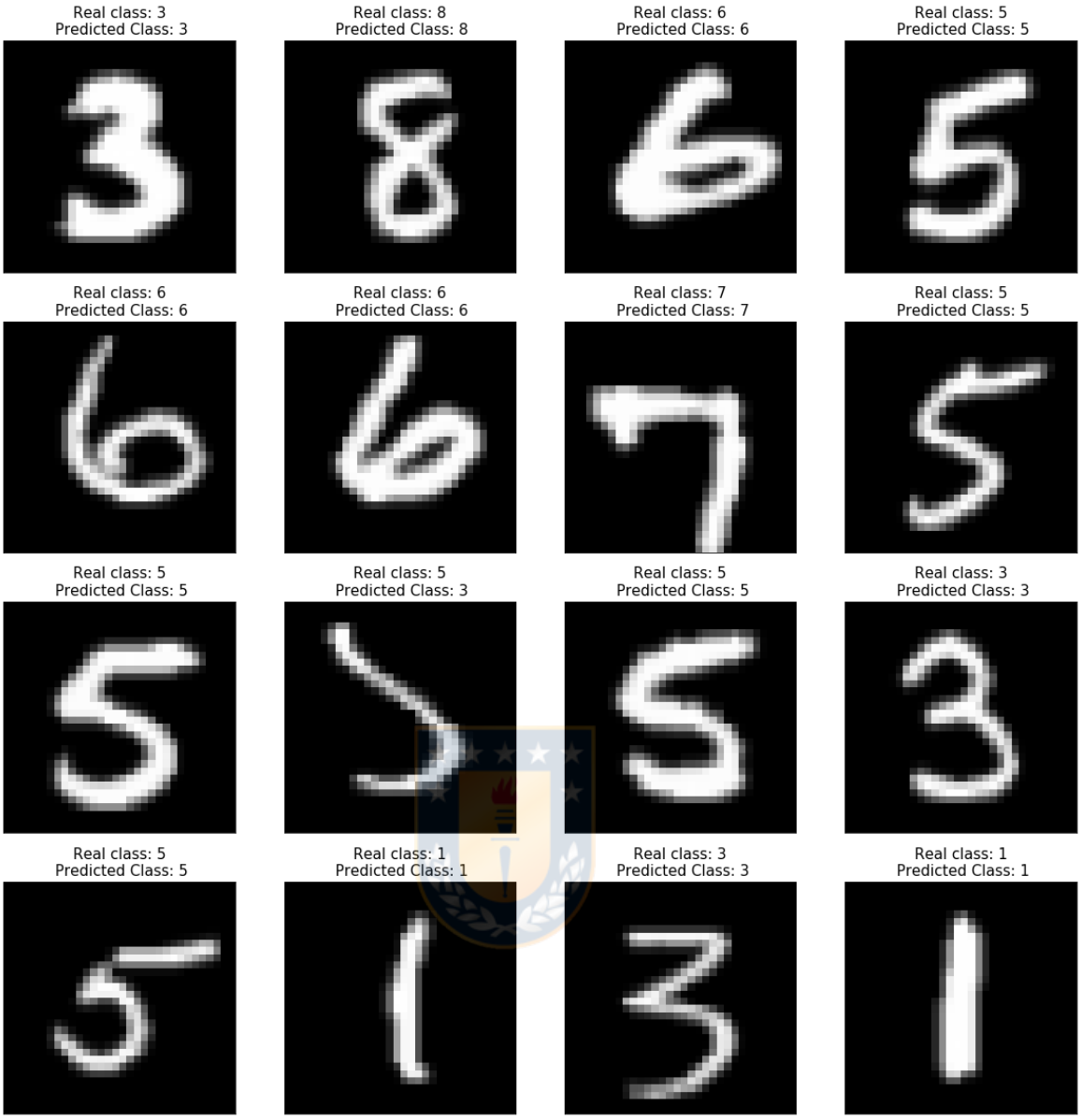

Figure 4.2: Visualization of predictions over target for the **S**→ **M** task considering a 5-shot scenario. On top of each image is displayed the real class and the class predicted by the model.

4.2 Galaxies

We use galaxy images from CANDELS [23] and CLASH [62] and address the problem of classifying them according to their morphologies: smooth, features, irregular, point

source, and unclassifiable[1]. To train our baseline model, we used HST images from a CANDELS field taken with WFC3 in the F160W band, GOODS-S [17][2]. Labels were created by expert as described in Kartaltepe et al. 2014 [38]. To these images we added the mosaic from Hubble Legacy Fields (HLF) Data Release 1.5 for the GOODS-S region (HLF-GOODS-S) [3]. We used the $0.06''$/pixel resolution version in the filter F160W, selecting galaxies with F160W magnitudes $H_{mag} < 24.5$. We created postage-stamp images from the GOODS-S mosaic setting the size to four times the galaxy radius as reported in the catalog of Guo et al. 2013 [24]. We obtain a final sample of $7,567$ galaxies.

For the transfer learning sample we used images from the CLASH Multi-Cycle Treasury program [63]. CLASH observed 25 clusters of galaxies with WFC3 over a period of 3 years, in up to 16 filters, namely F225W, F275W, F336W, F390W, F435W, F475W, F606W, F625W, F775W, F814W, F850W, F105W, F110W, F125W F140W and F160W, covering the ultra violet (UV), optical (OPT) and NIR regions of the spectrum. Labels for CLASH were also created by experts as described in Perez-Carrasco et al. 2018 [61], containing $1,600$ labels in total. Molino et al. 2017 [50] published accurate multiwavelength photometric catalogs for these clusters which also provide the galaxy radius. We created postage-stamp images for each filter separately following the same criterion for the magnitude cut and the size that we adopted for CANDELS.

For both datasets, labels are described as the probability for the galaxy to have a certain morphological type. This probability is defined as:

$$P_T = \frac{N_T}{N_{\rm tot}}, \tag{4.1}$$

where N_T is the number of people who assigned a type T to the galaxy and $N_{\rm tot}$ is the total number of people who classified that galaxy. We used as the sample class the class given by P_T if $P_T \geq 0.5$

[1] Data is public available at:
`https://drive.google.com/open?id=1BSc42VfAb2Mw0zlQShTFUnbCQaf11q4q`

[2] The Great Observatories Origins Deeps Survey

4.2.1 Implementation Details

For this task, images were rescaled to $[0.0, 1.0]$ and resized to 32x32. The hyperparameters were empirically selected by measuring the performance on 3 randomly selected samples from the training set. α^s was set as 1000 and α^t as 10. γ was set as 1.0 and linearly decreased to 0.85 after 25 unsupervised batches, τ was set as 3. Our embedding is created on a 20-dimensional space, where the means μ_ϕ and standard deviations σ_ϕ of each Gaussian component are learnt via backpropagation. The means are initially set along different axes so that $||\mu_\phi|| = 10$ and $\sigma_\phi = \vec{1}$ (all-ones vector). Our training was performed using the Adam optimizer [39] with parameters $\beta_1 = \beta_2 = 0.5$ and a learning rate of 0.001 using mini-batches of 128 samples. We used similar network architectures as the proposed in [45].

4.2.2 Results

For the morphology classification task, we trained 6 different models using 0, 1, 5, 10, 25 and 50 labeled target samples per class. Also, a model using full labeled target was trained. As in a classical semi-supervised setting, all unlabeled target images were used for training and evaluation. We show the results in Figure 4.3. We can notice that just a few number of labeled samples are enough to make important corrections in the domain shift, observing a significant speed-up in performance when labels are included.

We can notice that there is a difference of $\sim 8\%$ between a model trained with 50 labeled samples and a model trained using all the labels. Given that the classes are computed using P_T if $P_T \geq 0.5$, for the classes exist a probability of $1 - P_T$ that this galaxy does not belong to the right class implying that the galaxy could should has a doubtful morphology. In this sense, the galaxies with lower P_T should be close to another class in latent space. Using full labeled source, we are able to perform better decision boundaries around the galaxies with doubtful morphologies.

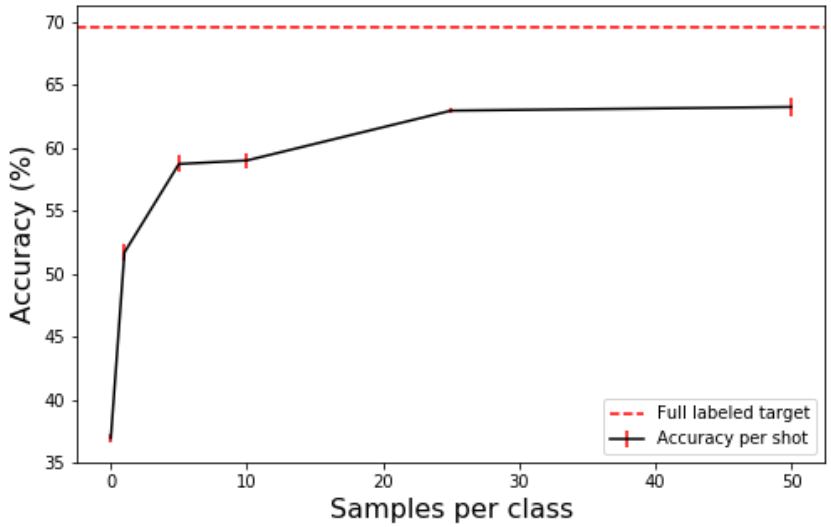

Figure 4.3: Performance in terms of accuracy in morphology recognition task using 0, 1, 5, 10, 25 and 50 number of labeled target samples per class represented by a black solid line. Performance by using full labeled target is presented by a red dashed line. Results are reported by performing 5 random experiments.

4.2.3 Visualization

We visualize the embedding space by using t-SNE [84] for the task **CANDELS**$\rightarrow$ **CLASH** considering a 50-shot scenario. Figure 4.4 shows this visualization. On the left, we show each class in a different color. On the right, we show the source and target in different colors. In figure 4.5 we visualize some predictions of the model over target considering the same 50-shot scenario. For this experiment we achieve 62.06% of accuracy.

We can notice that in latent space some classes are very close, this is because for this experiment we do not have hard classes, instead we are using as the object class, the given by P_T if $P_T \geq 0.5$. Objects from different classes that look similar, should be close in latent space too given the reconstruction term.

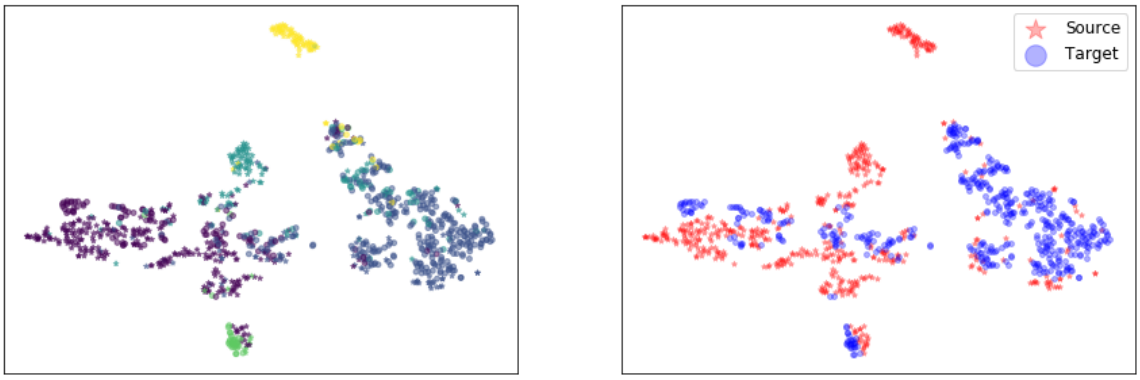

Figure 4.4: Visualization of the embedding space using t-SNE for the **CANDELS→ CLASH** task considering a 50-shot scenario. Colors on the left panel represent the data labels. Colors on the right panel represent source and target data.

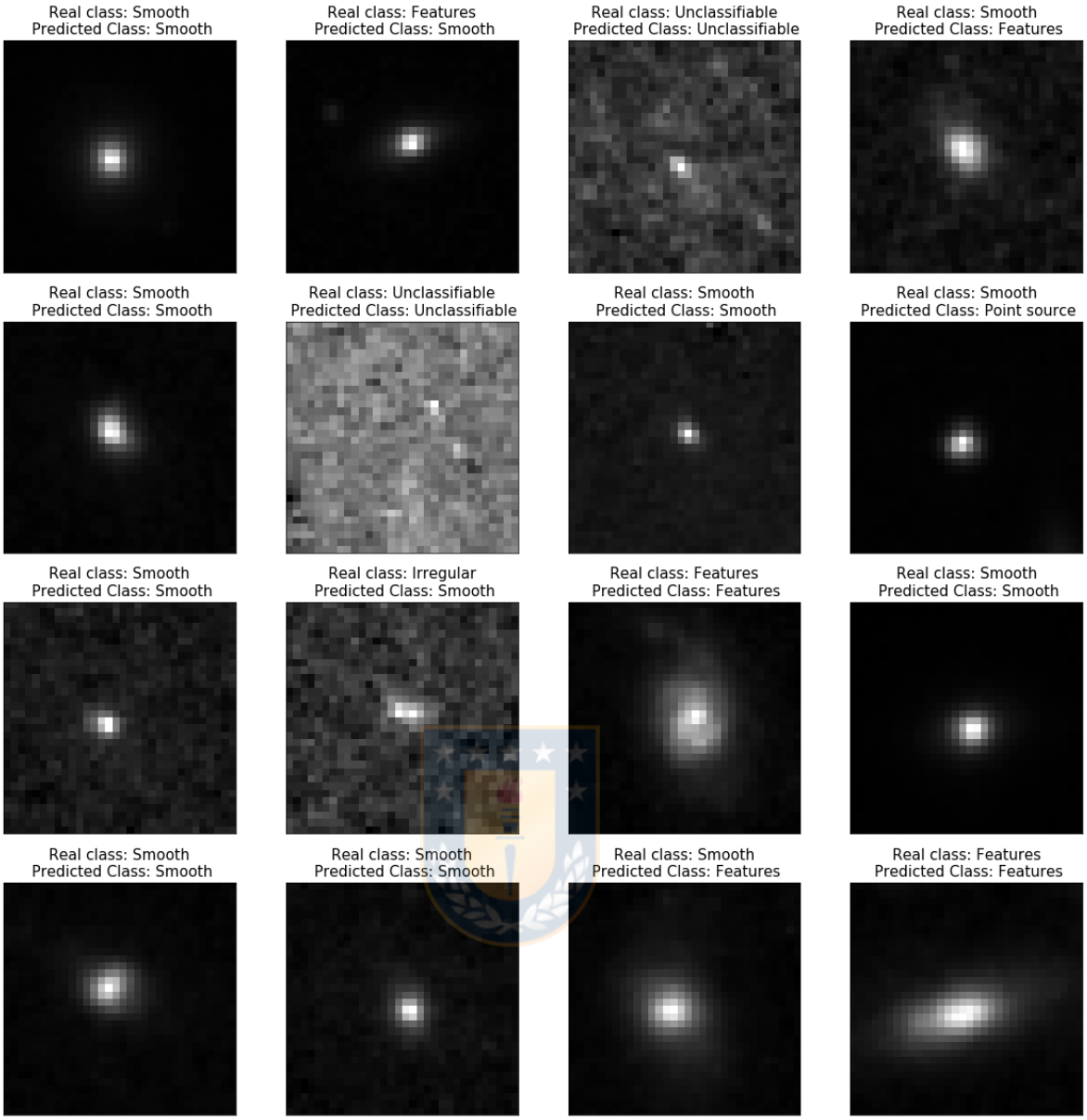

Figure 4.5: Visualization of predictions over target in the **CANDELS**→ **CLASH** task considering a 50-shot scenario. On top of each image is displayed the real class and the class predicted by the model.

Chapter 5

CONCLUSION

In this work we present Adversarial Variational Domain Adaptation (AVDA), a semi-supervised approach for domain adaptations problems were a vast annotated source domain is available but none or few labels from a target domain exist. Unlike previous methods which align source and target domains into a single common feature space, we use a variational embedding and align samples that belong to the same class into the same embedding component using adversarial methods.

Experiments on digits and galaxy morphology classification problems are used to validate the proposed approach. Despite we do not achieve competitive results in all the tasks in an unsupervised fashion on digits classification, we outperform previous state of the art methods from 0.3 to 1.5% of accuracy in a semi-supervised fashion using 1 and 5 labels per class. Our model presents a significant speed-up in terms of the increase in accuracy as more labeled examples are used from the target domain, increasing the accuracy more than 15% using only one label per class for the morphology classification task, demonstrating the capability of the model to align embedding spaces in high domain shift scenarios.

One of the difficulties of this work was finding the prior distribution parameters for the variational model. In some low-domain shift tasks (such as **M** $\leftrightarrow$ **U**), the chosen of fixed parameters μ and $sigma$ for each class achieve good results, while in high domain shift it was necessary to learn these parameters of the distributions during training. For future work we investigate better ways to find the prior distribution parameters. By finding better prior distributions we will be able to align difficult tasks such as Office-31 [70].

Also, we investigate the using of deeper architectures for encoding complex data into embedding representations.

Finally, we intend to extend the present work in order to be useful for semi-supervised

domain adaptation using videos by mixing VaDE [34] and LSTM [27]. The idea is to generate a hidden representation of the data that correspond to an embedding distribution depending on the class, being able to map target samples that belong to the same class into the same distribution into the LSTM's latent representation, as AVDA does.

Bibliography

[1] Marouan Belhaj, Pavlos Protopapas, and Weiwei Pan. Deep variational transfer: Transfer learning through semi-supervised deep generative models. *CoRR*, abs/1812.03123, 2018.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *arXiv e-prints*, page arXiv:1206.5538, Jun 2012.

[3] R. J. Bouwens, G. D. Illingworth, P. A. Oesch, I. Labbé, P. G. van Dokkum, M. Trenti, M. Franx, R. Smit, V. Gonzalez, and D. Magee. UV-continuum Slopes of 4000 z ˜ 4-8 Galaxies from the HUDF/XDF, HUDF09, ERS, CANDELS-South, and CANDELS-North Fields. , 793:115, October 2014.

[4] Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010.

[5] W. Chu, F. D. L. Torre, and J. F. Cohn. Selective transfer machine for personalized facial action unit detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3515–3522, June 2013.

[6] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Large scale transductive svms. *J. Mach. Learn. Res.*, 7:1687–1712, December 2006.

[7] Gabriela Csurka. Domain adaptation for visual applications: A comprehensive survey. *CoRR*, abs/1702.05374, 2017.

[8] Gabriela Csurka, Boris Chidlowskii, Stéphane Clinchant, and Sophia Michel. Unsupervised domain adaptation with regularized domain instance denoising. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, pages 458–466, Cham, 2016. Springer International Publishing.

[9] John S. Denker, W. R. Gardner, Hans Peter Graf, Donnie Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Henry S. Baird, and Isabelle Guyon. Neural network recognizer for hand-written zip code digits. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 323–331. Morgan-Kaufmann, 1989.

[10] J. Donahue, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell. Semi-supervised domain adaptation with instance constraints. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 668–675, June 2013.

[11] Miroslav Dudík, Steven J. Phillips, and Robert E Schapire. Correcting sample selection bias in maximum entropy density estimation. In Y. Weiss, B. Schölkopf,

and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 323–330. MIT Press, 2006.

[12] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, Apr 1980.

[13] Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation. *arXiv e-prints*, page arXiv:1409.7495, Sep 2014.

[14] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January 2016.

[15] Timnit Gebru, Judy Hoffman, and Li Fei-Fei. Fine-grained recognition in the wild: A multi-task domain adaptation approach. *CoRR*, abs/1709.02476, 2017.

[16] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation. *CoRR*, abs/1607.03516, 2016.

[17] M. Giavalisco, H. C. Ferguson, and A. N. Koekemoer. The great observatories origins deep survey: Initial results from optical and near-infrared imaging. *The Astrophysical Journal*, 2004.

[18] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[19] 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.

[20] Ian Goodfellow, Honglak Lee, Quoc V. Le, Andrew Saxe, and Andrew Y. Ng. Measuring invariances in deep networks. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 646–654. Curran Associates, Inc., 2009.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[22] R. Gopalan, Ruonan Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 International Conference on Computer Vision*, pages 999–1006, Nov 2011.

[23] Norman A. Grogin, Dale D. Kocevski, and S. M. Faber. Candels: The cosmic assembly near-infrared deep extragalactic legacy survey. *The Astrophysical Journal Supplement*, 2011.

[24] Y. Guo, H. C. Ferguson, M. Giavalisco, G. Barro, S. P. Willner, M. L. N. Ashby, T. Dahlen, J. L. Donley, S. M. Faber, A. Fontana, A. Galametz, A. Grazian, K.-H. Huang, D. D. Kocevski, A. M. Koekemoer, D. C. Koo, E. J. McGrath, M. Peth, M. Salvato, S. Wuyts, M. Castellano, A. R. Cooray, M. E. Dickinson, J. S. Dunlop, G. G. Fazio, J. P. Gardner, E. Gawiser, N. A. Grogin, N. P. Hathi, L.-T. Hsu, K.-S. Lee, R. A. Lucas, B. Mobasher, K. Nandra, J. A. Newman, and A. van der Wel. CANDELS Multi-wavelength Catalogs: Source Detection and Photometry in the GOODS-South Field. *apjs*, 207:24, August 2013.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[26] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[27] Sepp Hochreiter and Jrgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[28] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *CoRR*, abs/1711.03213, 2017.

[29] Ehsan Hosseini-Asl, Yingbo Zhou, Caiming Xiong, and Richard Socher. Augmented cyclic adversarial learning for domain adaptation. *CoRR*, abs/1807.00374, 2018.

[30] Lanqing Hu, Meina Kan, Shiguang Shan, and Xilin Chen. Duplex generative adversarial network for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[32] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. *CoRR*, abs/1611.07004, 2016.

[33] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. *arXiv e-prints*, page arXiv:1611.01144, Nov 2016.

[34] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: A generative approach to clustering. *CoRR*, abs/1611.05148, 2016.

[35] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv e-prints*, page arXiv:1401.4082, Jan 2014.

[36] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[37] Guoliang Kang, Lu Jiang, Yi Yang, and Alexander G. Hauptmann. Contrastive adaptation network for unsupervised domain adaptation. *CoRR*, abs/1901.00976, 2019.

[38] J. S. Kartaltepe, M. Mozena, D. Kocevski, D. H. McIntosh, J. Lotz, E. F. Bell, S. Faber, H. Ferguson, D. Koo, R. Bassett, M. Bernyk, K. Blancato, F. Bournaud, P. Cassata, M. Castellano, E. Cheung, C. J. Conselice, D. Croton, T. Dahlen, D. F. de Mello, L. DeGroot, J. Donley, J. Guedes, N. Grogin, N. Hathi, M. Hilton, B. Hollon, A. Koekemoer, N. Liu, R. A. Lucas, M. Martig, E. McGrath, C. McPartland, B. Mobasher, A. Morlock, E. O'Leary, M. Peth, J. Pforr, A. Pillepich, D. Rosario, E. Soto, A. Straughn, O. Telford, B. Sunnquist, J. Trump, B. Weiner, S. Wuyts, H. Inami, S. Kassin, C. Lani, G. B. Poole, and Z. Rizer. CANDELS Visual Classifications: Scheme, Data Release, and First Results. *apjs*, 221:11, November 2015.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[40] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *arXiv e-prints*, page arXiv:1406.5298, Jun 2014.

[41] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv e-prints*, page arXiv:1312.6114, Dec 2013.

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International*

Conference on Neural Information Processing Systems - Volume 1, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[43] 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.

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

[45] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017.

[46] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 97–105. JMLR.org, 2015.

[47] Mingsheng Long, ZHANGJIE CAO, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1640–1650. Curran Associates, Inc., 2018.

[48] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2208–2217. JMLR.org, 2017.

[49] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary Deep Generative Models. *arXiv e-prints*, page arXiv:1602.05473, Feb 2016.

[50] A. Molino, N. Benítez, B. Ascaso, D. Coe, M. Postman, S. Jouvel, O. Host, O. Lahav, S. Seitz, E. Medezinski, P. Rosati, W. Schoenell, A. Koekemoer, Y. Jimenez-Teja, T. Broadhurst, P. Melchior, I. Balestra, M. Bartelmann, R. Bouwens, L. Bradley, N. Czakon, M. Donahue, H. Ford, O. Graur, G. Graves, C. Grillo, L. Infante, S. W. Jha, D. Kelson, R. Lazkoz, D. Lemze, D. Maoz, A. Mercurio, M. Meneghetti, J. Merten, L. Moustakas, M. Nonino, S. Orgaz, A. Riess, S. Rodney, J. Sayers, K. Umetsu, W. Zheng, and A. Zitrin. CLASH: accurate photometric redshifts with 14 HST bands in massive galaxy cluster cores. *mnras*, 2017.

[51] Saeid Motiian, Quinn Jones, Seyed Iranmanesh, and Gianfranco Doretto. Few-shot adversarial domain adaptation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6670–6680. Curran Associates, Inc., 2017.

[52] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. *CoRR*, abs/1709.10190, 2017.

[53] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cirean, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347, Nov 2011.

[54] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[55] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 01 2011.

[56] Augustus Odena. Semi-Supervised Learning with Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1606.01583, Jun 2016.

[57] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1717–1724, June 2014.

[58] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010.

[59] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359, 2010.

[60] Xingchao Peng and Kate Saenko. Synthetic to real adaptation with deep generative correlation alignment networks. *CoRR*, abs/1701.05524, 2017.

[61] Manuel Pérez-Carrasco, Guillermo Cabrera-Vives, Monserrat Martinez-Marín, Pierluigi Cerulo, Ricardo Demarco, Pavlos Protopapas, Julio Godoy, and Marc

Huertas-Company. Multiband galaxy morphologies for clash: a convolutional neural network transferred from candels. *arXiv preprint arXiv:1810.07857*, 2018.

[62] M. Postman, D. Coe, N. Benítez, L. Bradley, T. Broadhurst, M. Donahue, H. Ford, O. Graur, G. Graves, S. Jouvel, A. Koekemoer, D. Lemze, E. Medezinski, A. Molino, L. Moustakas, S. Ogaz, A. Riess, S. Rodney, P. Rosati, K. Umetsu, W. Zheng, A. Zitrin, M. Bartelmann, R. Bouwens, N. Czakon, S. Golwala, O. Host, L. Infante, S. Jha, Y. Jimenez-Teja, D. Kelson, O. Lahav, R. Lazkoz, D. Maoz, C. McCully, P. Melchior, M. Meneghetti, J. Merten, J. Moustakas, M. Nonino, B. Patel, E. Regös, J. Sayers, S. Seitz, and A. Van der Wel. The Cluster Lensing and Supernova Survey with Hubble: An Overview. , 199:25, April 2012.

[63] M. Postman, M. Franx, N. J. G. Cross, B. Holden, H. C. Ford, G. D. Illingworth, T. Goto, R. Demarco, P. Rosati, J. P. Blakeslee, K.-V. Tran, N. Benítez, M. Clampin, G. F. Hartig, N. Homeier, D. R. Ardila, F. Bartko, R. J. Bouwens, L. D. Bradley, T. J. Broadhurst, R. A. Brown, C. J. Burrows, E. S. Cheng, P. D. Feldman, D. A. Golimowski, C. Gronwall, L. Infante, R. A. Kimble, J. E. Krist, M. P. Lesser, A. R. Martel, S. Mei, F. Menanteau, G. R. Meurer, G. K. Miley, V. Motta, M. Sirianni, W. B. Sparks, H. D. Tran, Z. I. Tsvetanov, R. L. White, and W. Zheng. The Morphology-Density Relation in z ˜ 1 Clusters. *apj*, 623:721–741, April 2005.

[64] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder network. *CoRR*, abs/1507.02672, 2015.

[65] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Ann. Math. Statist.*, 22(3):400–407, 09 1951.

[66] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[67] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[68] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015.

[69] Paolo Russo, Fabio Maria Carlucci, Tatiana Tommasi, and Barbara Caputo. From source to target and back: symmetric bi-directional adaptive GAN. *CoRR*, abs/1705.08824, 2017.

[70] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 213–226, Berlin, Heidelberg, 2010. Springer-Verlag.

[71] Kuniaki Saito, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. Semi-supervised Domain Adaptation via Minimax Entropy. *arXiv e-prints*, page arXiv:1904.06487, Apr 2019.

[72] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.

[73] Cicero Nogueira dos Santos, Kahini Wadhawan, and Bowen Zhou. Learning loss functions for semi-supervised learning via discriminative adversarial networks. *arXiv preprint arXiv:1707.02198*, 2017.

[74] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018.

[75] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[76] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[77] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. *CoRR*, abs/1511.05547, 2015.

[78] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. *CoRR*, abs/1607.01719, 2016.

[79] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[80] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. *CoRR*, abs/1808.01974, 2018.

[81] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. *CoRR*, abs/1510.02192, 2015.

[82] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *CoRR*, abs/1702.05464, 2017.

[83] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[84] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. 2008.

[85] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[86] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *CoRR*, abs/1802.03601, 2018.

[87] Hongliang Yan, Yukang Ding, Peihua Li, Qilong Wang, Yong Xu, and Wangmeng Zuo. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. *CoRR*, abs/1705.00609, 2017.

[88] Jun Yang, Rong Yan, and Alexander G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *Proceedings of the 15th ACM International Conference on Multimedia*, MM '07, pages 188–197, New York, NY, USA, 2007. ACM.

[89] T. Yao, , C. Ngo, and and. Semi-supervised domain adaptation with subspace learning for visual recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2142–2150, June 2015.

[90] Y. Yao and G. Doretto. Boosting for transfer learning with multiple sources. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1855–1862, June 2010.

[91] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014.

[92] Guoqiang Peter Zhang. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 30(4):451–462, 2000.

[93] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017.

[94] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: Transfer learning with deep autoencoders. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4119–4125. AAAI Press, 2015.

[95] Han Zou, Yuxun Zhou, Jianfei Yang, Huihan Liu, Hari Prasanna Das, and Costas Spanos. Consensus adversarial domain adaptation. In *AAAI Conference on Artificial Intelligence 33*, 01 2019.

Chapter 6

Suplementary Material

6.1 Apendix A

In this appendix we provide complete derivations for Equation 3.10. This Equation can be decomposed in the following terms:

$$\begin{aligned} D_{KL}\Big[q_\phi(z|x)||p_\theta(z)\Big] &= \int_z q_\phi(z|x)\log\frac{q_\phi(z|x)}{p_\theta(z)}dz \\ &= \int_z q_\phi(z|x)\log q_\phi(z|x)dz - \int_z q_\phi(z|x)\log p_\theta(z)dz \\ &= \int_z \mathcal{N}(z|\tilde{\mu},\tilde{\sigma}^2 I)\log\mathcal{N}(z|\tilde{\mu},\tilde{\sigma}^2 I)dz - \int_z \mathcal{N}(z|\tilde{\mu},\tilde{\sigma}^2 I)\log\mathcal{N}(z|\mu,\sigma^2 I)dz \end{aligned} \tag{6.1}$$

Following the derivations of Lemma 1 in appendix B of Jiang et al. 2016 [34], we can compute an analytical solution for the first term as follows:

$$\int_z \mathcal{N}(z|\tilde{\mu},\tilde{\sigma}^2 I)\log\mathcal{N}(z|\mu,\sigma^2 I)dz = \frac{1}{2}\sum_{j=1}^{J} -\log 2\pi\sigma_j^2 - \frac{\tilde{\sigma_j^2}}{\sigma_j^2} - \frac{(\tilde{\mu_j}-\mu_i)^2}{\sigma_j^2} \tag{6.2}$$

Using the same Lemma, the analytical solution for the second term can be computed as follows:

$$\int_z \mathcal{N}(z|\tilde{\mu},\tilde{\sigma}^2 I)\log\mathcal{N}(z|\tilde{\mu},\tilde{\sigma}^2 I)dz = -\frac{J}{2}\log 2\pi - \frac{1}{2}\sum_{j=1}^{J}(1+\log\tilde{\sigma}_j^2) \tag{6.3}$$

By combining Equation 6.1 and 6.1, we obtain the solution introduced in Equation 3.10.

6.2 Apendix B

In this appendix we provide complete derivations for Equation 3.7. This Equation can be decomposed in two terms as follows:

$$
\begin{aligned}
D_{KL}(q_\psi\big(z,y|x)||p_\theta(z,y)\big) &= \mathbb{E}_{q_\psi(z,y|x)}\left[\log \frac{q_\psi(z,y|x)}{p_\theta(y,z)}\right] \\
&= E_{q_\psi(z,y|x)}\left[\log \frac{q_\psi(z|x)}{p_\theta(z|y)}\right] + E_{q_\psi(z,y|x)}\left[\log \frac{q_\psi(y|z)}{p_\theta(y)}\right] \\
&= E_{q_\psi(y|z)}\,\mathbb{E}_{q_\psi(z|x)}\left[\log \frac{q_\psi(z|x)}{p_\theta(z|y)}\right] + E_{q_\psi(y|z)}\,\mathbb{E}_{q_\psi(z|x)}\left[\log \frac{q_\psi(y|z)}{p_\theta(y)}\right] \\
&= \mathbb{E}_{q_\psi(y|z)}\Big[D_{KL}(q_\psi(z|x)||p_\theta(z|y))\Big] + \mathbb{E}_{q_\psi(z|x)}\Big[D_{KL}(q_\psi(y)||p_\theta(y))\Big]
\end{aligned}
\tag{6.4}
$$

This Equation can be further optimized following the procedures introduced in Equation 3.13 and Equation 3.15.