Université de Montréal

Representation learning in unsupervised domain translation

par

Samuel Lavoie-Marchildon

Département d'informatique et de recherche opérationelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

31 décembre 2019

Résumé

Ce mémoire s'adresse au problème de traduction de domaine non-supervisée. La traduction non-supervisée cherche à traduire un domaine, le domaine source, à un domaine cible sans supervision. Nous étudions d'abord le problème en utilisant le formalisme du transport optimal. Dans un second temps, nous étudions le problème de transfert de sémantique à haut niveau dans les images en utilisant les avancés en apprentissage de représentations et de transfert d'apprentissages développés dans la communauté d'apprentissage profond.

Le premier chapitre est dévoué à couvrir les bases des concepts utilisés dans ce travail. Nous décrivons d'abord l'apprentissage de représentation en incluant la description de réseaux de neurones et de l'apprentissage supervisé et non supervisé. Ensuite, nous introduisons les modèles génératifs et le transport optimal. Nous terminons avec des notions pertinentes sur le transfert d'apprentissages qui seront utiles pour le chapitre 3.

Le deuxième chapitre présente *Neural Wasserstein Flow*. Dans ce travail, nous construisons sur la théorie du transport optimal et démontrons que les réseaux de neurones peuvent être utilisés pour apprendre des barycentres de Wasserstein. De plus, nous montrons que les réseaux de neurones peuvent amortir n'importe quel barycentre, permettant d'apprendre une interpolation continue. Nous montrons aussi comment utiliser ces concepts dans le cadre des modèles génératifs. Finalement, nous montrons que notre approche permet d'interpoler des formes et des couleurs.

Dans le troisième chapitre, nous nous attaquons au problème de transfert de sémantique haut niveau dans les images. Nous montrons que ceci peut être obtenu simplement avec un GAN conditionné sur la représentation apprise par un réseau de neurone. Nous montrons aussi comment ce processus peut être rendu non-supervisé si la représentation apprise est un regroupement. Finalement, nous montrons que notre approche fonctionne sur la tâche de transfert de MNIST à SVHN.

Nous concluons en mettant en relation les deux contributions et proposons des travaux futures dans cette direction.

Mot-clés Apprentissage profond, transport optimal, transfert d'apprentissage, traduction d'images non-supervisée, traduction de domaine non-supervisée.

Abstract

This thesis is concerned with the problem of unsupervised domain translation. Unsupervised domain translation is the task of transferring one domain, the source domain, to a target domain. We first study this problem using the formalism of optimal transport. Next, we study the problem of high-level semantic image to image translation using advances in representation learning and transfer learning.

The first chapter is devoted to reviewing the background concepts used in this work. We first describe representation learning including a description of neural networks and supervised and unsupervised representation learning. We then introduce generative models and optimal transport. We finish with the relevant notions of transfer learning that will be used in chapter 3.

The second chapter presents Neural Wasserstein Flow. In this work, we build on the theory of optimal transport and show that deep neural networks can be used to learn a Wasserstein barycenter of distributions. We further show how a neural network can amortize any barycenter yielding a continuous interpolation. We also show how this idea can be used in the generative model framework. Finally, we show results on shape interpolation and colour interpolation.

In the third chapter, we tackle the task of high level semantic image to image translation. We show that high level semantic image to image translation can be achieved by simply learning a conditional GAN with the representation learned from a neural network. We further show that we can make this process unsupervised if the representation learning is a clustering. Finally, we show that our approach works on the task of MNIST to SVHN.

We conclude by relating the two contributions and propose future work in that direction.

Keywords Deep learning, optimal transport, transfer learning, unsupervised image to image translation, unsupervised domain translation.

Contents

List of Tables

List of Figures

Acronyms, abbreviations and notations

GAN	Generative Adversarial Network
WGAN	Wasserstein Generative Adversarial Network
VAE	Variational auto-encoder
MNIST	Modified National Institute of Standards and Technology database, a dataset of handwritten digit
SVHN	Street View House number, a dataset of digits of house number
XOR	Exclusive or
ReLU	Rectified Linear Unit
ELU	Exponential Linear Unit
Sigmoid	Logistic function
$\mathbb{I}$	Indicator function

$\mathbf{1}$	One-hot vector
$\boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{z}\ \boldsymbol{\lambda}\ \boldsymbol{...}$	A vector in $\mathbb{R}^d$
$\mathcal{L}$	Objective function
$\mathcal{T}\ \mathcal{U}$	Familly of functions
$\mathcal{X}\ \mathcal{Y}\ \mathcal{Z}\ \mathcal{H}$	Topological spaces
π	Transport plan
Π	Family of transport plan
$\lvert \cdot \rvert$	Cardinality
$\lvert\lvert \cdot \rvert\rvert_p$	p-norm
Δ^k	k probability simplex
D_{KL}	KL divergence
$\#$	Push-forward operator
$\circ$	Composition of function operator

d, f, g, ψ, ϕ	Parametric functions
∇	Gradiant operator
∂	Partial derivative operator

Acknowledgements

This project taught me a lot about the discipline of research, its purpose and its craft. I started this project thinking naively that research was simply solving open problems. Firstly, I learned that solving open problems is far from simple and requires a lot of dedication and effort. Secondly, I learned that research is much more than solving problems. In research, you need to communicate your ideas and listen to others. You need to be careful about every details. And, you need to contribute to the community.

I had the chance to meet a lot of people that helped me improve in different aspects of research. I would like to first mention Aaron who advised me throughout this journey. I thank you for sharing your experience with me and guiding me in my research endeavours. It was truly enjoyable working with you.

I would also like to acknowledge everyone who collaborated with me on different projects. This includes Devon, Joseph, Francis, Sébastien, Jacob, Amjad and Faruk. I am grateful to have had the chance to work with you.

Next, I wish to mention the Mila community. It's truly amazing to have the chance to exchange daily with passionate researchers, whether it is informally at the coffee machine or in a reading group. In particular, I would like to mention the following colleagues and friends that I had the pleasure to meet during my journey: Joseph, Chin-Wei, Philippe, Francis, Devon, Tristan, Simon, Shawn, Jae Hyun, Evan, Ankesh, David, Julien, Paul, Akram, Mark, Nicolas, Gabriel, Rémi, Martin and Olexa.

Finalement, j'aimerais remercier mes proches pour leur support inconditionnel tout au long de ce parcours. Merci Catherine, Marie-Hélène et Ghislaine.

Introduction

A *good representation* of the data and the problem is at the core of solving complex tasks. For example, the problem of classifying real images can be extremely hard when we consider it as simply fitting a linear model on the values of the pixels. Hence, for a long time, the computer vision community worked hard at designing good representations of that data or the problem. Techniques for edges detection and clever heuristics for classifying images were invented, without a considerable and generalizable breakthrough. Such breakthrough came after the convolution was successfully applied in a *deep neural network* trained to classify images. This convolutional neural network learns to capture relevant features in the images so that it can accurately predict the content of the image. In this example, the representation is learned through the composition of linear and non-linear functions. This example was one of the realizations that stemmed the resurgence of representation learning using deep neural networks. In essence, we can summarize the task of representation learning as learning a representation such that it can solve a downstream task of interest.

In this thesis, we study how the recent advances in deep representation learning can be applied to the task of unsupervised domain translation. This task was initially formalized and studied by the mathematician Monge (Monge 1781). In his work, he was interested in the optimal transportation of masses. Since then, the field of optimal transport has greatly evolved and been applied to various problems.

In the first chapter, we will review current the literature in deep representation learning, optimal transport and domain translation that is relevant to the sequel of this document. In the second chapter, we present a framework for learning a continuous flow on the Wasserstein geodesic using deep neural networks. Finally, in the third chapter, we explore how representation learning and transfer learning can be applied to high-level semantic domain translation

Chapter 1

Background

In this chapter, we present different strategies for learning a representation. Before presenting any learning paradigm, we introduce neural networks and a general technique to train them. Then, we present two supervised representation learning algorithms: the classification and the regression. We follow with two unsupervised representation learning paradigms: clustering and latent factors analysis. Then, we reverse the problem and consider generative algorithms that allow us to implicitly sample from the distribution of the data. We will also introduce some concepts of optimal transport that will be useful in chapter 2. We finish with a discussion on transfer learning; more specifically we discuss domain adaptation and image to image translation.

1.1. Representation Learning

Machine learning is a tool for solving problems for which we have data, but no analytic solution. This paradigm becomes increasingly interesting as data is more accessible and the problems more complex. In essence, machine learning aims at solving a task by fitting a parameterized function f in order to optimize an objective function $\mathcal{L}$.

Sometimes, the optimal solution involves a non-linear relation of different features of the input representations. A popular toy example is the XOR problem. Given two binary features $\boldsymbol{x} \in \{0,1\}^2$, apply the XOR function to $\boldsymbol{x}$. The XOR function is defined as:

$$\text{XOR}(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{otherwise.} \end{cases}$$

It can be verified that such function requires a non-linear relationship between the inputs – x_1 and x_2 – and the output. Hence, for any $\mathcal{L}$, a linear function on the euclidean space cannot represent the XOR function. One way to picture why a linear function cannot represent the XOR function is to represent it in space as shown in Figure 1.1. As we can see, there is not straight line that can correctly separate the space.

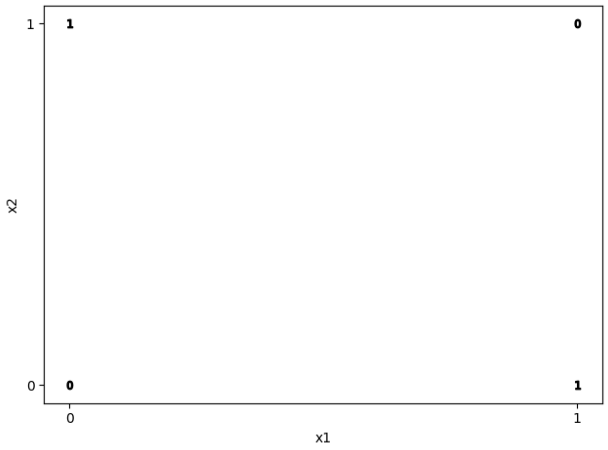

Figure 1.1. XOR function. The markers indicate the output of the function given its inputs x_1 and x_2.

Deep neural networks can learn non-linear functions of the input features by composing linear and non-linear continuous functions. It is this composition that allows them to transform the space such that it can be possible to finally solve non-linear problem with a simple linear function.

More formally, let $\boldsymbol{h}_0 = \boldsymbol{x}$ the input representation. A neural network is the composition of functions f_l. In the feedforward chain of compositions setup, we call the output of each function f_l a hidden layer computed as $\boldsymbol{h}_{l+1} = f_l(\boldsymbol{h}_l)$, for any $0 \leq l < L$.

A function $f_l : f_l^2 \circ f_l^1$ is normally a composition of two continuous functions. We will consider f_l^1 a linear function and f_l^2 a non-linear function.

$$\boldsymbol{a}_{l+1} = f_l^1(\boldsymbol{h}_l)$$

is called the pre-activation functions and is the representation obtained before applying the non-linear function.

1.1.1. Linear functions

A commonly used linear function is the affine transformation. The layer resulting from an affine transformation is usually called a linear layer. Let $W_l \in \mathbb{R}^{D_l \times D_{l+1}}$ the *weights*, which are free parameters that can be learned. It is common practice to also use biases $\boldsymbol{b}_l \in \mathbb{R}^{D_{l+1}}$, which are also free parameters. The pre-activation of a linear layer is defined as

$$\boldsymbol{a}_{l+1} = W^\top \boldsymbol{h}_l + \boldsymbol{b}_l.$$

Another commonly used linear function is the cross-correlation operation, which is related to convolution operation. The layer resulting from a convolution is usually called a convolutional layer. This operation involves repeatedly applying a kernel of free parameters to every positions of a representation.

The idea of applying the same kernel at every feature of a layer, or channel for 2D inputs, is reminiscent of the idea of weights sharing. This has the advantage of having to store fewer parameters. But the real strength of the convolution is due to the property of equivariance on the translation. In other words, because the same kernel is applied at every location of an image, the location of an object in the image will affect the pre-activation representation by shifting its position accordingly. Finally, weights sharing also has a regularization effect.

1.1.2. Non-linear functions

The non-linear activation functions are usually simple continuous functions applied element-wise to a pre-activation. However, composing these simple functions enable the network to represent any functions given enough capacity. Some examples of such functions are the the Leaky-ReLU:

$$\text{Leaky-ReLU}(\boldsymbol{x}, \lambda) = \max(0, \lambda\boldsymbol{x})$$

where $\lambda > 0$. The ReLU (Nair and Hinton 2010) is a special case where $\lambda = 1$. The ELU (Clevert, Unterthiner, and Hochreiter 2015):

$$\text{ELU}(\boldsymbol{x}, \lambda) = \max(0, \boldsymbol{x}) + \min(0, \lambda(\exp(\boldsymbol{x}) - 1))$$

and the Sigmoid (Han and Moraga 1995)

$$\text{Sigmoid}(\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{x})}.$$

1.1.3. Training

There is currently no closed-form solution known for finding the global optimum of a neural network. Hence, in practice, it is common to use gradient-based technique to find an optimum. However, because a neural network can be non-convex, we don't have any guarantee that the optimum found is the global optimum.

The procedure for learning the parameters of a neural network relies on basic calculus. We want to optimize the free-parameters of a function $f \in F$ in a family of parametrizable functions F in order to minimize (or maximize) an objective function $\mathcal{L}(f)$. F is determined by the architectural choices, e.g. the number of layers, the width of each layers and the non-linearities.

For now, we will consider that we want to minimize $\mathcal{L}(f)$. But, the procedure to maximize is almost identical.

Finding the minimum of $\mathcal{L}$ involves finding a stationary point which is defined as follow

$$\nabla_f \mathcal{L}(f) = 0.$$

As we will see in the sequel, $\mathcal{L}$ are continuous functions which have a derivative that we can compute. Moreover, because f is defined as a composition of continuous functions with derivative defined almost everywhere, we can use the chain rule to backpropagate the gradient signal to minimize $\mathcal{L}$ back to every parameters. The backpropagation is defined as

$$\frac{\partial \mathcal{L}}{\partial f_l} = \sum_{j=l}^{L-1} \frac{\partial \mathcal{L}}{\partial f_{j+1}} \frac{\partial f_{j+1}}{\partial f_j}$$

where f^i is the i^{th} layer and L is the total number of layers.

Using this signal, it is possible to update every parameters of a layer f^i using different algorithms. Probably, the most common one is stochastic gradient descent (Kunnumkal and Topaloglu 2009)

$$f_l \leftarrow f_l - \alpha \frac{\partial \mathcal{L}}{\partial f_l} \tag{1.1.1}$$

where $\alpha > 0$ is called the *learning rate* and is used to avoid that the optimization diverge. Other optimization algorithms involve adding momentum (Rumelhart, Hinton, and Williams 1988) or running average (Diederik P. Kingma and Ba 2014) to SGD.

1.1.4. Supervised learning

In supervised learning, we assume that we have access to supervised annotation $l \sim L$ for each $\boldsymbol{x}$. Depending on the task, l can be discrete or continuous. The objective of supervised learning is to learn a parameterized function f to predict l given $\boldsymbol{x}$

1.1.4.1. *Classification*

A classical supervised learning task is classification. Practical applications of classification are numerous and can range from predicting the content of a natural image to predicting the sentiment in a sentence. In this context, a discrete label l is assigned to every sample $\boldsymbol{x}$. The task is to learn a function f to predict the label l of a sample $\boldsymbol{x}$. It does so by minimizing the error between $f(\boldsymbol{x})$ and the true l

$$\mathcal{L}(f) = \mathcal{R}(f(\boldsymbol{x}), l).$$

For a classification task, $\mathcal{R}$ is usually defined as the binary cross entropy when $|L| = 2$

$$\mathcal{L}(f) = -l \log(f(\boldsymbol{x})) - (1 - l) \log(1 - f(\boldsymbol{x}))$$

and the categorical cross entropy when $|L| > 2$

$$\mathcal{L}(f) = -\sum_{i=1}^{|L|} \mathbf{1}_{l=i} \log f(\boldsymbol{x})_i.$$

In the categorical cross entropy, we usually represent the predicted label $f(\boldsymbol{x})$ as a normalized vector where $f(\boldsymbol{x})_i \geq 0, \forall i$ and $\sum_{i=1}^{|L|} f(\boldsymbol{x})_i = 1$. Another subtlety is the use of the one-hot encoding $\mathbf{1}_{i=l}$ which is defined as the vector containing a 1 at index l and 0 everywhere else.

1.1.4.2. *Regression*

Regression is a supervised learning task where l is continuous rather than discrete. Some practical applications of the regression include predicting the future value of a time-serie or the price of an asset given different factors. The learning algorithm used to learn a regression is very similar to the one used to learn a classification. We have a mapping $f : \boldsymbol{x} \to \mathbb{R}$ and the objective is to minimize the risk $\mathcal{R}$. However, the risk functions is now usually defined as a norm $||\cdot||_p^p$ instead of the cross-entropy.

$$\mathcal{L}(f) = ||l - f(\boldsymbol{x})||_p^p.$$

In practice, the 1-norm or the 2-norm squared is almost always used.

1.1.5. Unsupervised learning

Unsupervised learning is the setup where no extrinsic annotations are available. This setup is increasingly interesting as more data are available. Two unsupervised learning paradigms that have seen a surge recently, and that will be at the core of this thesis, are unsupervised learning representation of the data and learning generative models of the data. We finish this section on representation learning with a presentation of two unsupervised representation learning paradigms: clustering and latent factor analysis.

1.1.5.1. *Clustering*

Clustering is a canonical example of unsupervised representation learning. In its simplest case, the main assumption of clustering is that the data can be separated into K discrete clusters. While, in theory, K can be unbounded and parameterized by a Dirichlet process (Murphy 2012), we will only consider the simplest case in this work. There exist a number of algorithms for finding clusters, each of them making different assumptions. We will review K-Means (Lloyd 2006) in this section.

K-Means main assumption is that the data is linearly separated in clusters. Intuitively, it finds K clusters where points in a cluster have a small distance in comparison to points in a different cluster. More precisely, let $\boldsymbol{x} \in \mathbb{R}^D$ with $D \geq 1$, $k \in \mathbb{N}$ and $\boldsymbol{\mu}_k \in \mathbb{R}^D$ the *prototype* associated to the k^{th} cluster. In K-Means, a prototype $\boldsymbol{\mu}_k$ is the the average point of a cluster. Essentially, K-Means tries to find K prototypes and assign every $\boldsymbol{x}$ to a prototype such that the objective

$$\mathcal{L}(\{\boldsymbol{\mu}_k\}_{k=1}^K) = \sum_{i=1}^{|X|} \sum_{k=1}^{K} S_{i,k} D(\boldsymbol{x}_i, \boldsymbol{\mu}_k)$$

is minimized, S is the indicator function and D is the distortion. In practice, D is defined as the 2-norm squared.

We can see that a two-stages optimization process is involved. One stage is the cluster assignment where $\mathcal{L}$ is minimized with respect to S. The second stage involves minimizing $\mathcal{L}$ with respect to $\{\boldsymbol{\mu}_k\}_{k=1}^K$. The optimization alternates each stage until convergence is obtained. A common algorithm for solving this optimization is the Expectation-Maximization algorithm.

The first step consist of assigning one of K cluster to every $\boldsymbol{x}$ as follow

$$S_{i,k} = \begin{cases} 1 & \text{if } k = \arg\min_j D(\boldsymbol{x}_i, \boldsymbol{\mu}_j) \\ 0 & \text{otherwise.} \end{cases}$$

The second step consist of minimizing $\mathcal{L}$ with respect to $\boldsymbol{\mu}_k$. When D is the 2-norm, a solution for $\mathcal{L}$ exists in closed form:

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{i=1}^{|X|} S_{i,k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_i||_2^2 = 0 \implies 2 \sum_{i=1}^{|X|} S_{i,k}(\boldsymbol{x}_i - \boldsymbol{\mu}_k) = 0$$

which we can solve to obtain $\boldsymbol{\mu}_k$ in closed form

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{|X|} S_{i,k} \boldsymbol{x}_i}{\sum_{i=1}^{|X|} S_{i,k}}.$$

1.1.5.2. *Latent factor analysis*

Latent factor analysis aims at learning a continuous representation of the data on a lower manifold. For example, PCA (Pearson 1901) aims at learning a linear mapping $W \in \mathbb{R}^{D \times (D-M)}$, with $0 \leq M < D$, that maximize the variance, or equivalently minimize the distortion.

A more general framework for learning continuous latent factors is the auto-encoder (Kramer 1991) which we will now describe.

Auto-Encoder

In complex datasets, semantic can be encoded as a non-linear combination of the input features. For example, in real images, the category is usually defined as the combination of high-level features, like shapes and edges, which are in turn defined as a composition of pixels. Thus, it is necessary to consider a non-linear mapping to extract those non-linear features.

One such algorithm for unsupervised learning of a representation, using non-linear function, is the Auto-Encoder. In its simplest form, the auto-encoder learns a mapping that minimizes the reconstruction error of a sample $\boldsymbol{x} \in \mathcal{X}$ and its mapped representation $\tilde{\boldsymbol{x}} = f(\boldsymbol{x})$

$$\mathcal{L}(f) = ||\boldsymbol{x} - f(\boldsymbol{x})||_2^2.$$

The mapping is a composition of an encoder and a decoder: $f = d \circ e$. If d and e are linear mapping, the auto-encoder recovers PCA up to a rotation of the latent space.

However, this added flexibility comes at the cost that the representation learned might be meaningless. Given enough capacity, the mapping can simply learn the identity. This is this reason that motivates the regularization of auto-encoders.

One such regularization is defining a transformation $t : \mathcal{X} \to \mathcal{X}$ and minimize the following objective

$$\mathcal{L}(f) = ||\boldsymbol{x} - f(t(\boldsymbol{x}))||_2^2.$$

This regularization forces the mapping to be invariant to the transformation t. A transformation t often used is

$$t(\boldsymbol{x}) := \boldsymbol{x} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon}$ is a noise usually sampled from the isotropic Gaussian $\mathcal{N}(0, 1)$. This regularized auto-encoder is called the Denoising Auto-Encoder (Vincent et al. 2008), because it learns to remove Gaussian noise from the input.

1.2. Generative model

In general, we only have access to a finite set of samples $\boldsymbol{x}$ from the true density function $\mathbb{P}_x$. However, having access to $\mathbb{P}_x$ could be interesting for many practical reasons. For example, anomaly detection is interested in detecting if a sample $\hat{\boldsymbol{x}}$ comes from $\mathbb{P}_x$. We could also be interested in sampling data from $\mathbb{P}_x$ directly. Thus, the objective of generative models is to learn $\mathbb{P}_x$ using only the samples available.

A common strategy used in generative models is to learn a mapping $g : \boldsymbol{x} \to \boldsymbol{x}$, such that $\mathbb{P}_x = g\#\mathbb{P}_z$[1]. The idea is to choose a probability distribution $\mathbb{P}_z$ that we can sample from, for example, the isotropic Gaussian distribution, and to learn a function g that transforms $\mathbb{P}_z$ to the desired distribution $\mathbb{P}_x$. A number of frameworks build on this strategy. For example, the VAE (Diederik P Kingma and Welling 2013) framework learns an encoder that maps to a parametrized prior distribution $\mathbb{P}_z$, while also learning a generative model $\mathbb{P}_{x|z}$ using a decoder. Hence, the VAE is an example of a model that acts as a representation learning and a generative model at the same time.

The generative adversarial network, which we will go in more detail in the next section, learns a mapping $g : \boldsymbol{z} \to \boldsymbol{x}$ by solving a min-max optimization problem between two functions d and g, which they train simultaneously.

GANs were initially introduced as an adversarial game where a generator g tries to fool a discriminator d, which in turn tries to distinguish true samples from fake samples. More precisely, let $\boldsymbol{x} \sim \mathcal{X}$ be the usual measurable data space which induces a probability measure $\mathbb{P}_x$, $d : \boldsymbol{x} \to [0, 1]$ and $g : \boldsymbol{z} \to \boldsymbol{x}$. We have $\boldsymbol{z} \sim \mathbb{P}_z$ usually taken to be the isotropic Gaussian

[1] $g\#\mathbb{P}_z$ is the pushforward defining the measure obtained by transforming $\mathbb{P}_z$ using g

distribution. The objective function is the following

$$\mathcal{L}(d, g) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x} \log d(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z} \log(1 - d(g(\boldsymbol{z})))$$

and optimized as a min-max game

$$\min_g \max_d \mathcal{L}(d, g).$$

In practice, d and g are alternatively updated. In this particular iteration of the algorithm, it is important that d does not reach the optimum until the end of the training. The reason is that, as we will see in the following iteration, d converges to an estimation of the Jensen Shannon Divergence. However, the Jensen Shannon Divergence is ill-defined for two distributions that do not share the same support. This is the case of $\mathbb{P}_x$ and $g\#\mathbb{P}_z$, which we will denote $\mathbb{P}_{x'}$, at the beginning of the training.

$$\begin{aligned}
&\max_d \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x} \log d(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_{x'}} \log(1 - d(\boldsymbol{x})) \\
&\quad\implies \frac{\partial}{\partial d} \int_{\boldsymbol{x}} \mathbb{P}_x(\boldsymbol{x}) \log d(\boldsymbol{x}) + \mathbb{P}_{x'}(\boldsymbol{x}) \log(1 - d(\boldsymbol{x}))d\boldsymbol{x} = 0 \\
&\quad\implies \int_{\boldsymbol{x}} \frac{\mathbb{P}_x(\boldsymbol{x})}{d(\boldsymbol{x})} - \frac{\mathbb{P}_{x'}(\boldsymbol{x})}{1 - d(\boldsymbol{x})} d\boldsymbol{x} = 0 \\
&\quad\implies \int_{\boldsymbol{x}} \mathbb{P}_x(\boldsymbol{x})(1 - d(\boldsymbol{x})) - \mathbb{P}_{x'}(\boldsymbol{x})d(\boldsymbol{x})d\boldsymbol{x} = 0
\end{aligned}$$

which attains a stationary point at

$$d^*(\boldsymbol{x}) = \frac{\mathbb{P}_x(\boldsymbol{x})}{\mathbb{P}_x(\boldsymbol{x}) + \mathbb{P}_{x'}(\boldsymbol{x})}. \tag{1.2.1}$$

Substituting our optimal discriminator in equation 1.2.1, we obtain

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x(X)} \log d^*(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_{x'}} \log(1 - d^*(\boldsymbol{x})) \\
&= \int_{\boldsymbol{x}} \mathbb{P}_x(\boldsymbol{x}) \log \left(\frac{\mathbb{P}_x(\boldsymbol{x})}{\mathbb{P}_x(\boldsymbol{x}) + \mathbb{P}_{x'}(\boldsymbol{x})} \right) d\boldsymbol{x} + \int_{\boldsymbol{x}} \mathbb{P}_x(\boldsymbol{x}) \log \left(\frac{\mathbb{P}_{x'}(\boldsymbol{x})}{\mathbb{P}_{x'}(\boldsymbol{x}) + \mathbb{P}_x(\boldsymbol{x})} \right) d\boldsymbol{x}.
\end{aligned}$$

Which is the definition of the Jensen Shannon Divergence.

This framework has inspired many follow-up work that derives an estimation of different metrics which they try to minimize with g. Different metrics have different properties. For example, the Wasserstein distance, which we will take some time to derive, is defined even when two distributions do not share the same support. In the following subsection, we will describe the properties of the Wasserstein distance that will be useful later. WGAN proposes to learn a parametric function $f : \boldsymbol{x} \to \mathbb{R}$ to estimate the Earth-mover distance.

$$\max_{f:\text{Lipschitz-1}} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x} f(\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\sim g\#\mathbb{P}_z} f(\boldsymbol{x}). \tag{1.2.2}$$

In the next section, we introduce more notions of optimal transport. But the reader who wants to learn more can consult the book Computation Optimal Transport (Peyré and Cuturi 2019), which is aimed at readers with a computer science background.

1.3. Optimal transport

In our everyday life, we often wonder what is the optimal way to do something. For example, what is the path from my house to work that will minimize my time spent walking? In logistics and economics, the concept of transportation and minimizing cost is at the center of their interests. Even in physics, it can be observed that particles follow the path of *minimum effort* (McCann 1997). Optimal transport, initiated by the work of Monge (Monge 1781), offers a powerful yet simple formalism for defining the optimal way of transporting distributions. The field of optimal transport has been widely studied and is still being studied to this day as more connections with other formalism and more applications are being found.

In this work, we explored the application of optimal transport for the problem of unsupervised domain translation. In this section, we present a primer on optimal transport. More precisely, we present the Monge and Kantorovich formulation. We then draw the connection between the Earth-mover distance and Wasserstein GANS. We offer more background in the second chapter where we discuss the geodesics induced by the Wasserstein distance as well as barycenters in Wasserstein space.

Monge was interested in the **mapping** of minimal cost c. In this work, we will consider mapping of probability distribution measure $\mathbb{P}_x$ and $\mathbb{P}_y$. Hence, we want a formalism for defining the cost $c : \mathcal{X} \times \mathcal{Y} \to [0, \infty]$ of moving a point in $\mathcal{X}$ to a point in $\mathcal{Y}$. Consider a mapping $t : \mathcal{X} \to \mathcal{Y}$, the Monge formulation define the optimal transport cost as

$$\inf_{t \in \mathcal{T}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_x} c(\boldsymbol{x}, t(\boldsymbol{x})) \tag{1.3.1}$$

where $\mathcal{T}$ is the family of admissible functions defined as

$$\mathcal{T} := \{t \text{ s.t. } \mathbb{P}_y = t\#\mathbb{P}_x\}. \tag{1.3.2}$$

This formulation has a solution if a deterministic mapping satisfying equation 1.3.2 exists. For example, if $\mathcal{X}$ and $\mathcal{Y}$ are continuous or are two discrete sets that have the same number of element, there exist a deterministic mapping between the two sets. However, if the two discrete sets have a different number of atoms, such a deterministic mapping does not exists.

The **Kantorovich** formulation is a relaxation of the Monge Mapping. Instead of learning a deterministic mapping, this formulation proposes to learn a probabilistic **transport plan**. The transport plan essentially allows one to probabilistically split mass, solving the issue of the Monge map. It can be observed that the Monge map is a special case of the transport

plan. The Kantorovich formulation is formalized as follow

$$\inf_{\pi \in \Pi(\mathbb{P}_x, \mathbb{P}_y)} \int_{(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{X} \times \mathcal{Y}} c(\boldsymbol{x}, \boldsymbol{y}) d\pi(\boldsymbol{x}, \boldsymbol{y}) \tag{1.3.3}$$

where Π is the family of possible transport plan defined as

$$\Pi(\mathbb{P}_x, \mathbb{P}_y) := \left\{ \pi \in \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \text{ s.t. } \int_{\boldsymbol{y}} \pi(\cdot, \boldsymbol{y}) = \mathbb{P}_x, \int_{\boldsymbol{x}} \pi(\boldsymbol{x}, \cdot) = \mathbb{P}_y \right\}.$$

We define $\mathcal{P}$ the set of probability measures on the joint space $\mathcal{X} \times \mathcal{Y}$. The transport plan can be intuitively understood as a mapping with marginal $\mathbb{P}_x$ and $\mathbb{P}_y$. Hence, the Monge problem and the Kantorovich problem simply tries to find an arrangement of each distribution that minimizes a cost.

A well studied case of equation 1.3.3 is the case when c is the $||\cdot||_p^p$ and is called the Wasserstein distance

$$\inf_{\pi \in \Pi(\mathbb{P}_x, \mathbb{P}_y)} \left(\int_{(\boldsymbol{x},\boldsymbol{y}) \sim \mathcal{X} \times \mathcal{Y}} ||\boldsymbol{x} - \boldsymbol{y}||_p^p d\pi(\boldsymbol{x}, \boldsymbol{y}) \right)^{\frac{1}{p}}. \tag{1.3.4}$$

We will come back to the Wasserstein distance later. But for now, we present the dual formulation of equation 1.3.3, which exists and can be easier to estimate in some cases.

$$\sup_{(\phi,\psi) \in \mathcal{U}(c)} \mathbb{E}_{\boldsymbol{x} \sim \mu} \phi(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{y} \sim \nu} \psi(\boldsymbol{y}) \tag{1.3.5}$$

where $\phi : \mathcal{X} \to \mathbb{R}$ and $\psi : \mathcal{Y} \to \mathbb{R}$. In the optimal transport, it is common to call these functions *Kantorovich potential*. Equivalently, in the deep learning community, it is common to call these functions *critics* (Arjovsky, Chintala, and Bottou 2017). $\mathcal{U}(c)$ is defined as the family of constrained functions defined as

$$\mathcal{U}(c) := \phi(\boldsymbol{x}) + \psi(\boldsymbol{y}) \leq c(\boldsymbol{x}, \boldsymbol{y}) \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \times \mathcal{Y}.$$

Essentially, ϕ and ψ can be any function as long as their sum is lower or equal to the cost c for all $\boldsymbol{x}$ and $\boldsymbol{y}$

For a convex cost function c, we can define ψ as a function of c and ϕ. We then call ψ the c-transform of ϕ and denote it ϕ^c. The c-transform is defined as

$$\phi^c(\boldsymbol{y}) = \inf_{\boldsymbol{x} \in \mathcal{X}} (\phi(\boldsymbol{x}) + c(\boldsymbol{x}, \boldsymbol{y})).$$

Allowing us to define equation 1.3.5 as a sup over only one critic. Using the c-transform, we can show that in the special case where c is the L_1 norm, ϕ can simply be constrained to be Lipschitz-1, recovering the earth-mover distance used in WGAN (Santambrogio 2015).

In the next section, we look at a slightly different problem where we are concerned with learning a hypothesis function which is robust to changes in distribution rather than learning a transport from one distribution to the other.

1.4. Domain adaptation

Machine learning is a powerful tool for fitting data distributions and capturing statistical correlation. However, in practical applications, the task and/or the feature space of the data may be different at test time and the function learned might not generalize. Hence, the associations learned during training might be spurious or not representative of the true objective. This problem has been recognized for a long time by the community and many strategies to counter this problem (Pan and Yang 2010). In this section, we will concentrate on the problem of domain adaptation, as defined by (Ben-David et al. 2010) and review some practical strategies used for tackling this problem.

Domain adaptation is the problem of adapting a classifier trained on a labelled source domain to a target domain with few or no labelled samples. In this work, we will only consider the unsupervised case where no label is available in the target domain. We will define a domain, as in (Ben-David et al. 2010), as a tuple $(\mathbb{P}_x, f_x(x))$, where f_x is the ground truth labelling function. In the case of binary classification, $f_x : X \to [0, 1]$ is a mapping on the line between 0 and 1. However, this definition can be easily generalized if f_x maps to the K-simplex where K is the number of classes.

The objective of classification, as described earlier, is to learn a hypothesis function h in order to minimize the risk $\mathcal{R}(f_x, h)$.

However, the problem of domain adaptation is hard because we cannot optimize our objective explicitly, at least not for the target domain. We instead have to rely on assumptions and implicit biases. The literature describing tricks to achieve good domain adaptation is vast. For a comprehensive and more complete review, the reader can read (Wilson and Cook 2018).

Gradient reversal. The first trick that we will present, is the one of projecting the source and target distribution to a shared space with matched marginal distribution. Given $\mathcal{X}$ and $\mathcal{Y}$, two spaces with distributions $\mathbb{P}_x$ and $\mathbb{P}_y$ respectively, one can learn a function $h_1 : \{\mathcal{X}, \mathcal{Y}\} \to \mathcal{Z}$ with $\mathbb{P}_{zx} = h_1 \# \mathbb{P}_x$ and $\mathbb{P}_{zy} = h_1 \# \mathbb{P}_y$ the distributions induced by *pushing* $\mathbb{P}_x$ and $\mathbb{P}_y$ through h_1. One can match the marginals $\mathbb{P}_{zx}$ and $\mathbb{P}_{zy}$ by minimizing some distributional distance. For example, it is possible to use the same strategy as the GAN framework (Goodfellow et al. 2014) on the representation of a neural network (Ganin et al. 2016). The objective function is essentially

$$\mathcal{L}_d(h_1) = \max_d \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_x} \log(d(h_1(\boldsymbol{x}))) + \mathbb{E}_{\boldsymbol{y} \sim \mathbb{P}_y} \log(1 - d(h_1(\boldsymbol{y})))$$

where $d : \{\boldsymbol{z}_x, \boldsymbol{z}_y\} \to [0, 1]$. They also train a classifier $h : h_2 \circ h_1$ where h_2 is also a function $h_2 : \{\boldsymbol{z}_x, \boldsymbol{z}_y\} \to \Delta^K$ mapping from the hidden representation to the K-simplex, using the cross-entropy

$$\mathcal{L}_x(h) = -\mathbb{E}_{\boldsymbol{x} \sim X}[f_x(\boldsymbol{x})^\top \ln h(\boldsymbol{x})].$$

Reducing the objective to

$$\min_h \mathcal{L}_y(h) + \lambda \mathcal{L}_d(h)$$

where $\lambda > 0$ is a regularization constant defined as an hyper parameter.

Cluster assumption. Introduced by (Shu et al. 2018) in the context of unsupervised domain adaptation, enforcing the *cluster assumptions* is another trick that showed impressive empirical results on domain adaptation. The cluster assumption enforces that the decision boundary of the classifier is in the low-density region for the source and target domain. A panoply of strategies has been proposed to enforce this assumption, showing constant gains empirically. (Grandvalet and Bengio 2005; Shu et al. 2018) proposes to minimize the conditional entropy

$$\mathcal{L}_c(h) = -\mathbb{E}_{\boldsymbol{y} \sim \mathbb{P}_y} \left[h(\boldsymbol{y})^\top \ln h(\boldsymbol{y}) \right].$$

In practice, because the classification loss already enforces a clustering with respect to the true labels, the above is only needed for the target domain. We can essentially view this loss as constraining f to cluster target data. However, (Grandvalet and Bengio 2005) recommends that f should be locally-Lipschitz to avoid f changing abruptly its prediction for small variation in the pixel values. (Shu et al. 2018) proposes to use a constraint (proposed by (Miyato et al. 2018)), calling it the virtual adversarial training, to enforces h to be locally-lipschitz. Hence they propose to use the additional term

$$\mathcal{L}_v(h) = \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_x} \left[\max_{||r|| \leq \epsilon} D_{\mathrm{KL}}(h(\boldsymbol{x}), h(\boldsymbol{x} + r)) \right]$$

where $\epsilon \sim \mathcal{N}(0, 1)$. One can interpret the above objective as encouraging h to be robust around an ϵ-ball around each h.

(Mao et al. 2019) propose Virtual Mixup loss to enforce the lipschitz property in an area of data interpolated in pixel space. The idea is that for a point $\tilde{\boldsymbol{x}} = \lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{x}'$ a linear interpolation of two points $\boldsymbol{x} \in \mathbb{P}_x$ and $\boldsymbol{x}' \in \mathbb{P}_x$, the prediction of h should be a linear interpolation of its input. More precisely, let $\tilde{\boldsymbol{y}} = \lambda h(\boldsymbol{x}) + (1 - \lambda) h(\boldsymbol{x}')$, the objective function is defined as

$$\mathcal{L}_m(h) = -\mathbb{E}_{(\boldsymbol{x}, \boldsymbol{x}') \sim \mathbb{P}_x} \tilde{\boldsymbol{y}}^\top \ln h(\tilde{\boldsymbol{x}}).$$

Combining all of theses objectives, we obtain the following objective

$$\min_h \mathcal{L}_y(h) + \lambda_1 \mathcal{L}_d(h) + \lambda_2 \mathcal{L}_c(h) + \lambda_3 \mathcal{L}_{vx}(h) + \lambda_4 \mathcal{L}_{vy}(h) + \lambda_5 \mathcal{L}_{mx}(h) + \lambda_6 \mathcal{L}_{my}(h).$$

Finally, in our last section, we look at the current works on image to image translation. It can be argued that, in theory, the problem of image to image translation is a transport problem where we want to transport images from one set to images to a target set. However, many practical considerations arise when we try to define the transport. For example, how should we define the cost of transferring images from one set to the other? The $L2$ distance captures variation at the pixel level. But is that enough to ensure a good transport? Perhaps

it is if the change in distribution is only local (e.g. changes in colour). But if the change in images is non-local, for example a translation, then the $L2$ distance is not practical. Fortunately, some techniques have been developed lately for performing image to image translation. We will review these techniques in the following section.

1.5. Image to image translation

The task of transferring images from one domain to a different domain, also known as Image-to-Image translation, has been popularized by (Isola et al. 2016). In essence, they propose to learn a conditional GAN (Mirza and Osindero 2014) on **paired** samples of the source and target domain.

Let $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathbb{P}_{x,y}$, paired samples. The objective is to learn a discriminator d on the joint $\mathbb{P}_{x,y}$ and a conditional generator $g : \boldsymbol{x} \to \boldsymbol{y}$ in an adversarial setting. The proposed Pix2Pix is

$$\mathcal{L}(d, g)_{\text{cGAN}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbb{P}_{x,y}} \left[\log d(\boldsymbol{x}, \boldsymbol{y})\right] + \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x} \left[\log(1 - d(\boldsymbol{x}, g(\boldsymbol{x}))\right].$$

Althought the above objective is the classical GAN loss, any alternative GAN objective could be used.

Pix2Pix also recommends using the L_1 loss between $g(\boldsymbol{x})$ and its paired $\boldsymbol{y}$ to encourage the mapping to be near the ground truth

$$\mathcal{L}_{\text{L}_1}(g) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbb{P}_{x,y}} ||\boldsymbol{y}, g(\boldsymbol{x})||.$$

Thus, Pix2Pix uses the following objective

$$\min_g \max_d \mathcal{L}_{\text{cGAN}}(g, d) + \lambda \mathcal{L}_{\text{L}_1}(g)$$

with $\lambda > 0$.

Taigman, Polyak, and Wolf 2016 proposed a **supervised** approach where the generator g is conditioned on the output of a pre-trained feature extraction f. They propose to train g using a mixture of losses described as follows. Given $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{y} \in \mathcal{Y}$, $\boldsymbol{x}' = g(f(\boldsymbol{x}))$, $\boldsymbol{y}' = g(f(\boldsymbol{y}))$, they first propose to train a discriminator $d : \boldsymbol{x} \to \Delta_3$ (the 3-simplex), which predict if the samples comes from $g(f(\boldsymbol{x}))$, $g(f(\boldsymbol{y}))$ or $\boldsymbol{y}$. We will denote $d(\cdot)_1$ the estimation of $g(f(\boldsymbol{x}))$, $d(\cdot)_1$ the estimation of $g(f(\boldsymbol{y}))$ and $d(c\dot{})_2$ the estimation of $\boldsymbol{y}$. The objective function of the discriminator is:

$$\mathcal{L}_D(d) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x} \mathbf{1}_0 \log d(g(f(\boldsymbol{x}))) + \mathbb{E}_{\boldsymbol{y}\sim\mathbb{P}_y} \mathbf{1}_1 \log d(g(f(\boldsymbol{y}))) + \mathbb{E}_{\boldsymbol{y}\sim\mathbb{P}_y} \mathbf{1}_2 \log d(\boldsymbol{y}).$$

Using the above discriminator, they define the GAN loss as follow:

$$\mathcal{L}_{\text{GAN}}(g) = -\mathbb{E}_{\boldsymbol{x}'\sim\mathbb{P}_x} \mathbf{1}_2 \log d(g(f(\boldsymbol{x}'))) - \mathbb{E}_{\boldsymbol{y}'\sim\mathbb{P}_y} \mathbf{1}_2 \log d(g(f(\boldsymbol{y}'))).$$

They also propose to use an L_1 constancy loss on the pre-trained representation of $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ to enforce that high level features captured by f are preserved after generation.

$$\mathcal{L}_{\text{const}}(g) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x}||f(\boldsymbol{x}) - f(g(f(\boldsymbol{x})))||_1.$$

Another objective they propose is the identity loss at the pixel level between $\boldsymbol{x}$ and $g(f(\boldsymbol{x}))$ which enforces the generated image to be close in pixel space to the source image:

$$\mathcal{L}_{\text{TID}}(g) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x}||\boldsymbol{x} - g(f(\boldsymbol{x}))||_1.$$

Finally, they propose an anisotropic total variation loss, which enforce a smoothing between pixels on generated images

$$\mathcal{L}_{\text{TV}}(g) = \mathbb{E}_{\boldsymbol{x}'\sim g\#f\#\mathbb{P}_x} \sum_{i=1}^{W}\sum_{j=1}^{H}((x'_{i,j+1} - x'_{i,j})^2 + (x'_{i+1,j} - x'_{i,j})^2)^{0.5}.$$

Combining all the losses, the global objective is the following

$$\max_d \min_g \mathcal{L}_D(d) + \mathcal{L}_{\text{GAN}}(g) + \lambda_1\mathcal{L}_{\text{const}}(g) + \lambda_2\mathcal{L}_{\text{TID}}(g) + \lambda_3\mathcal{L}_{\text{TV}}(g).$$

CyCADA (Hoffman et al. 2018) propose a similar idea where the substitute the $\mathcal{L}_{\text{TID}}$ and $\mathcal{L}_{\text{TV}}$ with a *cycle-consistency* loss, defined in the following.

CycleGAN Zhu et al. 2017 introduced the task of **unpaired** and **unsupervised** Image-to-Image translation by proposing to learn a mapping and its inverse constrained with a *cycle-consistency* loss and a GAN loss (Goodfellow et al. 2014) for each mapping. Let g_{xy} and g_{yx} the mapping $\mathcal{X} \mapsto \mathcal{Y}$ and $\mathcal{Y} \mapsto \mathcal{X}$ respectively, the cycle-consistency loss is defined as

$$\mathcal{L}_{\text{cyc}}(g_{xy}, g_{yx}) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x}||g_{yx}(g_{xy}(\boldsymbol{x})) - \boldsymbol{x}||_1 + \mathbb{E}_{\boldsymbol{y}\sim\mathbb{P}_y}||g_{xy}(g_{yx}(\boldsymbol{y})) - y)||_1$$

with the full objective defined as

$$\min_{g_{xy},g_{yx}} \mathcal{L}_{\text{GAN}_{XY}}(g_{xy}) + \mathcal{L}_{\text{GAN}_{YX}}(g_{yx}) + \lambda\mathcal{L}_{\text{cyc}}(g_{xy}, g_{yx})l$$

A number of works have been inspired by this task and have proposed a solution to improve on CycleGAN. But, as observed in Bézenac, Ayed, and Gallinari 2019; Galanti, Wolf, and Benaim 2018; Benaim, Galanti, and Wolf 2018, the cycle-consistency is theoretically ill-posed and can result in arbitrary mapping that may not be semantically relevant. Having said that, CycleGAN and its derivatives still show impressive empirical results. Bézenac, Ayed, and Gallinari 2019 argue that the cause of these good results is effectively not the *cycle-consistency* objective, but an implicit bias toward transformations that stay close to the source samples in features space due to the architectural design. In fact, they observe that the architecture used in these models usually have skip connections (He et al. 2015) which favor a transfer closer to the identity.

Chapter 2

Neural Wasserstein Flow

2.1. Introduction

The problem of Optimal Transport has been formalized in 1781 by Monge (Monge 1781). This problem – concerned with the optimal way of transporting distributions – has then been widely studied and applied in many fields including economics (Galichon 2016) and computer vision (Peyré and Cuturi 2019). More recently, the deep learning community has become interested in the field. Notably, it has seen great interest in the generative adversarial framework, as it can be been shown that approximating the Earth-mover distance it more stable than other metrics for training GANs (Arjovsky, Chintala, and Bottou 2017). Follow-up works have shown that it is possible to estimate the optimal transport for any cost c using neural networks by estimating the regularized transport plan using the smooth dual (Sanjabi et al. 2018; Seguy et al. 2018).

In this work, we are interested in how we can learn a flow between distributions following the Wasserstein geodesic using neural networks. The interest of this work is two-fold. First, Wasserstein geodesics, more specifically interpolations following the geodesic in the $\mathcal{W}_2$ space, have interesting geometric and uniqueness properties that can have practical utility. Second, neural networks can potentially enable this framework to scale to problems with data of high dimensionality.

To this end, we first show than neural networks can be used to learn a mapping to a weighted barycenter of k distributions using the formalism of (Agueh and Carlier 2011). We then generalize this idea to show that a neural network can infer any barycenters by framing the optimization problem as a Monte-Carlo integration over the interpolation parameter. This effectively allows us to learn a continuous interpolation between distributions. Finally, we show how this framework can be adapted to the GAN framework and show that we can sample from barycenter distributions.

Finally, we show results on shape interpolation and colour interpolation on images. We also show results on generating samples from the barycenter.

2.2. Background

2.2.1. Regularized optimal transport

In the GAN literature, the objective is usually to learn a parameterized generative model g to generate a target distribution of $\mathbb{P}_1$. Hence, without loss of generality, the mapping is usually learned as follow

$$\min_g W_1(g\#\mathbb{P}_0, \mathbb{P}_1) \tag{2.2.1}$$

where, in this specific case, $\mathbb{P}_0$ is usually defined as a prior distribution which we can sample such as the isotropic Gaussian. While equation 2.2.1 is fine for a generative model, it completely disregards the initial distribution $\mathbb{P}_0$ as the distance is computed between the target distribution $\mathbb{P}_1$ and the parameterized distribution $g\#\mathbb{P}_0$.

Using regularized optimal transport (Cuturi 2013)

$$\inf_{\pi \in \Pi(\mathbb{P}_0, \mathbb{P}_1)} \int_{(\boldsymbol{x}, \boldsymbol{x}') \sim \mathcal{X}_0 \times \mathcal{X}_1} c(\boldsymbol{x}, \boldsymbol{x}') d\pi(\boldsymbol{x}, \boldsymbol{x}') + \lambda \Omega(\pi), \tag{2.2.2}$$

where Ω is a strongly convex, such as L_2 norm or negative entropy, it is possible to recover a regularized transport plan π^ϵ (Blondel, Seguy, and Rolet 2018; Seguy et al. 2018) by training the smooth dual. In the sequel, we will only consider $\Omega = L_2$ without loss of generality. The smooth dual of equation 2.2.2 is

$$\begin{aligned} \sup_{\phi, \psi} \; & \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_0} \phi(\boldsymbol{x}) + \mathbb{E}_{\boldsymbol{x}' \sim \mathbb{P}_1} \psi(\boldsymbol{y}) \\ & - \frac{1}{2\lambda} \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{x}') \sim \mathbb{P}_0 \times \mathbb{P}_1} (\phi(\boldsymbol{x}) + \psi(\boldsymbol{x}') - c(\boldsymbol{x}, \boldsymbol{x}'))_+^2 \end{aligned} \tag{2.2.3}$$

and the regularized transport plan π^ϵ can be recovered as follow

$$\pi^\epsilon(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{\lambda} \mathbb{P}_0(\boldsymbol{x}) \mathbb{P}_1(\boldsymbol{x}') (\phi(\boldsymbol{x}) + \psi(\boldsymbol{x}') - c(\boldsymbol{x}, \boldsymbol{x}'))_+) \tag{2.2.4}$$

by re-arranging the terms, we obtain

$$\frac{\pi^\epsilon(\boldsymbol{x}, \boldsymbol{x}')}{\mathbb{P}_0(\boldsymbol{x}) \mathbb{P}_1(\boldsymbol{x}')} = \frac{1}{\lambda} (\phi(\boldsymbol{x}) + \psi(\boldsymbol{x}') - c(\boldsymbol{x}, \boldsymbol{x}'))_+) \tag{2.2.5}$$

which can be used to learn an optimal mapping between μ and ν

$$\min_g \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{x}') \sim \mathbb{P}_0 \times \mathbb{P}_1} c(g(\boldsymbol{x}), \boldsymbol{x}') \frac{\pi^\epsilon(\boldsymbol{x}, \boldsymbol{x}')}{\mathbb{P}_0(\boldsymbol{x}) \mathbb{P}_1(\boldsymbol{x})}. \tag{2.2.6}$$

We will re-use the identity from equation 2.2.5 in Algorithm 2 and Algorithm 1.

2.2.2. A flow on the Wasserstein geodesic

Geodesics are informally defined as paths of the minimum distance between two points on a manifold. Thus, to define a geodesic, we need a metric space. An example of a geodesic in Euclidean space between two points equipped with the L_2 norm is the straight line.

A Wasserstein geodesic is a geodesic defined in the Wasserstein space. We call a Wasserstein space $\mathcal{W}_p = (\mathcal{X}, W_p)$ simply a topological space equipped with the Wasserstein distance.

The Wasserstein geodesic has interesting theoretical properties, the first one being the existence of at least one geodesic for $p \geq 1$. However, the geodesic is unique only for $p \geq 2$. In other words, only one path of minimum distance exists between two distributions when $p \geq 2$. This is illustrated in the following example

Uniqueness of the geodesic on lines in 2D. Let $\mathbb{P}_0$ and $\mathbb{P}_1$ 2 uniform distributions defined in $\mathbb{R}^2$ as depicted in 2.1. We can see that at least two optimal mappings are possible when using the W_1 distance. However, only one mapping exists for W_2.

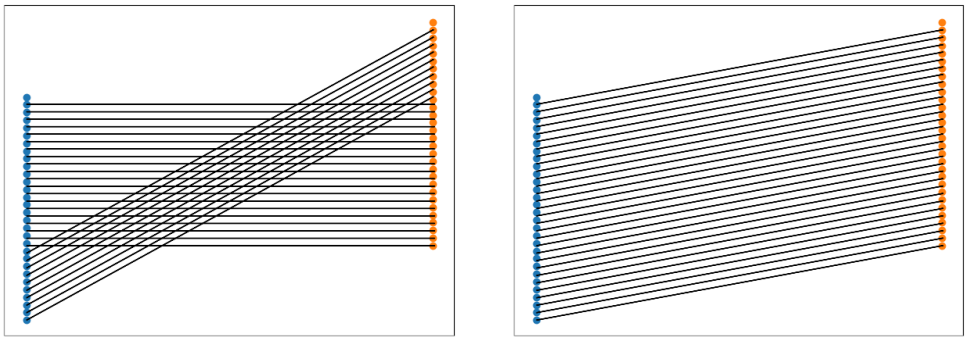

Figure 2.1. Two possible transportation plans. The two transportation plans are optimal for W_1 but only the one on the right is optimal for W_2.

Another interesting property of the Wasserstein geodesic is the geometry of the displacement interpolation of the distribution along the geodesic. In fact, the W_2 interpolation is spatially consistent because it captures the structure of the underlying metric space. This can be observed in the following example depicted in Figure 2.2 where we compare the interpolation in L_2, W_1 and W_2. We see that the L_2 interpolation captures the weighted average of the two distributions. While it is a consistent interpolation, it offers the effect of two static distribution appearing and disappearing depending on λ. The W_1 interpolation that we illustrate is one possible interpolation. Other interpolations, such as the one obtained by W_2 could be possible with W_1, but are not guaranteed. Finally, the unique W_2 interpolation offers a consistent interpolation that moves more naturally between the distributions in a geometric sense.

Other work has been interested in learning a flow on the Wasserstein geodesic using gradient flows (Peyré and Cuturi 2019). While this approach is interesting, we propose an alternative to learning a flow on the Wasserstein geodesic by considering weighted barycenters.

A barycenter distribution is a distribution on the geodesic. We show examples of barycenters between two Gaussians in Figure 2.2 and observe different barycenters depending on how

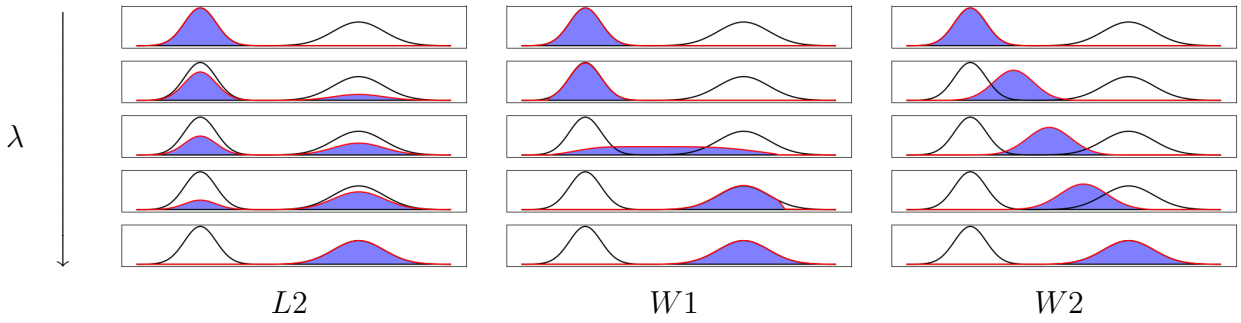

Figure 2.2. Comparison of the L_2, W_1 and W_2 interpolation. We show the barycenters at $\lambda = \{0, 0.25, 0.5, 0.75, 1\}$

we weight each distribution. It is possible to define a weighted barycenter between k distributions in Wasserstein space (Agueh and Carlier 2011). Concretely, given a set of probability measures $\{\mathbb{P}_i\}_{i=1}^k$ defined on $\mathcal{X}$ and let $\boldsymbol{\lambda} \in \Delta_k$ a vector on the k-simplex of positive weights that sum to 1, the Wasserstein Barycenter is defined as

$$\min_{\mathbb{P}^* \in \mathcal{P}(\mathcal{X})} \sum_{i=1}^{k} \lambda_i W_p^p(\mathbb{P}^*, \mathbb{P}_i). \tag{2.2.7}$$

2.3. Neural Wasserstein Barycenter

In equation 2.2.7, we have seen a definition of the barycenter of k probability measures $\{\mathbb{P}_i\}_{i=1}^k$. In this section, we propose to learn an approximation of $\mathbb{P}^*$ by learning a mapping $g : \mathcal{X} \to \mathcal{X}$, parametrized as a neural network, pushing an arbitrary probability measure $\mathbb{P}^* = g\#\mathbb{P}$ to the barycenter. In practice, we will define such arbitrary measures as samples from one of $\mathbb{P}_i$ and will only learn $\mathbb{P}^*$ implicitly. For simplicity, we will consider $\mathbb{P} = \mathbb{P}_1$.

We have seen that its possible to learn a regularized transport plan (equation 2.2.4) using the smooth dual equation 2.2.3 by parametrizing the critics as a neural network (Seguy et al. 2018). It is also possible to use this transport plan to learn a mapping from $\mathbb{P}_0$ to $\mathbb{P}_1$ as seen in equation 2.2.6.

Building on that, we propose to learn a mapping to a barycenter parametrized by a neural network by using the regularized transport plan. In essence, we propose to learn g by solving the following objective

$$\min_g \sum_{i=1}^{k} \lambda_i \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{x}') \sim \pi^\epsilon(\mathbb{P}_1, \mathbb{P}_i)} c(g(\boldsymbol{x}), \boldsymbol{x}') \tag{2.3.1}$$

where $\boldsymbol{\lambda} \in \Delta_k$. Intuitively, g maps to a barycenter of barycenter. The first barycenter is defined by the transport plan of equation 2.3.1 which define a target barycentric point within a distribution $\mathbb{P}_i$ for each point in $\mathbb{P}_1$. The second barycenter is defined as the Fréchet mean over the k distributions $\mathbb{P}_i$.

In practice, we need to train two critics for each transport plan π when computing the W_2 distance. However, we note that it could be possible to reduce the number of critics to one if using the c-transform (Villani 2008). However, we leave such exploration for future work.

Contrary to WGAN, because the transport plans are computed with respect to a fixed distribution, it is possible to pre-train them before training g, potentially stabilizing the optimization. Considering the regularized transport plan, we use the identity in equation 2.2.5 to train the mapping g. We depict in more detail how to train a neural Wasserstein barycenter in algorithm 1

Algorithm 1 Neural Wasserstein Barycenter

Input: α (learning rate), c (cost function), g (initialized generator), $\{\phi_i\}_{i=1}^k$, $\{\psi_i\}_{i=1}^k$ (initialized critics), $\{\mathbb{P}_i\}_{i=1}^k$ (target distributions), $\boldsymbol{\lambda}$ (Concentration of each distribution)
Train the critics $\{\phi_i\}_{i=1}^k$ and $\{\psi_i\}_{i=1}^k$
for $j = 1$ **to** k **do**
  repeat
    $\boldsymbol{x} \sim \mathbb{P}_1$
    $\boldsymbol{x}' \sim \mathbb{P}_j$
    $\delta \leftarrow \phi_j(\boldsymbol{x}) + \phi_j^c(\boldsymbol{x}') - \frac{1}{2\lambda}(\phi_j(\boldsymbol{x}) + \psi_j(\boldsymbol{x}') - c(\boldsymbol{x}, \boldsymbol{x}'))_+^2$
    $\phi_j \leftarrow \phi_j + \alpha \partial_{\phi_j} \delta$
    $\psi_j \leftarrow \psi_j + \alpha \partial_{\psi_j} \delta$
  until ϕ_j and ψ_j have converged
end for
repeat
  $\delta_g \leftarrow 0$
  for $j = 1$ **to** k **do**
    $\boldsymbol{x} \sim \mathbb{P}_1$
    $\boldsymbol{x}' \sim \mathbb{P}_j$
    $\delta_g \leftarrow \delta_g + \frac{1}{\lambda_j}(\phi_j(\boldsymbol{x}) + \psi_j(\boldsymbol{x}') + c(\boldsymbol{x}, \boldsymbol{x}')_+) \nabla_g c(g(\boldsymbol{x}), \boldsymbol{x}')$
  end for
  $g \leftarrow g - \alpha \delta_g$
until g has converged

2.4. Neural Wasserstein Flow

We propose to learn a continuous interpolation between distributions by learning a mapping to any barycenters on the geodesic of the distributions of interest. We propose to do so by using a similar strategy from equation 2.3.1, but instead of learning a fixed barycenter, we propose to condition g on $\boldsymbol{\lambda}$. Hence, by taking the expectation over Δ_k, which we can do by sampling over a uniform Dirichlet distribution, we can learn an amortized map $g_{\boldsymbol{\lambda}}$ to a $\boldsymbol{\lambda}$ weighted barycenter of $\{\mathbb{P}_i\}_{i=1}^k$.

The objective function is

$$\min_g \mathbb{E}_{\boldsymbol{\lambda}\sim\Delta_k} \sum_{i=1}^{k} \lambda_i \mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\sim\pi^\epsilon(\mathbb{P}_1,\mathbb{P}_i)} c(g(\boldsymbol{x},\boldsymbol{\lambda}),\boldsymbol{x}'). \tag{2.4.1}$$

We provide more details about the training in algorithm 2. The training is essentially the same procedure as algorithm 1, with the difference that g is conditioned on $\boldsymbol{\lambda}$

Algorithm 2 Neural Wasserstein Flow

Input: α (learning rate), c (cost function), g (initialized generator), $\{\phi_i\}_{i=1}^k$, $\{\psi_i\}_{i=1}^k$ (initialized critics), $\{\mathbb{P}_i\}_{i=1}^k$ (target distributions)
Train the critics $\{\phi_i\}_{i=1}^k$ and $\{\psi_i\}_{i=1}^k$ (see Algorithm 1)
repeat
  $\delta_g \leftarrow 0$
  $\boldsymbol{\lambda} \leftarrow \mathrm{Dir}(\mathbf{1})$
  for $j = 1$ **to** k **do**
    $\boldsymbol{x} \sim \mathbb{P}_1$
    $\boldsymbol{y} \sim \mathbb{P}_j$
    $\delta_g \leftarrow \delta_g + \frac{1}{\lambda_j}(\phi_j(\boldsymbol{x}) + \psi_j(\boldsymbol{x}') + c(\boldsymbol{x},\boldsymbol{x}')_+)\nabla_g c(g(\boldsymbol{x},\boldsymbol{\lambda}),\boldsymbol{x}')$
  end for
  $g \leftarrow g - \alpha\delta_g$
until g has converged

2.4.1. Generative Wasserstein Flow

The above Wasserstein flow can also be adapted to the generative framework, allowing one to sample from the barycenter distribution.

In the traditional GAN framework, we are interested in learning g using, for example equation 1.2. Hence, at every iteration, the critics used to estimate the Earth-mover distance have to be estimated using equation 2.2.1. This dynamic yields a min-max game objective as shown in (Arjovsky, Chintala, and Bottou 2017).

It has been shown that equation 2.2.6 can be used to learn a generative model (Sanjabi et al. 2018) as follows

$$\min_g \max_{\phi,\psi} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\sim g\#\mathbb{P}_0\times\mathbb{P}_1} c(\boldsymbol{x},\boldsymbol{x}') \frac{\pi^\epsilon(\boldsymbol{x},\boldsymbol{x}')}{\mathbb{P}_0(\boldsymbol{x})\mathbb{P}_1(\boldsymbol{x}')} \tag{2.4.2}$$

where $\mathbb{P}_0$ is taken as a prior distribution like the isotropic gaussian $\mathcal{N}(0,1)$.

It is straightforward to extend equation 2.4.2 to our Neural Wasserstein Barycenter framework as presented in equation 2.3.1. The only difference is that instead of taking $\mathbb{P}$ from one of our empirical distribution, we take $\mathbb{P}$ as the generated distribution $g\#\mathcal{N}(0,1)$. We use the isotropic Gaussian, but other prior distributions could be used as well. Hence, this framework can allow us to generate samples at the barycenter distribution.

Because we can learn a generative model to a barycenter, we can also extend it to learn a generative model on any Wasserstein barycenter similarly as in equation 2.4.4. This effectively yield a generative model on the Wasserstein geodesic. However, the extension is not as straightforward, because here the transport plan $\pi^\epsilon(g_{\boldsymbol{\lambda}}\#\mathbb{P}, \mathbb{P}_i)$ has now to take into account the interpolation parameter $\boldsymbol{\lambda}$. More precisely, because the generated distribution is dynamic in $\boldsymbol{\lambda}$, the critics used to estimate the transport plan also have to be dynamic for $\boldsymbol{\lambda}$. This can be done by conditioning the critics on $\boldsymbol{\lambda}$. Hence, equation 2.2.3 used to learn the transport plan has to be adapted. We propose the following adaptation where we also simply condition the critics on $\boldsymbol{\lambda}$

$$\begin{aligned}\sup_{\phi,\psi}\ & \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}}\phi(g(\boldsymbol{x},\boldsymbol{\lambda}),\boldsymbol{\lambda}) + \mathbb{E}_{\boldsymbol{x}'\sim\mathbb{P}_i}\psi(\boldsymbol{x}',\boldsymbol{\lambda}) \\ & -\frac{1}{2\epsilon}\mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\sim\mathbb{P}\times\mathbb{P}_i}(\phi(g(\boldsymbol{x},\boldsymbol{\lambda}),\boldsymbol{\lambda}) + \psi(\boldsymbol{x}',\boldsymbol{\lambda}) - c(g(\boldsymbol{x},\boldsymbol{\lambda}),\boldsymbol{x}'))_+^2.\end{aligned} \tag{2.4.3}$$

The above critics can be used in equation 2.2.4 to obtain a $\boldsymbol{\lambda}$ regularized transport plan $\pi^\epsilon_{\boldsymbol{\lambda}}$.

Similarly as in equation 2.4.2, we can learn a generative flow by using $\pi^\epsilon_{\boldsymbol{\lambda}}$ and solving the following min-max game

$$\min_g \max_{\phi,\psi} \mathbb{E}_{\boldsymbol{\lambda}\sim\Delta_k} \sum_{i=1}^k \mathbb{E}_{(\boldsymbol{x},\boldsymbol{x}')\sim g_{\lambda_i}\#\mathbb{P}\times\mathbb{P}_i} c(\boldsymbol{x},\boldsymbol{x}') \frac{\pi^\epsilon_{\lambda_i}(\boldsymbol{x},\boldsymbol{x}')}{\mathbb{P}(\boldsymbol{x})\mathbb{P}_i(\boldsymbol{x}')}. \tag{2.4.4}$$

2.5. Experiments

2.5.1. Shape interpolation

We first show results on shape interpolation. In these experiments, we represent shapes as points cloud. More precisely, we consider a square, a circle and a star and we want to map points from the square to points on the geodesic conditioned on a concatenation of the interpolation vector $\boldsymbol{\lambda}$ on the 3-simplex and the points in $\mathbb{R}^2$.

The mapping g is simply a three-layer MLP with ReLU activations. The critics are also three-layers MLPs with ReLUs. Everything is trained using Adam solver.

We show in figure 2.4 that our model interpolates reasonably well on the shape and each point preserves its relative position. For example, red points stay at the top right corner throughout the non-rigid transformation. Thus, this model could have great potential for application on shape matching or point cloud interpolation. However, we leave the exploration of such applications as well as comparison with current literature on these tasks for future work.

2.5.2. Images experiments

In this subsection, we conduct experiments on images directly. While other techniques exist for computing Wasserstein interpolation in low dimensions like gradient flow, no known

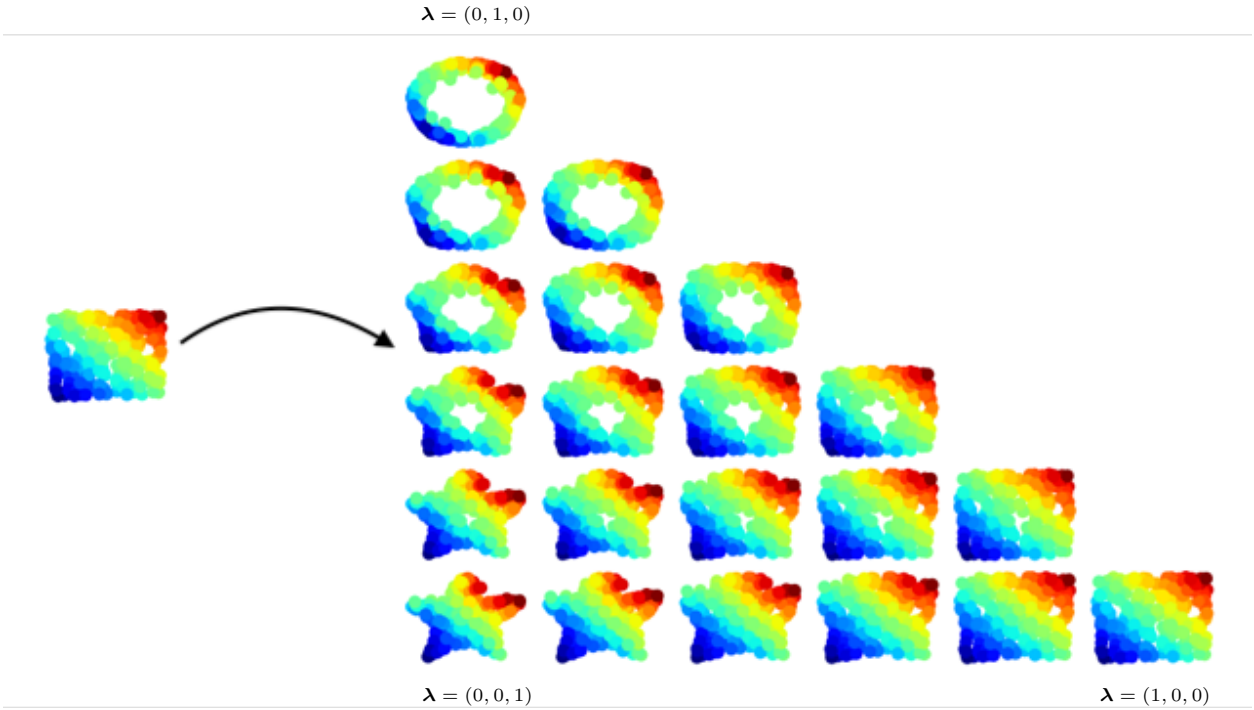

Figure 2.3. Shape interpolation. The initial shape is a square which is interpolated between a square $\boldsymbol{\lambda} = (1, 0, 0)$, an open circle $\boldsymbol{\lambda} = (0, 1, 0)$ and a star $\boldsymbol{\lambda} = (0, 0, 1)$. The purpose of the color is to show the initial positions of the points in order to show that the interpolation yields minimum distortion.

technique scale to high dimension. Hence, as far as we know, these are the first results of Wasserstein's interpolation in high dimensions. However, we note that practically, it could also be possible to perform the colour interpolation presented below by representing the colour of MNIST as a 3D histogram.

In these experiments, we will use the MNIST dataset which we will colour in blue or red. The MNIST dataset is a dataset of 60000 handwritten digits, we will consider three channels. In other words, the dimensionality of each image will be $28 \times 28 \times 3$.

2.5.2.1. *Image to image interpolation*

For the task of image to image interpolation, we condition a function, parameterized as a neural network, on a source image and we expect such an image to interpolate between distributions. In **??** we demonstrate the ability of our model to interpolate colours in MNIST images. More precisely, we define the source distribution as the usual white MNIST digits and the target distributions are red, blue and green digits. Because we have three target distributions, $\boldsymbol{\lambda}$ is defined on the 3-simplex.

We observe that our model can interpolate between the different colours and even generate a colour that is outside the training distribution but on the geodesic.

The mapping g is parametrized as an encoder-decoder function. First, we encode the source image to a vector. We then concatenate $\boldsymbol{\lambda}$ to the encoded embedding. Finally, we decode the concatenated embedding back to the pixel space.

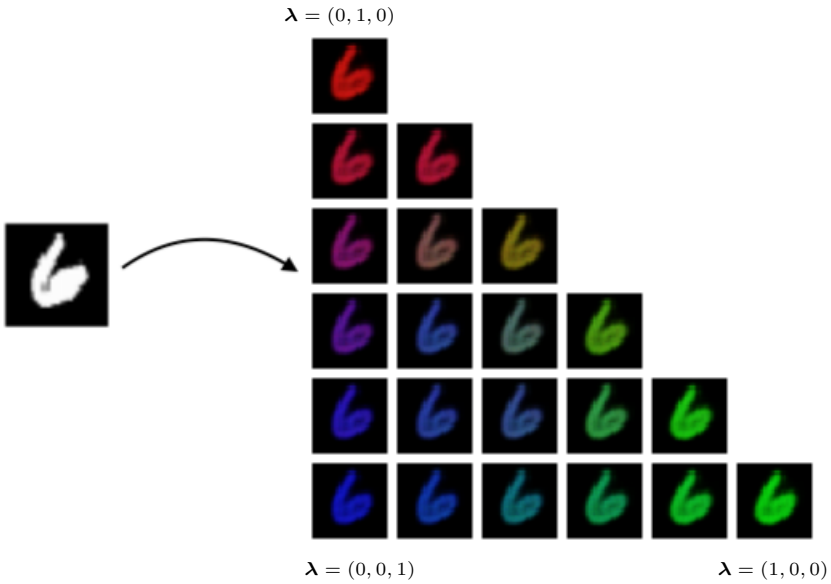

Figure 2.4. Image-to-image interpolation. The source image is a white "6" which is interpolated betweed red $\lambda = (0, 1, 0)$, blue $\lambda = (0, 0, 1)$ and green $\lambda = (1, 0, 0)$) "6".

2.5.2.2. *Generative interpolation*

In figure 2.5 we demonstrate our ability to generate samples along the geodesic. To do so, we train a GAN with similar architecture as in (Radford, Metz, and Chintala 2015) with the only difference that neither the generator nor the critics use batch norm. However, we could probably use batch norm if we conditioned the parameters on the interpolation parameter λ. Hence, the generator generates the barycenter distribution corresponding to λ by taking as input a sample from a prior distribution $\boldsymbol{z} \sim \mathcal{N}(0, 1)$ and λ.

Our training set is only composed of blue MNIST digits ($\lambda = 0$) and red MNIST digits ($\lambda = 1$). As we see, we can generate purple digits that are outside of the training distribution.

2.6. Conclusion

In conclusion, we presented Neural Wasserstein flow, a model parameterized by a neural network that can learn a mapping to any barycenter on the Wasserstein geodesic, effectively learning a continuous interpolation between distributions. We show results on points cloud as well as colour interpolation on the pixel representation of the images. Finally, we also show that our technique can also be used as a generative model.

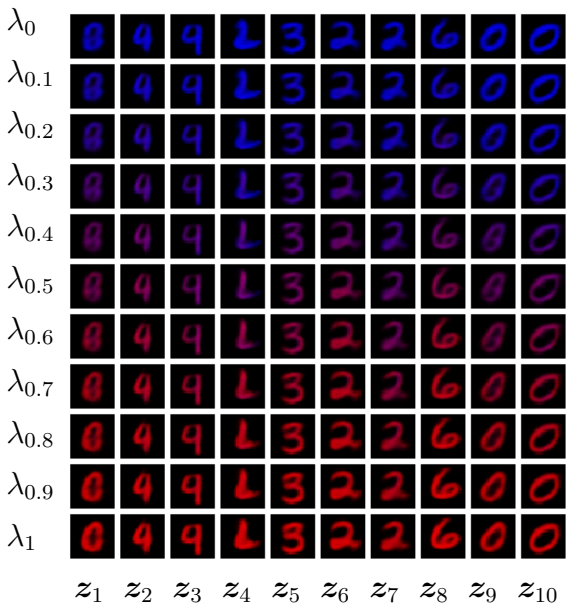

Figure 2.5. Generated interpolation between blue and red MNSIT.

However, one of the downsides of the Wasserstein distance when applying it to images is that it relies on the L_2 distance defined pixel-wise. Hence, only local interpolation can be applied, such as colour interpolation. High-level interpolation such as class semantic interpolation or even affine transformation on images would require either an objective function designed for images or a different representation of the images which would allow the L_2 norm to capture theses high-level semantic features.

Chapter 3

Toward high level semantic unsupervised domain translation

3.1. Introduction

Unpaired and unsupervised image-to-image translation has captured a lot of interest after it has seen great empirical success (Kim et al. 2017; Zhu et al. 2017) on tasks that require low-level semantic transfer like colours and textures. However, transferring high-level semantic features is still an open question. An example of such a task would be transferring MNIST digits to SVHN digits and vice-versa or transferring doodle images to real images and vice-versa. In such a task, the model has to capture semantically shared features.

In this chapter, we study the problem of such semantic-preserving unsupervised domain translation (SPUDT). While existing works have considered semi-supervised variants by training classifiers with labels on the source domain, we show it is possible to perform SPUDT without any pairing or any labels, but simply with access to samples from the two domains.

Based on empirical observations, we propose a framework for performing SPUDT. Our framework decouples the translation task into three subtasks: (1) unsupervised learning of a semantic representation in the source domain, which we use to map a sample from the source domain to a code in a latent space. The representation learning methods include but are not limited to clustering or latent factor taken as the code of the encoder of an auto-encoder; (2) adapting these representations to the target domain, such that we preserve the shared semantic information across domains; and finally (3) learning a conditional generative model for generation in the target domain, where the conditioning input is the semantic-preserving and domain-invariant representation. We implement our framework through leveraging advances in unsupervised clustering techniques for semantic representation learning, domain adaptation for adapting representations across domains, and conditional GANs for finally generating samples in the target domain.

While existing works have tended to show results that transfer from complex domains to simpler ones (such as SVHN→MNIST, or Faces→Emoji), using our pipeline we are able to show unsupervised domain translation from simpler to complex domains such as with the MNIST→SVHN and QuickDraw→FashionMNIST tasks. To the best of our knowledge, this is the first time successful unsupervised domain translation has been demonstrated in such challenging source-to-target domain-shift settings.

3.2. Related work

Domain translation has been popularized following the work on Image-to-Image translation by (Isola et al. 2016) in which they propose to learn a conditional GAN (Mirza and Osindero 2014) on paired samples of the source and target domain.

Domain Transfer networks (Taigman, Polyak, and Wolf 2016) propose a semi-supervised approach where the generation objective is a composition of a GAN loss (Goodfellow et al. 2014) an identity loss $||\boldsymbol{x} - \boldsymbol{x}'||_2$ and a *f-constancy* loss $||f(\boldsymbol{x}), f(\boldsymbol{x}')||_2$, where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are source and generated samples respectively and f is a supervised pre-trained function on the source domain. In our work, we observe that how f is learned is fundamental to a good transfer and essentially propose to learn it in an unsupervised manner.

CycleGAN (Zhu et al. 2017) introduced the task of unpaired and unsupervised Image-to-Image translation by proposing to learn a mapping and its inverse constrained with a *cycle-consistency* loss and a GAN loss for each mapping. Follow-up work have been inspired by this task and have proposed solutions to improve on CycleGAN (Liu, Breuel, and Kautz 2017; Kim et al. 2017; Almahairi et al. 2018; Huang et al. 2018). But, as observed in Benaim, Galanti, and Wolf 2018; Galanti, Wolf, and Benaim 2018; Bézenac, Ayed, and Gallinari 2019, the *cycle-consistency* is theoretically ill-posed and can result in arbitrary mapping that may not be semantically relevant. Having said that, CycleGAN and its derivatives still show impressive empirical results. However, Bézenac, Ayed, and Gallinari 2019 argues that the cause of the good results is effectively not the *cycle-consistency* objective, but an implicit bias toward transformations that stay close to the source samples in features space due to the architectural design.

While enforcing this implicit bias can improve results on tasks that require little spatial or geometrical transformation (e.g. transferring edges to shoes), going in this direction won't solve more difficult tasks where the semantics are not encoded at the feature level.

In order to tackle this problem, we propose to re-use ideas from the clustering and the domain adaptation communities. Regularized Information Maximization (Gomes, Krause, and Perona 2010) proposes to learn a regularized mapping from samples to clusters that maximizes mutual information while minimizing a pre-defined constraints. IMSAT (Hu et al. 2017) proposes a regularization that constraint a sample and its transformation to map to the sample cluster.

Recent work on domain adaptation (Shu et al. 2018; Mao et al. 2019) has achieved impressive results on datasets like MNIST to SVHN by proposing a number of regularizations and assumptions – mainly the cluster assumption – on the classifier.

3.3. Conditional representation GAN

Learning generative models have seen tremendous progress recently. We first present some background on generative adversarial networks before presenting the conditional representation generation framework that is general and agnostic to the domain invariant representation learning scheme.

3.3.1. Generative adversarial network

Given an unknown target probability measure $\mathbb{P}_x$ over a topological space $\mathcal{X}$ (e.g. a space of images), Generative Adversarial Networks (GANs) (Goodfellow et al. 2014) aim at learning a map $g : \mathcal{Z} \to \mathcal{X}$ minimizing some discrepency[1] between $g\#\mathbb{P}_z$ and $\mathbb{P}_x$, where $\mathbb{P}_z$ is a known fixed probability measure (often an isotropic gaussian) over a space $\mathcal{Z}$, and $g\#\mathbb{P}_z$ denotes the push-forward of $\mathbb{P}_z$ by g. The optimization procedure is generally characterized by a min-max game where a critic aims to maximize the objective and a generator aims at minimizing the same objective. In the following, without loss of generality, we will consider the minimization of the Earth-mover distance:

$$\min_{g:\mathcal{Z}\to\mathcal{X}} W_1(g\#\mathbb{P}_z, \mathbb{P}_x). \tag{3.3.1}$$

The Earth-mover distance can be cast into an optimization problem over 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$, yielding the following min-max optimization problem (**arjovsky2017wasserstein**)[2]:

$$\min_{g} \max_{f:\text{ 1-Lipschitz}} \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x} f(\boldsymbol{x}) - \mathbb{E}_{\hat{\boldsymbol{x}}\sim g\#\mathbb{P}_z} f(\hat{\boldsymbol{x}}). \tag{3.3.2}$$

Conditional GAN (Mirza and Osindero 2014) generalizes this setup to the case where the data is labelled with elements in a discrete set $\mathcal{C}$. That is, a target joint distribution $\mathbb{P}_{x,c}$ over $\mathcal{X} \times \mathcal{C}$ is approximated via the objective

$$\min_{g:\mathcal{Z}\times\mathcal{C}\to\mathcal{X}\times\mathcal{C}} W_1(g\#\mathbb{P}_{z,c}, \mathbb{P}_{x,c}), \tag{3.3.3}$$

where $\mathbb{P}_{z,c}$ is a fixed and known joint probability measure over the latent space $\mathcal{Z} \times \mathcal{C}$, often taken to be the product of a Gaussian and a uniform distribution over the labels. Evidently, equation 3.3.3 implies $g\# : \mathbb{P}_{z,c} \to \mathbb{P}_{x,c}$ and thus, g models the conditional probability distribution $\mathbb{P}_{x|c}$. A visual depiction of the conditional GAN is provided in figure 3.1.

[1]The discrepancy was originally defined as the Jensen Shannon Divergence for an optimal discriminator.
[2]In practice, the hard Lipschitz constraint can be relaxed into a soft constraint, see for instance the gradient penalty in WGAN-GP (Gulrajani et al. 2017)

3.3.2. Conditional Representation GAN

Conditional Representation GAN (CR-GAN) generalizes the formulation of conditional GANs given in equation equation 3.3.3. Figure 3.1 compare the traditional GAN framework with CRGAN. Let two spaces be $\mathcal{X}$ and $\mathcal{Y}$. We define generally $h : \{\mathcal{X}, \mathcal{Y}\} \to \mathcal{H}$ a given function that maps $\mathcal{X}$ and $\mathcal{Y}$ to a common hidden representation $\mathcal{H}$, capturing the features we are interested to transfer. Two specific methods for learning h that capture high level semantic features without supervision will be given in the following section. We define a function $g : \mathcal{Z} \times \mathcal{H} \to \mathcal{Y}$ a generator conditioned on the representation $\mathcal{H}$. Without loss of generality, the training objective of CRGAN is

$$\min_{g} \max_{f:\text{1-Lipschitz}} \mathbb{E}_{\boldsymbol{z}\sim\mathbb{P}_z, \boldsymbol{h}\sim h\#\mathbb{P}_x} f(g(\boldsymbol{z}, \boldsymbol{h}), \boldsymbol{h}) - \mathbb{E}_{\boldsymbol{y}\sim\mathbb{P}_y, \boldsymbol{h}\sim h\#\mathbb{P}_y} f(\boldsymbol{y}, \boldsymbol{h}). \tag{3.3.4}$$

In this paper we use the earth-mover distance, but other distributional discrepencies could be used as well.

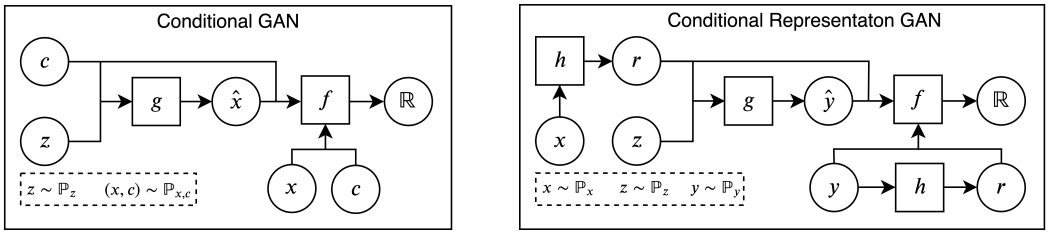

Figure 3.1. Left - Conditional GAN framework: the generator g is conditioned on a sample from a prior distributions and a class label. f is a critic which take either a real image and its associated label or a generated image and its conditioned label. Right - CR-GAN framework: the generator is conditioned on the representations $r \in \mathcal{R}$ from a pre-trained function h. f is a critic which take either a real image and its associated representation from a pre-trained function h or a generated image and its representation from h.

As in conditional GANs, a discrepancy between the joint distributions $\mathbb{P}_{y,h}$ and $\mathbb{P}_{\hat{y},h}$ is learned. Hence, similarly as in conditional GAN, the optimum is obtained when $\mathbb{P}_{y,h} = g\# : \mathbb{P}_{z,h}$. Thus, g models the conditional probability $\mathbb{P}_{y|h}$. A special case of CR-GAN is simply the conditional GAN where h is simply a human that classified samples from both $\mathcal{X}$ and $\mathcal{Y}$. In other words, $\boldsymbol{z}$ models the structure unrelated to the representation. In this case, because h compresses an image to a simple category, $\mathcal{Z}$ models the attributes unrelated to the category, as observed in conditional GANs (Mirza and Osindero 2014). As we will observe in the experiments section, this is true in general when h learns a compressed representation.

In this work, we are more interested in learning h with unsupervised learning algorithms that capture invariant features in $\mathcal{X}$ and $\mathcal{Y}$. However, we note that supervision can easily be incorporated into the representation learning algorithm depending on the task.

3.4. Domain invariant representation learning

In representation learning, it is assumed that latent factors generate the data distribution. The interest is usually in finding a representation that describes some of the factors to do well at solving a given task. For example, the goal of learning a representation for the task of classification is to learn a representation such that the data is linearly separated into classes.

Learning a *good* representation (Bengio, Courville, and Vincent 2013) in an unsupervised manner as well as learning invariant features across different domains (Ben-David et al. 2010) are still two active areas of research. As it was argued in Locatello et al. 2019, unsupervised learning has to define a set of assumption and inductive biases. In the following, we propose an algorithm for solving the specific case of unsupervised domain translation where we know a priori how many high-level categories exists. Given a different set of assumptions, other representation learning algorithms could be learned, potentially yielding a more general representation learning algorithm. We leave the exploration of these algorithms for future work.

3.4.1. Clustering adaptation

Our approach is conceptually straightforward. We propose to first learn k clusters and to use them as ground truth for learning an adapted classifier. The quality of the domain translation is proportional to the quality of the clusters and the adapted classifier. Fortunately, it is possible to evaluate both. In practice, any clustering and domain adaptation algorithm can be used, as long as they produce well-shared clusters. Hence, as unsupervised representation learning and domain adaptation techniques will be improved, a larger class of domain translation might be possible.

3.4.1.1. *Clustering*

Clustering defines the task of grouping similar objects. At the core of this unsupervised approach is the notion of similarity. By defining correctly the notion of similarity, clustering algorithms can extract relevant semantic information in a dataset.

Following RIM (Gomes, Krause, and Perona 2010) and IMSAT (Hu et al. 2017), we propose to learn a mapping $c : \mathcal{X} \to \mathcal{C}$, where $\mathcal{C} \in \mathbb{R}^k$ is a continuous space representing a soft clustering of $\mathcal{X}$, by optimizing the following objective

$$\min_c \lambda \mathcal{R}(c) - I(\mathcal{X}; \mathcal{C}), \tag{3.4.1}$$

where $\lambda > 0$ is a Lagrange multiplier, I is the mutual information defined as

$$I(\mathcal{X}; \mathcal{C}) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{X}),$$

$\mathcal{R}$ is a regularizer to restrict the class of functions. As it was proposed in IMSAT, we propose to define the regularization as follows

$$\mathcal{R}(c) = \mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x}||c(\boldsymbol{x}) - c(\boldsymbol{x}')||_2 \tag{3.4.2}$$

where $\boldsymbol{x}' = T(\boldsymbol{x})$, and T is defined as some transformation such as affine transformation. Essentially, this ensure that the mapping is invariant under the set of transformation defined by T.

If c is a deterministic function and $\mathcal{C}$ is discrete, then $H(\mathcal{C}|\mathcal{X}) = 0$, and $\max I(\mathcal{X};\mathcal{C}) = \max H(\mathcal{C})$. Hence, we are interested in a clustering of maximum entropy. This can be achieved if $\mathbb{P}_c = \mathbb{P}_{\text{cat}}$ where $\mathbb{P}_{\text{cat}}$ is the categorical distribution with uniform probability for every category. In practice, we can represent $\mathcal{C}$ as the space of one-hot vectors.

Hence, we can maximize the mutual information by mapping to the uniform categorical distribution denoted $\mathbb{P}_{\text{cat}}$. To do so, we propose to estimate the earth-mover distance $W_1(c\#\mathbb{P}_x, \mathbb{P}_{\text{cat}})$ using the Wasserstein GAN framework

$$I(\mathcal{X};\mathcal{C}) = \max_{f:\text{Lipschitz-1}} E_{\hat{\boldsymbol{u}}\sim c\#\mathbb{P}_x} f(\hat{\boldsymbol{u}}) - \mathbb{E}_{\boldsymbol{u}\sim\mathbb{P}_{\text{cat}}} f(\boldsymbol{u}). \tag{3.4.3}$$

Hence, minimizing equation 3.4.3 would essentially minimize the Earth-mover distance between $c\#\mathbb{P}_x$ and $\mathbb{P}_{\text{cat}}$ giving us a mapping of maximal $I(\mathcal{X};\mathcal{C})$.

Combining equation 3.4.2 and equation 3.4.3 in equation 3.4.1, we obtain the following objective for clustering

$$\min_c \max_{f:\text{Lipschitz-1}} \lambda\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x}||c(\boldsymbol{x}) - c(\boldsymbol{x}')||_2 - (\mathbb{E}_{\hat{\boldsymbol{u}}\sim c\#\mathbb{P}_x} f(\hat{\boldsymbol{u}}) - \mathbb{E}_{\boldsymbol{u}\sim\mathbb{P}_{\text{cat}}} f(\boldsymbol{u})). \tag{3.4.4}$$

3.4.1.2. *Domain adaptation*

Domain adaptation aims at adapting a function from a domain $\mathcal{X}$ so that it can perform well on domain $\mathcal{Y}$. Unsupervised domain adaptation refers to the case where $\mathcal{Y}$ is unlabelled during training. Different strategies exist for tackling this problem (Ganin et al. 2016; Shu et al. 2018; Mao et al. 2019). The strategy we will use in this work consists of matching the marginal distribution of a hidden representation of a neural network (Ganin et al. 2016) while enforcing the cluster assumption (Chapelle and Zien 2005). We motivate this approach because of the empirical success it has achieved in the modality of interest for this paper (Mao et al. 2019).

It has been shown that the error of a hypothesis function h on the target domain $\mathcal{Y}$ is upper bounded by the following (Ben-David et al. 2010)

$$\epsilon_{\mathcal{Y}}(h) \leq \epsilon_{\mathcal{X}}(h) + d(\mathcal{X},\mathcal{Y}) + \min_{h'} \epsilon_{\mathcal{X}}(h') + \epsilon_{\mathcal{Y}}(h') \tag{3.4.5}$$

where ϵ is the risk and can be computed given a loss function, for example the cross entropy

$$\mathcal{L}_x(h) = \epsilon_x(h) := -\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_x} t(\boldsymbol{x})^\top \ln h(\boldsymbol{x}). \tag{3.4.6}$$

Where $t : \mathcal{X} \to \mathcal{C}$ is a labelling function. Typically, a human labels each example, but in our case, it can also be learned using an unsupervised learning algorithm.

Considering that h is a neural network, it can be decomposed as follows: $h := h_2 \circ h_1$. Gradient reversal (Ganin et al. 2016) proposes matching the marginal of some hidden representation $h_1(\mathcal{X})$ and $h_1(\mathcal{Y})$:

$$\mathcal{L}_d(h_1) := \max_D \ \mathbb{E}_{\boldsymbol{h}_x \sim h_1 \# \mathbb{P}_x}[\log D(\boldsymbol{h}_x)] + \mathbb{E}_{\boldsymbol{h}_y \sim h_1 \# \mathbb{P}_y}[\log(1 - D(\boldsymbol{h}_y)]. \tag{3.4.7}$$

Here, $D : \{h_1 \# \mathbb{P}_x, h_1 \# \mathbb{P}_y\} \to [0, 1]$ is called the discriminator.

(Shu et al. 2018) point out that this is not enough and propose to leverage the cluster assumption which simply states that the decision boundary of h should be in the low-density region and points within the same cluster should belong to the same class. More concretely, the entropy of $h(\boldsymbol{y})$ should be low. This can be achieved with the following objective (Grandvalet and Bengio 2005)

$$\mathcal{L}_c(h) = -\mathbb{E}_{\boldsymbol{y} \sim \mathbb{P}_y} h(\boldsymbol{y})^\top \ln h(\boldsymbol{y}). \tag{3.4.8}$$

Because we estimate equation 3.4.8 with finite samples of $\mathcal{Y}$, h may not be locally-lipschitz (Grandvalet and Bengio 2005; Shu et al. 2018). Thus, they propose to further regularize the hypothesis function class by constraining it to be robust to local perturbation using the virtual adversarial training objective presented in (Miyato et al. 2018). Virtual adversarial training can be summarized as the following objective

$$\mathcal{L}_{vx}(h) = \max_{||\boldsymbol{r}||_2 \leq \epsilon} D_{\mathrm{KL}}(h(\boldsymbol{x}) || h(\boldsymbol{x} + \boldsymbol{r}))$$

where $\epsilon > 0$ is a small number. This essentially force h to be robust around an epsilon ball of radius ϵ. $\mathcal{L}_{vy}(h)$ is similarly defined for $\mathcal{Y}$.

Virtual Mixup Training (Shu et al. 2018) proposes to further regularize h by enforcing that the predictions at points $\tilde{\boldsymbol{x}}$ should themselves be linear interpolations of the predictions at $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. More concretely, let

$$\tilde{\boldsymbol{x}} = \alpha \boldsymbol{x}_1 + (1 - \alpha)\boldsymbol{x}_2$$
$$\tilde{\boldsymbol{y}} = \alpha h(\boldsymbol{x}_1) + (1 - \alpha) h(\boldsymbol{x}_2).$$

With $\alpha \sim U(0, 1)$, where $U(0, 1)$ is a continuous uniform distribution between 0 and 1.

The objective proposed is

$$\mathcal{L}_{mx}(h) = -\mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_x} \tilde{\boldsymbol{y}}^\top \ln h(\tilde{\boldsymbol{x}}).$$

With $\mathcal{L}_{my}(h)$ similarly defined for $\mathcal{Y}$.

Hence, the final domain adaptation objective used is

$$\min_h \mathcal{L}_x(h) + \mathcal{L}_d(h_1) + \mathcal{L}_c(h) + \mathcal{L}_{vx}(h) + \mathcal{L}_{vy}(h) + \mathcal{L}_{mx}(h) + \mathcal{L}_{my}(h).$$

3.4.2. Auto-encoder adaptation

We propose an invariant auto-encoder for learning a representation where the perceptual difference between domains is contained in low-level features.

An auto-encoder learns two functions, an encoder $e : \mathcal{X} \to \mathcal{H}$ and a decoder $d : \mathcal{H} \to \mathcal{X}$. The encoder is trained end-to-end to minimize the following objective

$$\mathcal{L}_{\text{ae}}(e, d) = \mathbb{E}_{\boldsymbol{x} \sim X} ||\boldsymbol{x} - d(e(\boldsymbol{x}))||_2. \tag{3.4.9}$$

A variant of the auto-encoder has been proposed where the auto-encoder has extra regularization, like VAE and denoising auto-encoder. While such variant has been shown to have the potential to learn good representations, we leave their analysis to future work.

Because we want $\mathcal{H}$ to be domain invariant, we propose to adapt the encoder of the auto-encoder by simply matching the marginal distribution of the hidden code. More precisely, let $\boldsymbol{x} \sim \mathbb{P}_x$ and $\boldsymbol{y} \sim \mathbb{P}_y$, samples following two distributions. Let $f : e(\mathcal{X}) \to \mathbb{R}$ a critic function, without loss of generality, we can define the marginal distribution loss as

$$\mathcal{L}_{\text{d}}(e) = \sup_{f:\text{Lipschitz-1}} \mathbb{E}_{\tilde{\boldsymbol{x}} \sim e\#\mathbb{P}_x} f(\tilde{\boldsymbol{x}}) - \mathbb{E}_{\hat{\boldsymbol{y}} \sim e\#\mathbb{P}_y} f(\hat{\boldsymbol{y}}). \tag{3.4.10}$$

Combining equation 3.4.9 and equation 3.4.10, we obtain the global loss of the auto-encoder adaptation

$$\min_{e,d} \mathcal{L}_{\text{ae}}(e, d) + \mathcal{L}_{\text{d}}(e). \tag{3.4.11}$$

3.5. Experiments

We now provide experimental results on our proposed approach. We first show that the clustering adaptation can be used to learn a shared clustering between domains, allowing us to cluster harder datasets. Then, we show that the representation learned can be used for high-level semantic image to image translation using CRGAN. We demonstrate our results on two benchmarks: MNIST (LeCun and Cortes 2010) - SVHN (Netzer et al. 2011) and Quickdraw (Fernandez et al. 2019) - Fashion (Xiao, Rasul, and Vollgraf 2017). Finally, we show that CRGAN can be used with a more general representation learning algorithm by applying it to the representation of edges to shoes (Yu and Grauman 2014) learned using an adapted auto-encoder.

3.5.1. Datasets

MNIST. is a balanced dataset of 60000 handwritten digits. The MNIST images are original 28×28 with one channel. To facilitate the training and easily compare them with SVHN, we upsample the MNIST images to 32×32 and triple the number of channels.
SVHN. is a dataset of 73257 digits from real-world images obtained using google street view. The images a 32×32 with three channels.

Quickdraw. is a dataset of handwritten objects from 345 categories containing millions of samples.
FashionMNIST. is a balanced dataset of 60000 black and white fashion objects across 10 classes.
Shoes. is a dataset of edges and coloured shoes. For the experiments involving shoes, we resize the dataset to 64×64.

In our experiments, we take classes that are both presents in the quickdraw dataset and Fashion MNIST. Moreover, we merge categories that are drawn identically by humans such as top and pullover. Hence, the categories we transfer are the following: top, pants, shoes, handbag. Finally, we resize both datasets to 32×32.

3.5.2. Implementation details

For the clustering algorithm, we use a six-layer convolutional neural network with average pooling for downsampling, Leaky ReLU for nonlinearity and batch normalization without affine parameters for normalization. For domain adaptation, we use the same architecture as in (Shu et al. 2018) except that we removed the affine parameters from batch normalization. Finally, for CRGAN, the critic and the generator follow the DCGAN architecture without batch normalization in the critic. Everything is trained using ADAM (Diederik P. Kingma and Ba 2014).

3.5.3. Clustering adaptation evaluation

We first compare the proposed clustering algorithm against IMSAT (Hu et al. 2017) and K-Means (Lloyd 2006) on our different benchmarks. The K-Means results were obtained using the generic implementation from Scikit-learn. The IMSAT results were obtained following the implementation details from (Hu et al. 2017). Contrary to what is sometimes being done in the literature, we do not pre-process the datasets by applying techniques edges filtering. Instead, we apply the clustering algorithms directly to the raw image.

We use the Purity evaluation metric for evaluating and comparing the variant of our clustering algorithm with other methods. The purity is computed as follows

$$\frac{1}{N}\sum_{i=1}^{N} \arg\max_{j} |l_i \cap c(\boldsymbol{x}_i)_j| \tag{3.5.1}$$

where l_i is the ground truth label and c is the clustering function.

Table 3.1 present the comparative results.

What we want to highlight here is the fact that some datasets like SVHN are hard to cluster by digit identity without any pre-processing and with the current clustering techniques. One of the causes is the different contrast of the colour of the digit as well as the artifacts in the images. Hence, adapting the clustering learned on a source dataset to a target dataset

Table 3.1. Comparing different clustering algorithms on MNIST, SVHN, Quickdraw and FashionMNIST (FMNIST) using equation 3.5.1. Our1 correspond our raw clustering algorithm, which is a modification of IMSAT. Our2 Correspond our clustering algorithm trained on a source dataset and adapted to a target dataset. The SVHN results were obtained by adapting the MNIST clustering on SVHN and the FashionMNIST results were obtained by adapting the Quickdraw clustering to FashionMNIST.

Data	K-Means	IMSAT	Our1	Our2
MNIST	59.45	98.24	98.89	N/A
SVHN	19.5	12.5	25.1	88.0
Quickdraw	80.08	84.96	92.57	N/A
FMNIST	89.8	94.8	81.15	92.4

that contains the same categories of interest but that does not have the same distractor can be an interesting strategy for learning clustering on real datasets.

3.5.4. CRGAN evaluation

We now present qualitative and quantitative results on our proposed CRGAN learned with the clustering adaptation representation learning presented above. The quantitative numbers are obtained by pre-training a classifier on the target domain and evaluating whether the label is preserved throughout the transfer. More precisely, let f a pre-trained classifier and $g_x = g \circ h$ the function which is the composition of the representation learning h and the generative model g. The quantitative evaluation is obtained as

$$\frac{1}{N}\sum_{i=1}^{N} f(g_x(\boldsymbol{x}_i, \boldsymbol{z})) \cap l_{\boldsymbol{x}_i} \tag{3.5.2}$$

where $l_{\boldsymbol{x}_i}$ is the true label of sample $\boldsymbol{x}_i$ and $\boldsymbol{z} \sim \mathcal{N}(0, 1)$.

We compare our algorithm with two other unsupervised and unpaired algorithms: CycleGAN and MUNIT. For CycleGAN and MUNIT, we followed the implementation from the authors' official repository. In both cases, the results were negative for transferring high level semantic.

3.5.4.1. *Quantitative analysis*

We present our quantitative results in table 3.2 using the evaluation metric presented in equation 3.5.2. The classification column presents the supervised results obtained when evaluating a classifier trained on the target domain. We see that current unsupervised and unpaired image-to-image translation techniques currently fails at the image to image translation task that requires transferring high-level semantic. CycleGAN and MUNIT obtain both random performances on MNIST-SVHN and FashionMNSIT to Quickdraw. However, for the Quickdraw to FashionMNIST results, we observe that CycleGAN may transfer the

high-level semantic class, but it fails at aligning it to the correct class in the target domain. Hence, our proposed approach is the only one that obtains sensible results on the task of transferring high-level semantic in an unsupervised and unpaired fashion.

Table 3.2. Domain translation accuracy obtained using equation 3.5.2 on MNIST-SVHN and Quickdraw-FashionMNIST.

Data	Cycle	MUNIT	Our	Class
MNIST-SVHN	10.89	10.37	73.15	93.00
SVHN-MNIST	11.27	9.84	77.51	99.14
QD-FMNIST	2.00	24.79	85.66	99.73
FMNIST-QD	26.18	25.06	85.60	99.87

3.5.4.2. *Qualitative analysis*

We now compare qualitative samples on MNIST-SVHN and Quickdraw-FashionMNIST generated from our benchmarks and our method.

By sampling different $z \in \mathbb{P}_z$, we can samples a diversity of samples from the target domain as observed in the example given in figure 3.4 for MNIST to SVHN. We observe that z indeed controls de style of SVHN while the source sample controls the digit identity.

We conclude this section by showing in figure 3.5 that CRGAN can be applied to other types of representation. In this example, we train an autoencoder with a gradient reversal loss on the encoder as described in section 3.4.2 on the usual edges to shoes dataset. Thus, the representation used is the output of the encoder. This result implicates that a more general representation learning algorithm that captures shared high-level semantic features could be used. We leave the exploration of such exploration for future work.

3.6. Conclusion and future works

In this work, we have presented a general view of conditional GAN. We have shown that conditional GAN can be used with any domain invariant representation learning scheme. Thus, we can essentially separate the task of representation learning from the task on generation. Given that insight, we have presented clustering adaptation, as a scheme to adapt the clustering learned on a source domain to a target domain. Coupled with the Conditional representation GAN, this allowed us to transfer high-level semantics from a source domain to a target domain. Finally, we have noted that a more general representation learning scheme can be used. For example, it is possible to adopt a classifier to learn a transfer on domains where only low-level features change, like the edge to shoes dataset. This insight indicates that a more general representation learning scheme that could capture high level semantic could be used in place of clustering. This would allow for more fine-grained transfer. For

Figure 3.2. Qualitative comparison of CycleGAN and MUNIT with our methods on MNIST to SVHN. For each figure, the even column c correspond to the source samples and the odd column $c+1$ correspond to source samples translated to the target domain. The first row of figures are the results obtained on CycleGAN. The second row of figures are the results on MUNIT. The last row of results are the results obtained using our approach.

example, in MNIST-SVHN, the digit identity is not the only feature shared. For example, the inclination of the digit or the thickness of the digit is also shared attributes that are not captured by our method. Finally, another representation learning scheme where supervision is used could be valuable to investigate and could potentially have interesting applications.

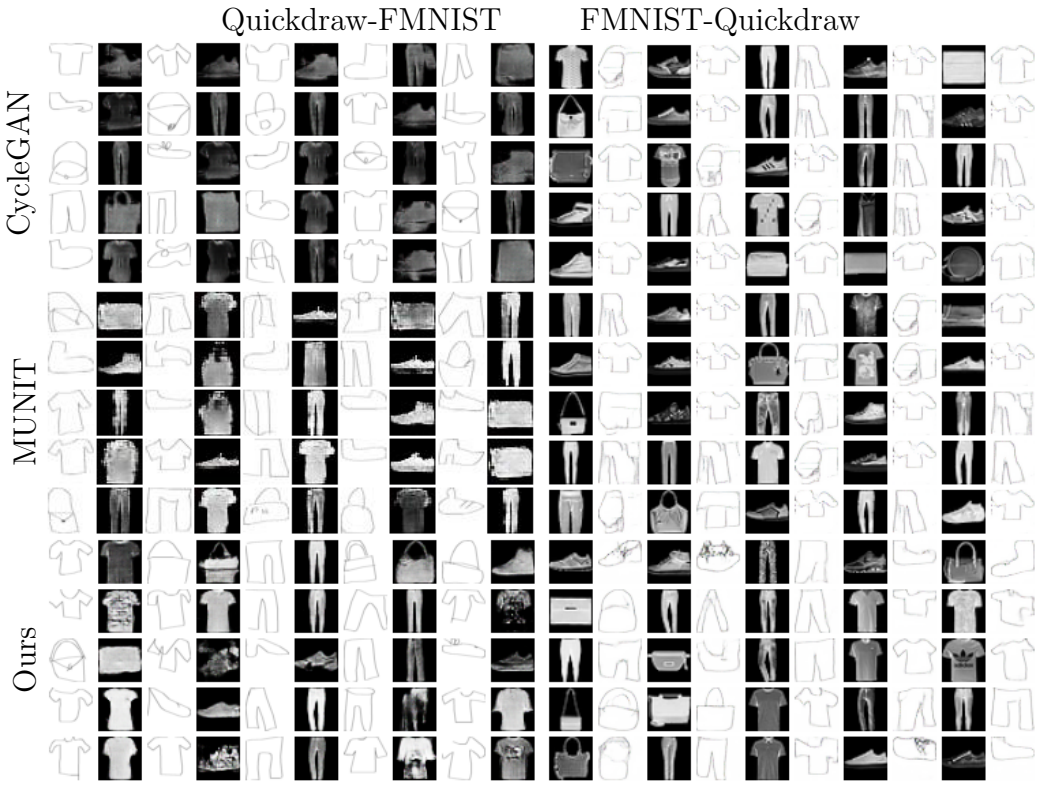

Figure 3.3. Qualitative comparison of CycleGAN and MUNIT with our methods on Quickdraw to FashionMNIST. For each figure, the even column c correspond to the source samples and the odd column $c + 1$ correspond to source samples translated to the target domain. The first row of figures are the results obtained on CycleGAN. The second row of figures are the results on MUNIT. The last row of results are the results obtained using our approach.

Figure 3.4. Effect of sampling a different z given an image. The first row presents the source samples to be transferred. Each subsequent rows the samples generated given a different z.

Figure 3.5. Edges-to-shoes by using the representation learned from an adapted autoencoder. The first row presents the source samples to be transferred. Each subsequent row is a sample generated given a different z.

Conclusion

We developed a framework for interpolating on the Wasserstein geodesic. This framework is flexible and can work with any number of distributions. We also show that it can be used as a generative model. The feature of it is that we can generate samples that are on the barycenter of distributions, enabling us to generate outside of the initial distributions. However, the main challenge of this approach goes back to the initial premise. If the content to be transferred is not representable feature-wise, then the L_2 norm is not a good measure. In Chapter 2, we presented the example of transferring colour, because colour can be easily compared pixel-wise. But, the experiment would not have been successful if we have tried to represent an affine transformation, for example, a translation.

To solve this problem, two avenues are possible. The first one is to develop cost functions that are better suited for the task. This angle is interesting and could potentially solve the problem for known transformation, such as affine. However, it is hard to imagine that we could define a cost function for complex non-linear transformation. The second avenue is to have a better representation. This is reminiscent of the Fourier transform which enables us to transform data in the frequency domain, where operations are easier, and then to transform back to the data domain. The challenge of such an approach is that we need to define how to obtain the desirable representation. The second issue is that we need to also define how to go from representation space to data space.

The third chapter was the first stab in this direction. The idea was simply to learn a representation that captures the high level semantic and to learn a mapping back to the data space. We see that such an approach can be successful but rely on having a definition of the high level semantic through labels. This is why we introduced the idea of learning the high level semantic through clustering. But this assumes that at least one of the domains can be clustered with respect to the desired semantic. Moreover, we map exactly to the same representation for all domains, which means that we could not possibly use the framework of optimal transport to transfer from one domain to the other in representation space.

I think that for future work, representation learning seems like a promising approach for solving the task of high level semantic unsupervised domain translation. But first, we have to ask how we can reliably capture all the high-level features. For example, in our

digits example, we captured the digit identity, but not the other features like thickness and rotation. We also have to think about how we can map back to the data domain.

Bibliography

Agueh, Martial and Guillaume Carlier (Jan. 2011). "Barycenters in the Wasserstein Space". In: *SIAM J. Math. Analysis* 43, pp. 904–924. DOI: 10.1137/100805741.

Almahairi, Amjad et al. (Oct. 2018). "Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 195–204. URL: http://proceedings.mlr.press/v80/almahairi18a.html.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (June 2017). "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 214–223. URL: http://proceedings.mlr.press/v70/arjovsky17a.html.

Ben-David, Shai et al. (May 2010). "A theory of learning from different domains". In: *Machine Learning* 79.1, pp. 151–175. ISSN: 1573-0565. DOI: 10.1007/s10994-009-5152-4. URL: https://doi.org/10.1007/s10994-009-5152-4.

Benaim, Sagie, Tomer Galanti, and Lior Wolf (2018). "Estimating the Success of Unsupervised Image to Image Translation". In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, pp. 222–238. ISBN: 978-3-030-01228-1.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (Aug. 2013). "Representation Learning: A Review and New Perspectives". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.8, pp. 1798–1828. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2013.50. URL: http://dx.doi.org/10.1109/TPAMI.2013.50.

Bézenac, Emmanuel de, Ibrahim Ayed, and Patrick Gallinari (2019). "Optimal Unsupervised Domain Translation". In: *CoRR* abs/1906.01292. arXiv: 1906.01292. URL: http://arxiv.org/abs/1906.01292.

Blondel, Mathieu, Vivien Seguy, and Antoine Rolet (Sept. 2018). "Smooth and Sparse Optimal Transport". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84.

Proceedings of Machine Learning Research. Playa Blanca, Lanzarote, Canary Islands: PMLR, pp. 880–889. URL: http://proceedings.mlr.press/v84/blondel18a.html.

Chapelle, O. and A. Zien (Jan. 2005). "Semi-Supervised Classification by Low Density Separation". In: *AISTATS 2005*. Max-Planck-Gesellschaft, pp. 57–64.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In: *CoRR* abs/1511.07289.

Cuturi, Marco (2013). "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., pp. 2292–2300. URL: http://papers.nips.cc/paper/4927-sinkhorn-distances-lightspeed-computation-of-optimal-transport.pdf.

Fernandez, Raul et al. (July 2019). *Quick, Stat!: A Statistical Analysis of the Quick, Draw! Dataset*.

Galanti, Tomer, Lior Wolf, and Sagie Benaim (2018). "The Role of Minimal Complexity Functions in Unsupervised Learning of Semantic Mappings". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1VjBebR-.

Galichon, Alfred (2016). *Optimal Transport Methods in Economics*. 1st ed. Princeton University Press. URL: https://EconPapers.repec.org/RePEc:pup:pbooks:10870.

Ganin, Yaroslav et al. (2016). "Domain-adversarial training of neural networks". In: *The Journal of Machine Learning Research* 17.1, pp. 2096–2030.

Gomes, Ryan, Andreas Krause, and Pietro Perona (2010). "Discriminative Clustering by Regularized Information Maximization". In: *NIPS*.

Goodfellow, Ian J. et al. (2014). "Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'14. Montreal, Canada: MIT Press, pp. 2672–2680. URL: http://dl.acm.org/citation.cfm?id=2969033.2969125.

Grandvalet, Yves and Yoshua Bengio (2005). "Semi-supervised Learning by Entropy Minimization". In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, pp. 529–536. URL: http://papers.nips.cc/paper/2740-semi-supervised-learning-by-entropy-minimization.pdf.

Gulrajani, Ishaan et al. (2017). "Improved Training of Wasserstein GANs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., pp. 5769–5779. ISBN: 978-1-5108-6096-4. URL: http://dl.acm.org/citation.cfm?id=3295222.3295327.

Han, Jun and Claudio Moraga (1995). "The Influence of the Sigmoid Function Parameters on the Speed of Backpropagation Learning". In: *Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation*. IWANN '96. London, UK, UK: Springer-Verlag, pp. 195–201. ISBN: 3-540-59497-3. URL: http://dl.acm.org/citation.cfm?id=646366.689307.

He, Kaiming et al. (2015). "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385. arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

Hoffman, Judy et al. (Oct. 2018). "CyCADA: Cycle-Consistent Adversarial Domain Adaptation". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. Stockholmsmässan, Stockholm Sweden: PMLR, pp. 1989–1998. URL: http://proceedings.mlr.press/v80/hoffman18a.html.

Hu, Weihua et al. (2017). "Learning discrete representations via information maximizing self-augmented training". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pp. 1558–1567.

Huang, Xun et al. (2018). "Multimodal Unsupervised Image-to-Image Translation". In: *ECCV*.

Isola, Phillip et al. (2016). "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5967–5976.

Kim, Taeksoo et al. (June 2017). "Learning to Discover Cross-Domain Relations with Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 1857–1865. URL: http://proceedings.mlr.press/v70/kim17a.html.

Kingma, Diederik P. and Jimmy Ba (2014). *Adam: A Method for Stochastic Optimization*. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. URL: http://arxiv.org/abs/1412.6980.

Kingma, Diederik P and Max Welling (2013). *Auto-Encoding Variational Bayes*. cite arxiv:1312.6114. URL: http://arxiv.org/abs/1312.6114.

Kramer, Mark (Feb. 1991). "Nonlinear Principal Component Analysis Using Auto-Associative Neural Networks". In: *AIChE Journal* 37, pp. 233–243. DOI: 10.1002/aic.690370209.

Kunnumkal, Sumit and Huseyin Topaloglu (2009). *A Stochastic Approximation Method with Max-Norm Projections and its Applications to the Q-Learning Algorithm*.

LeCun, Yann and Corinna Cortes (2010). "MNIST handwritten digit database". In: URL: http://yann.lecun.com/exdb/mnist/.

Liu, Ming-Yu, Thomas Breuel, and Jan Kautz (2017). "Unsupervised Image-to-Image Translation Networks". In: *ArXiv* abs/1703.00848.

Lloyd, S. (Sept. 2006). "Least Squares Quantization in PCM". In: *IEEE Trans. Inf. Theor.* 28.2, pp. 129–137. ISSN: 0018-9448. DOI: 10.1109/TIT.1982.1056489. URL: https://doi.org/10.1109/TIT.1982.1056489.

Locatello, Francesco et al. (Sept. 2019). "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 4114–4124. URL: http://proceedings.mlr.press/v97/locatello19a.html.

Mao, Xudong et al. (2019). "Virtual Mixup Training for Unsupervised Domain Adaptation". In: *CoRR* abs/1905.04215. arXiv: 1905.04215. URL: http://arxiv.org/abs/1905.04215.

McCann, Robert J. (1997). "A Convexity Principle for Interacting Gases". In: *Advances in Mathematics* 128.1, pp. 153–179. ISSN: 0001-8708. DOI: https://doi.org/10.1006/aima.1997.1634. URL: http://www.sciencedirect.com/science/article/pii/S0001870897916340.

Mirza, Mehdi and Simon Osindero (2014). "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784. arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

Miyato, Takeru et al. (2018). "Virtual adversarial training: a regularization method for supervised and semi-supervised learning". In: *IEEE transactions on pattern analysis and machine intelligence* 41.8, pp. 1979–1993.

Monge, Gaspard (1781). *Mémoire sur la théorie des déblais et des remblais*. De l'Imprimerie Royale.

Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press. ISBN: 0262018020, 9780262018029.

Nair, Vinod and Geoffrey E. Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, pp. 807–814. ISBN: 978-1-60558-907-7. URL: http://dl.acm.org/citation.cfm?id=3104322.3104425.

Netzer, Yuval et al. (2011). "The Street View House Numbers (SVHN) Dataset". In: URL: http://ufldl.stanford.edu/housenumbers/.

Pan, Sinno Jialin and Qiang Yang (Oct. 2010). "A Survey on Transfer Learning". In: *IEEE Trans. on Knowl. and Data Eng.* 22.10, pp. 1345–1359. ISSN: 1041-4347. DOI: 10.1109/TKDE.2009.191. URL: https://doi.org/10.1109/TKDE.2009.191.

Pearson, Karl (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". In: *Phil. Mag.* 6.2, pp. 559–572.

Peyré, Gabriel and Marco Cuturi (2019). "Computational Optimal Transport". In: *Foundations and Trends® in Machine Learning* 11.5-6, pp. 355–607. ISSN: 1935-8237. DOI: 10.1561/2200000073. URL: http://dx.doi.org/10.1561/2200000073.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. cite

arxiv:1511.06434Comment: Under review as a conference paper at ICLR 2016. URL: http://arxiv.org/abs/1511.06434.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1988). "Neurocomputing: Foundations of Research". In: ed. by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press. Chap. Learning Representations by Back-propagating Errors, pp. 696–699. ISBN: 0-262-01097-6. URL: http://dl.acm.org/citation.cfm?id=65669.104451.

Sanjabi, Maziar et al. (2018). "On the Convergence and Robustness of Training GANs with Regularized Optimal Transport". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., pp. 7091–7101. URL: http://papers.nips.cc/paper/7940-on-the-convergence-and-robustness-of-training-gans-with-regularized-optimal-transport.pdf.

Santambrogio, F. (2015). *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Progress in Nonlinear Differential Equations and Their Applications. Springer International Publishing. ISBN: 9783319208282. URL: https://books.google.ca/books?id=UOHHCgAAQBAJ.

Seguy, Vivien et al. (2018). "Large Scale Optimal Transport and Mapping Estimation". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1zlp1bRW.

Shu, Rui et al. (2018). "A DIRT-T Approach to Unsupervised Domain Adaptation". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1q-TM-AW.

Taigman, Yaniv, Adam Polyak, and Lior Wolf (2016). "Unsupervised Cross-Domain Image Generation". In: *ArXiv* abs/1611.02200.

Villani, C. (2008). *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg. ISBN: 9783540710509. URL: https://books.google.ca/books?id=hV8o5R7%5C_5tkC.

Vincent, Pascal et al. (2008). *Extracting and Composing Robust Features with Denoising Autoencoders*.

Wilson, Garrett and Diane Cook (Dec. 2018). *A Survey of Unsupervised Deep Domain Adaptation*.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". In: *CoRR* abs/1708.07747. arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

Yu, A. and K. Grauman (June 2014). "Fine-Grained Visual Comparisons with Local Learning". In: *Computer Vision and Pattern Recognition (CVPR)*.

Zhu, Jun-Yan et al. (2017). "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251.