Generative Deep Clustering with the Hierarchical and Relativistic Wasserstein Autoencoder GAN.

Gustavo Morais
gustavo.morais@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2021

Abstract

Clustering is one of the most important tasks in the unsupervised learning field, and traditional methods (like k-means) struggle when dealing with high-dimensional data. For this reason, many researchers have successfully attempted to use deep learning techniques to map the data into a lower dimensional, cluster friendly space. The process usually consists of using an autoencoder-based structure to find a meaningful latent space and then using traditional clustering algorithms on it.

In this dissertation, I developed a novel approach to clustering by using a generative hierarchical autoencoder containing multiple latent groups (the vanilla autoencoder has just one) that are regularized by latent relativistic discriminators in an adversarial manner. These latent groups are then clustered by the density-based clustering algorithm HDBSCAN, which automatically estimates the number of clusters. The resulting model is named Hierarchical and Relativistic Wasserstein Autoencoder GAN (HRWAE-GAN) and this work will show that utilizing multiple latent spaces does **not** offer any advantages when it comes to the task of clustering image datasets.

Keywords: Deep Clustering; Deep Learning; Generative Models; Hierarchical Models; Adversarial models; Relativistic discriminators.

1. Introduction

Clustering data is one of the fundamental problems in the unsupervised learning realm. Its goal is to identify data points that are similar to each other in an unsupervised manner (the data is unlabeled) based on some distance/similarity measure. A large number of clustering algorithms have been proposed [25] [3], but more traditional methods struggle when dealing with high-dimensional data due its underlying complexity [25].

One way to tackle this issue is to reduce the dimensionality of the data, mapping it to a new and more clustering friendly latent space. This is where deep learning techniques (deep clustering, in this case) have come into play. Although some non deep learning methods have been created, such as PCA [14] and UMAP [16], deep clustering has proved to greatly improve clustering performance and can even be combined with these aforementioned methods [15]. Several deep clustering algorithms have been proposed [12], and they usually consist of two parts: a latent representation finding (often Autoencoder based) and a clustering objective (where traditional clustering is commonly used). Their optimization can either be done jointly or separately.

Following this line of though, I will take on a novel approach to deep clustering using a hierarchical architecture with adversarial training. In deep learning autoencoders, a hierarchical architecture differs from a non-hierarchical one because it can include hierarchical groups between every pair of encoder-decoder hidden layers. In theory, this allows the model to capture both low level and high level details about the data at the different latent groups, thus holding a possible advantage over non-hierarchical models. The adversarial training is included to give the model generative abilities. It utilizes the well known GAN framework [6], which turns the training process into a competition between the generator (the model) and a Discriminator, who tries to tell generated samples apart from real samples. This results in the model being able to generate realistic looking samples.

2. Background

Deep clustering can be split into several categories. In this section, we'll discuss **Autoencoder** and **GAN** based clustering.

Autoencoder based clustering was first proposed in 2016 by Xie et al. with the paper "Un-

supervised deep embedding for clustering analysis" [24], where they formulated the architecture for **Deep Embedding Clustering** (DEC). DEC is widely considered as a deep clustering baseline and has been improved upon by many other researchers.

The main idea of AE based clustering is to perform clustering on the AE's latent space, optimizing both the network's loss as well as a clustering loss $L = \lambda L_n + (1-\lambda)L_c$ where $\lambda \in [0, 1]$ is a hyperparameter to balance the two losses.

DEC couples the standard AE loss with a KL divergence based loss that tries to minimize the distance between the soft assignments Q and a target distribution P (that depends on Q). The soft assignments can be defined as follows:

$$q_{ij} = \frac{(1+||z_i-\mu_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1+||z_i-\mu'_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}$$

where z_i is the embedding of an input x_i, α are the degrees of a Student's t-distribution and q_{ij} is the probability of a data point i belonging to cluster j. In this model, the training process minimizes L_n and L_c in two separate phases. **DCN** [26] came shortly after claiming that minimizing the two losses simultaneously would be beneficial.

N2D [15] uses a different approach. Instead of a joint optimization, it learns the embedded representations and, separately, performs a shallow clustering technique. Before starting the clustering phase, N2D uses UMAP to obtain a more clustering friendly latent space (keeping the same latent dimensionality). The authors claim that, after having tried different techniques, they found UMAP is able to find the more clusterable manifold. As discussed before, UMAP has the interesting property of providing well separated and compact clusters.

One clear advantage for N2D is the low level of complexity, since there is no joint optimization and UMAP is a fast algorithm.

SPICE [17], a Semantic Pseudo-labeling Framework for Image Clustering, utilizes self-learned pseudo-labels of high confidence in the embedded space to train a deep-clustering network. Initially, a latent space representation is constructed by a convolutional autoencoder SPICE-self. Based on this initial clustering, a set of reliable data points (a reliable data point will be assigned to its cluster with high certainty by some fuzzy clustering algorithm) are used to train a second network SPICE-semi. This network is trained directly with a loss function derived from the self-learned pseudo-labels. Consequently, SPICE is a semi-supervised classification model trained with pseudo-labels obtained from representation learning.

SPICE is a state-of-the-art model and currently ranks number one in unsupervised classification of several colored image datasets such as CIFAR-10, ImageNet-10 and Tiny-ImageNet.

SPC [13], which stands for Selective Pseudo-label clustering, is another semi-supervised learning deep clustering model. The general idea is similar to SPICE, but the two differ in what they consider to be confident pseudo labels and in the overall training procedure. While SPICE uses a confidence threshold, SPC pre trains a group of K autoencoders and then only uses the agreed points, which are points that received the same label from the clustering algorithm (HDBSCAN or GMM) applied to all K autoencoders' latent space, for classification training. An MLP is then trained with these agreed points for classification. This autoencoder and MLP training cycle is repeated until the number of agreed points stops increasing.

SPC is the current state-of-the-art model for unsupervised classification of the black and white numbers image datasets MNIST and USPS.

GAN based clustering uses the GAN's adversarial training framework in a few different ways.

Deep Adversarial Clustering (DAC) [8] uses an adversarial Autoencoder with a discriminator in the latent space to enforce a GMM prior. DAC has a similar architecture to WAE-GANs, except it is clustering oriented and uses an AE instead of a VAE.

3. Proposed method

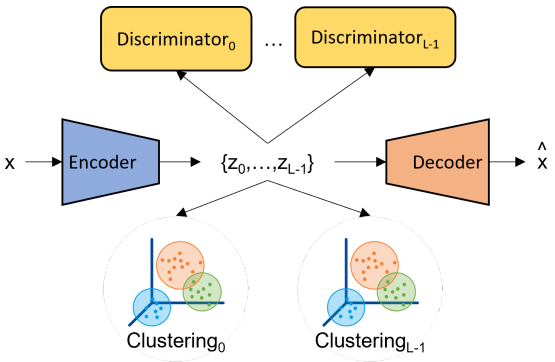


Figure 1: Simplified architecture of HRWAE-GAN (see Figure 2 for more details). The Encoder produces L latent variables, which are each judged by the discriminator during training and clustered when training is complete.

The Hierarchical and Relativistic Wasserstein Autoencoder GAN is comprised of two essential components: the Hierarchical and Relativistic Wasserstein GAN Autoencoder (HRWAE-GAN) and the clustering box (HDBSCAN paired with UMAP) applied to the latent space. These components will be described in the following sub subsections.

3.1. Autoencoder structure

This work can be divided in two big chunks: (1) building a generative framework, where the WAE-GAN [21] was adapted to make its training more stable, and (2) incorporating a hierarchical structure into the model. These chunks will be respectively explained in subsections 3.1.1 and 3.1.2.

3.1.1 Relativistic Wasserstein GAN

When comparing the Beta-VAE [9] and WAE-GAN [21] models, the latter has a better reconstruction ability but a more unstable training (Beta-VAE's training is completely stable). Reconstruction quality is of great importance as it allows us to infer how much information is retained in the latent space (in a fully unsupervised setting, at least), and so it correlates to its quality. The goal thus became to enable the usage of WAE-GAN by making its training more stable.

The training instability of GANs is a well known issue which has been addressed in many different works like [20], [10], [7] and [2]. While the label smoothing technique [20] is helpful but not completely effective, the Relativistic GAN's modified loss function [10] proved to completely solve training instability (with regards to the adversarial training) with minimal changes to the model and was therefore adopted (it is only required to make a small change in the adversarial loss function).

First, let us recall Generative Adversarial Networks (GANs): GANs are game theory based generative models. They use the concept of adversarial training to minimize a minimax loss function. As the name indicates, there are two adversaries: the Generator (G) and the Discriminator (D). Succinctly, G picks up a randomly generated variable z that is mapped to the dataset space and tries to generate a realistic data object G(z). Then, both real and generated data objects are fed to the discriminator D, who tries to correctly identify them as real or fake. This game proceeds as G tries to fool D and D tries to avoid that from happening. This can be expressed as:

$$min_G max_D L(D,G) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))] \quad (1)$$

The first half of the equation can be interpreted as the discriminator trying to maximize its ability to identify real data. The second part can be interpreted as the discriminator trying to maximize its ability to classify data generated by G as fake, while G tries to minimize it.

Wasserstein GANs (WAE-GANs) move the discriminator to the Autoencoder's latent space, where it will be tasked with distinguishing data points sampled from the prior distribution p_z (real points) and from the encoder's approximation $q_x(z)$ (fake points). Consequently, in WAE-GANs, the encoder serves as the GAN's generator. Equation 1 can be adapted to this use case as:

$$min_G max_D L(D,G) = E_{z\sim p_z}[logD(z)] + E_{\hat{z}\sim q_x(z)}[log(1-D(\hat{z}))] \quad (2)$$

where D is the latent discriminator and G is the encoder.

In [10], the author argues that standard GANs are missing a key property: the probability of real data being real should decrease as the probability of fake data being real increases. This makes use of the *a priori* knowledge that half of the dataset is fake. To fix this issue, a relativistic discriminator (RD) is used. The RD, instead of trying to classify single data points as being real or fake, takes pairs of points (z_{real}, z_{fake}) and outputs the probability of z_{real} being more realistic than z_{fake}. As the game progresses and G becomes more competent, D might even consider fake points to be more realistic than real ones, allowing G to generate samples of higher quality than before.

The loss functions for the RGAN's Discriminator and Generator are, respectively:

$$L_D = E_{(z,\hat{z})\sim(p_z,q_x(z))}[-log(f(D(z)-D(\hat{z})))] \quad (3)$$

$$L_G = E_{(z,\hat{z})\sim(p_z,q_x(z))}[-log(f(D(\hat{z})-D(z)))] \quad (4)$$

where f is generally the *sigmoid* function. In equation 3, the Discriminator tries to maximize the difference in realism between real and fake images; in equation 4, the Generator tries to maximize the opposite difference, thus trying to generate realistically looking fake samples.

Notice that while in equation (2) G only has influence in the rightmost part, in equations 3 and 4 it has full influence.

Experimentally, I found that the RD greatly improves the stability of adversarial training. Although it may require some tuning of the Discriminator's VC dimension, stable adversarial training is easily achievable.

3.1.2 Hierarchical structure

One of the main challenges with Autoencoder models is fully capturing both the global structure of data (such as general shape and color) as well as local structure (finer details such as lettering on an image of a shirt). The hierarchical latent structure presented in NVAE [22] was used to tackle this challenge to great effect. Instead of a single latent variable z, with a hierarchical Autoencoder we have a set of latent variables $\{z_0, ..., z_{L-1}\}$, where z_0 corresponds to the latent variable at the top of the hierarchy, z_{L-1} is positioned at the bottom and L is the number of latent groups. If we consider N to be the number of layers in the model, note that

$L \in [1, N]$, so $L \leq N$. In this setting, the latent groups at the top of the hierarchy are supposedly able to capture global structure whilst the bottom ones capture local structure.

Considering this hierarchical structure, the total Discriminator and Generator losses become:

$$L_D = \alpha_0 E_{(z,\hat{z})\sim(p_x,q_x(z))}[-log(f(D(z) - D(\hat{z})))] + \sum_{l=1}^{L-1} \alpha_l E_{(z,\hat{z})\sim(p_{(z_l|z_{<l})},(q_x(z|z_{<l}))}[-log(f(D(z) - D(\hat{z})))] \tag{5}$$

$$L_G = \alpha_0 E_{(z,\hat{z})\sim(p_x,q_x(z))}[-log(f(D(\hat{z}) - D(z)))] + \sum_{l=1}^{L-1} \alpha_l E_{(z,\hat{z})\sim(p_{(z_l|z_{<l})},(q_x(z|z_{<l}))}[-log(f(D(\hat{z}) - D(z)))] \tag{6}$$

where $\{\alpha_0, ..., \alpha_{L-1}\}$ are balancing terms with decreasing value and $q_x(z|z_{<l}) = \prod_{i=0}^{l-1} q_x(z_i|z_{<i})$ are the approximate posteriors the l^{th} group.

Regarding the priors, the top layer prior $p(z_0)$ corresponds to a simple normal distribution $\mathcal{N}(\mu, \sigma)$ with μ being a zero vector and σ being the identity covariance matrix with whatever dimensionality d_{latent} the top latent space may have. For the remaining priors $p(z_l|z_{<l})$ of the lower latent groups, they correspond to normal distributions as well but their parameters are computed by trainable parameter cells.

3.1.3 Dynamic architecture

The generic architecture of HRWAE-GAN can be visualized in Figure 2, where each cell is composed of convolutional and deconvolutional layers for the encoder and decoder structures, respectively. However, because images of larger dimensions are usually more complex, there is a need for the model depth to be data dependent. To account for this, the number of encoder cells N is determined by:

$$N = log_2(d) - 1 \tag{7}$$

where $d = \sqrt{D}$ with D being the dimensionality of each data point from some dataset (disregarding the number of channels C). This way, for 32x32xC images (of dimensionality $D = 32x32$), the model will form $log_2(32) - 1 = 4$ layers and output a feature map with dimensions 2x2xF (as it will for any other dimension whose square root is a square of 2), where F is the number of filters of the final layer. In order to fully reconstruct the input, the decoder must also have N cells.

3.1.4 Additional details

The implementation of HRWAE-GAN was additionally inspired by the work of [18], where the authors propose several techniques that improve the performance of deep convolutional GANs. Hence, the following strategy was employed:

- Replacement of pooling layers with strided convolutions because while pooling layers force the down-sizing artificially, strided convolutions allow the generator to learn it more naturally.
- Usage of batch normalization in both the generator and the discriminator. This method prevents mode collapse, which happens when the generator's outputs collapse to a single point that reliably fools the discriminator.

3.2. Clustering methodology

Considering the task of assigning N data points $X = [x^0, ..., x^{N-1}]$ to K clusters, we start by deriving the embeddings of each latent group $Z = \{[z_0^0, ..., z_0^{N-1}], ..., [z_{L-1}^0, ..., z_{L-1}^{N-1}]\}$, where L is the number of latent groups. Then, we may apply some clustering function to each group in Z.

In a fully unsupervised setting it's ideal to let the model identify the number of clusters on its own. For this reason, density based approaches can be extremely useful in such settings as they do not require the number of clusters K as a hyperparameter. In this work, HDBSCAN [4, 5] was used as the main clustering algorithm to cluster on the latent space. Because it is density based, HDBSCAN also has the useful feature of noise identification. Any point x_1 that does not have a neighbor x_2 with $d_{1,2} < \epsilon$, where $d_{1,2}$ is the distance between two points and ϵ is some distance determined by HDBSCAN, is considered as noise. The hierarchical aspect of HDBSCAN allows this distance ϵ to vary between clusters, thus allowing for clusters of different densities.

However, density based approaches are known to struggle with high feature dimensionality. For some image dataset X with moderate dimensions, such as 32x32x1, the original dimensionality D is too large for effective clustering. This is why it is key to derive a lower dimensionality latent space Z from X where HDBSCAN is capable of operating effectively.

3.2.1 UMAP cluster separator and compactor

UMAP [16] is a manifold neighbor graph technique that can be used to visualize data in lower dimensions, just like PCA [14] and t-SNE [23], but has the interesting properties of maintaining both global and local structures while providing more compact sub spaces. In [15], the authors took advantage of UMAP by applying it to the Autoencoder's latent space. As a result, they achieved a better clustering performance.

In this work, UMAP improved the clustering performance as well. In figure 3, we can see that the latent space becomes better separated and with more compact clusters after applying a UMAP transform.

As previously mentioned, UMAP is highly dependent on its number of neighbors hyperparameter. This hyperparameter controls how much at-

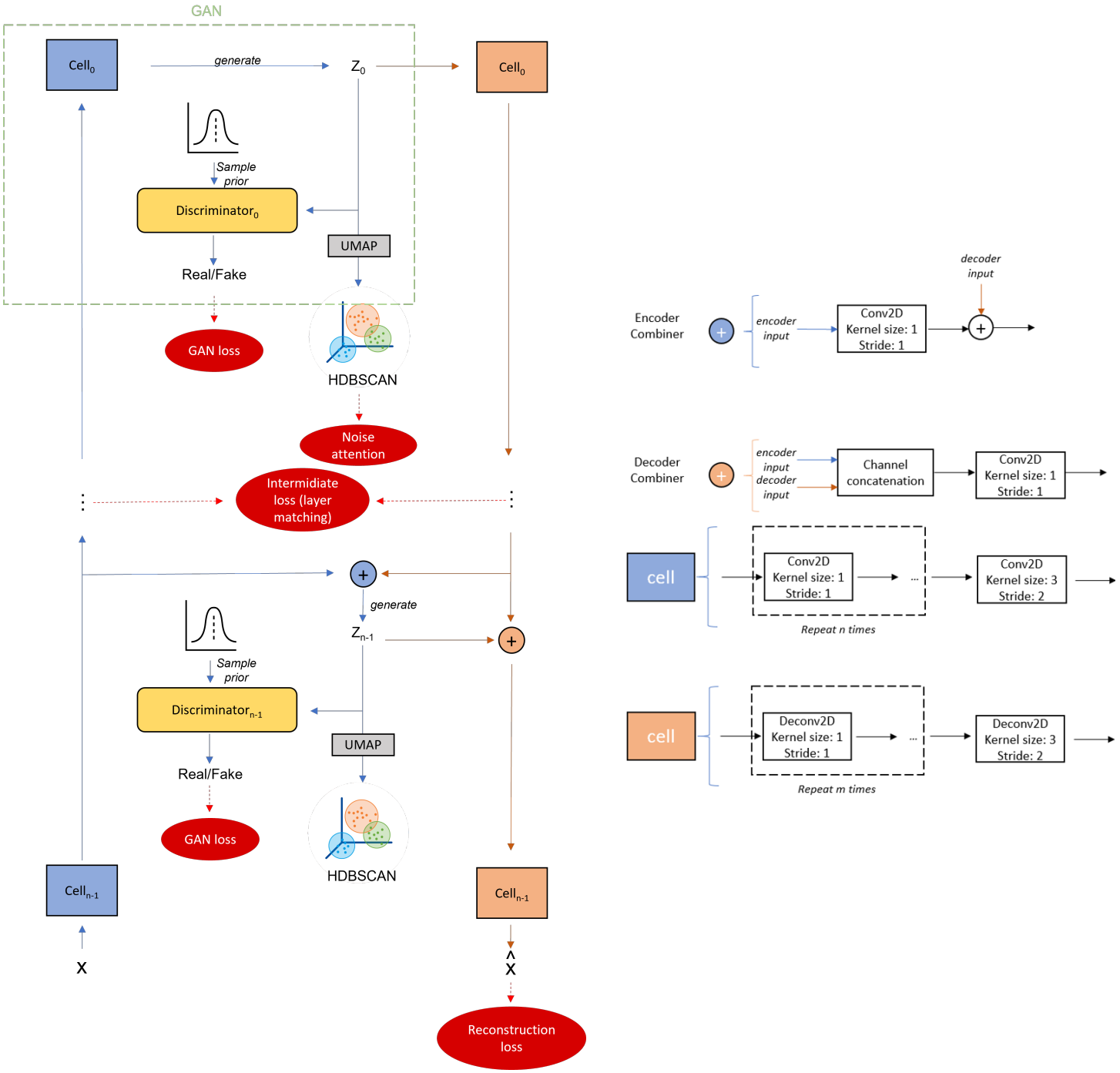


Figure 2: Detailed architecture of HRWAE-GAN. The input x is passed through $N = log_2(d) - 1$ convolutional cells, where $d = \sqrt{D}$ and D is the dimensionality of x (disregarding the number of channels C. The final convolutional cell will consequently and necessarily output a feature map of dimensions 2x2xF, where F is the number of filters of this last cell. Then, the latent variable z_0 is judged by the latent relativistic discriminator $Discriminator_0$, which will compute how realistic z_0 is compared to the prior p_z. The latent variable is then passed trough the N deconvolutional cells and combined with the intermediate feature maps of the encoder. The encoder combiner cells output the remaining latent variables $\{z_1, ..., z_L\}$, which are judged by their respective discriminators $\{Disciminator_1, ..., Discriminator_L\}$. These remaining latent variables are reintegrated in the Decoder with the Decoder combiners. Caption continues in the next page.

In each convolutional and deconvolutional cell, the feature maps may be passed through n and m non-reducing layers, which, as the name suggests, do not reduce the dimensions. Although this should increase the model's ability to understand the input and overall expressiveness in theory, it showed to be unpractical because it increases the instability beyond what allows training to be completed. For this reason, both hyperparameters were set to zero. During training, and as a refinement measure, the latent space at the top latent group (Z_0) is transformed by UMAP and then clustered by HDBSCAN, which identifies noisy points. These noisy points will then receive increased attention during the following epochs. After training, each latent group $\{Z_0, ..., Z_L\}$ can be independently clustered by some clustering algorithm (usually with HDBSCAN or a GMM).

tention is given to local versus global structure. For low values, it will be more focused on capturing local structure; for high values it will be focusing on global structure. Although this is the general idea behind this hyperparameter, what is considered as a high or low value is dependent on the dataset's size. For example, if we consider a dataset with only 100 samples per cluster, then 100 neighbors will provide a global view of the dataset. However, if the dataset contains 1000 samples per cluster, 100 neighbors will provide a more a more localized view.

To deal with this issue, Pedro Távora Santos decided it made sense to have a data dependent number of neighbors and created the following expression:

$$n\_neighbors = min\{\lfloor \frac{dataset\_size}{300} \rfloor, 100\} \quad (8)$$

The 300 in the denominator was determined empirically to deliver the best results, but the optimal value is often dependent on the dataset as well. The number of neighbors is not allowed to exceed 100 due to memory constraints.

It is still important to note that while UMAP does boost clustering performance, there may be some variance in its transforms. This variance may be more or less noticeable depending on the dataset and its properties. For the large image datasets used in this work, the variance is not problematic.

3.3. Loss function

The resulting loss function has 4 components and can be written as:

$$L = L_R + \alpha L_G + \kappa L_I + \beta L_N \quad (9)$$

where L_R is the reconstruction loss, L_G is the GAN loss, L_I is the intermediate loss, L_N is the noise attention and α, β and κ are balancing hyperparemeters. L_G has already been discussed in subsection 3.1.1; the remaining three losses will be described in the following sub subsections. The objective is optimized using the reparameterization trick [11] [19].

3.3.1 Reconstruction Loss

An Autoencoder is composed by an encoder, which takes some data point $x \in R^{d_x}$ and generates an embedding $z \in R^{d_z}$ with $d_z << d_x$, and a decoder, which reconstructs z to an approximation of x. The similarity between x and the reconstruction $\hat{x}$ is of great importance because it is a way of assessing how meaningful the latent embedding z is (if it's not meaningful, then the decoder cannot reconstruct effectively).

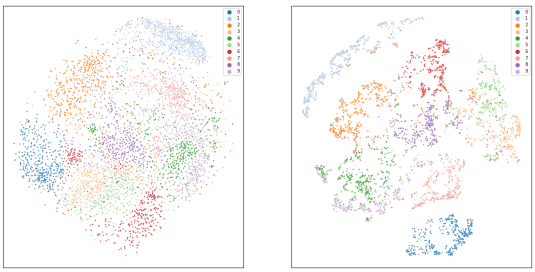


Figure 3: Visualizing MNIST's latent space with TSNE: raw (left) and after UMAP transform (right). It is visually clear that UMAP increases the separation and compactness of the clusters.

To guarantee this similarity, the reconstruction loss L_R is introduced. This loss measures the dissimilarity between x and $\hat{x}$ and is dependent on the dataset. For black and white images the binary cross-entropy can be used because the pixel values are approximately binary, while for more complex colored images the Mean Squared Error (MSE) is preferred.

3.3.2 Intermediate Loss

Experimentally, I found that cluster separation was only being achieved at some of the latent groups in Z. This indicates that the model does not naturally make use of all layers when the number of latent groups L is greater than some amount. I empirically verified that this effect is noticeable when $L > 2$. As discussed in [**?**], this issue was solved through layer matching. Layer matching consists of adding a restraint to ensure that the mid-level feature maps produced by the decoder cells match the ones produced by the encoder cells of their re-

spective level. Formally, this can be written as:

$$L_I = \sum_{l=1}^{L-1} D(i^l_{encoder}, i^{l-1}_{decoder}) \tag{10}$$

where l denotes some latent group (the top group, L is the total number of latent groups, D is some error function such as the Cosine Dissimilarity, and $i_{encoder}$ and $i_{decoder}$ are the intermediate feature maps.
The effect of this constraint will be demonstrated in the experiments subsection.

3.3.3 Noise attention

HDBSCAN is a density based clustering algorithm with the particularity of noise identification (it does not classify points which it considers as noise). Noisy points can arguably be considered as points that are not correctly embedded, as they are not within reasonable distance from other points from the same class (recall that density clustering algorithms consider two data points x_1 and x_2 to be from the same if $d_{1,2} < \epsilon$, where $d_{1,2}$ is the distance between the two points and ϵ is some minimum distance). For this reason, it makes sense to identify these noisy points and increase the model's attention towards them. This results in a diminishing of noise in the latent spaces, as will be demonstrated in the experiments subsection.

This noise attention can be formalized as:

$$L_N = \sum_{n=0}^{N-1} \begin{cases} 1 & if \quad x_n \in Noise \\ 0 & otherwise \end{cases} \tag{11}$$

where $Noise$ is the set of points that are classified as noise by HDBSCAN. The percentage of increased attention is determined by $\kappa - 1$, where κ is the hyperparameter used in equation (9).

4. Results

4.1. Number of latent groups

This subsection will be used to analyse the effects of varying the number of latent groups. This analysis will be based on three factors: clustering performance, quality of image reconstruction and quality of image generation. Note that when the number of latent groups L equals 1, the model drops its hierarchical structure and becomes a regular Wasserstein Autoencoder GAN.

In table 1 we can see how adding an extra latent group affects clustering performance. In latent group 0, clustering is slightly worse than the non-hierarchical model but without significant increase in variance. In latent group 1, however, results are a bit more interesting. When it comes to the average results, they are significantly worse and with very large standard deviation. This shows how the hierarchical architecture can have very high variance between different instances of the model. Even though the best accuracy result was achieved in the second latent group, the high variability makes it unusable in a fully unsupervised environment since there is no way of knowing how reliable an instance of the model is.

So, at least for this particular implementation, **using a hierarchical architecture did not present any advantages when it comes to clustering tasks**.

Nonetheless, in figure 4 it's visible that the quality of image reconstruction increases significantly with additional latent groups. Unfortunately, since the only use of reconstruction quality is to have an idea of how trustworthy the latent representations are and we've determined that hierarchical models do not increase clustering performance, this positive aspect is basically insignificant.

Finally, we'll take a look at the effect of multiple hierarchical groups on the generative capabilities of HRWAE-GAN. In figure 5, it's apparent that adding a latent group blurred the generated images, even though the FID score somehow improves. This was surprising because the authors of NVAE [22], who used a hierarchical VAE to generate images, obtained opposite results. Since the biggest difference between HRWAE-GAN and NVAE is the regularization of the latent spaces (HRWAE-GAN utilizes a GAN framework), I believe this could possibly mean that **mixing GANs with a hierarchical architecture is either not beneficial or requires a lot of fine tuning**.

4.2. Comparison of clustering performance with other state-of-the-art models

Table 2 shows how HRWAE-GAN compares to both MoE and the current best performing model for each one of the three experimented on datasets. It's visible that the attained model is outperformed by a significant margin on all cases, but especially of the CIFAR-10 dataset.

4.3. Discussion

All things considered, HRWAE-GAN's performance fell below expectations. After all the time I have spent with it, I can confidently say that the added complexity of multiple latent groups is not rewarding. Apart from its results not being up to par with the current state-of-the-art, it also suffers from three main issues:

1. As the number of latent groups L increases, training becomes considerably more unstable and often diverges before completion. This is why my experiments were mostly done with $L = 2$. This is unfortunate because it would have been interesting to see what would happen with larger Ls.

Table 1: Clustering results on the Fashion MNIST dataset for different amounts of latent groups (first column) and for each latent group (second column). The columns tagged with "(best)" correspond to the values of the model with highest total clustering accuracy.

#latent groups	latent group	#clusters (best)	%labeled (avg)	acc labeled (avg)	totall acc (avg)	%labeled (best)	acc labeled (best)	total acc (best)
1	0	10	68.42% +/- 4.6%	62.1% +/- 2.1%	47.84% +/- 5%	68.9%	63.4%	51.71%
2	0	10	56.4% +/- 4.26%	69% +/- 5.6%	44.2% +/- 2.98%	60.4%	72.8%	49.6%
	1	10	41.86% +/-28.9%	40% +/- 21%	29.38% +/- 20.1%	84.3%	65.1%	56.8%

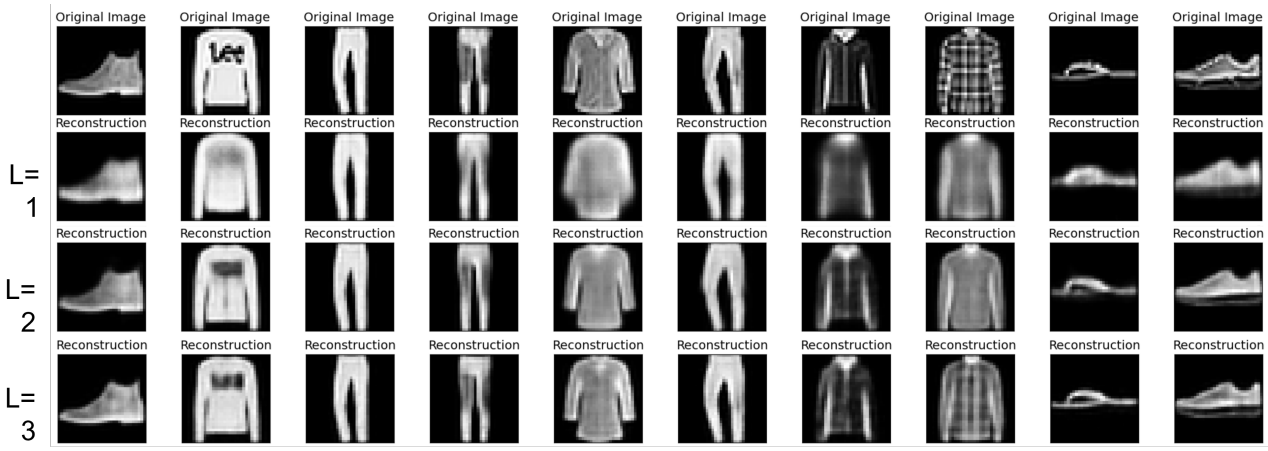
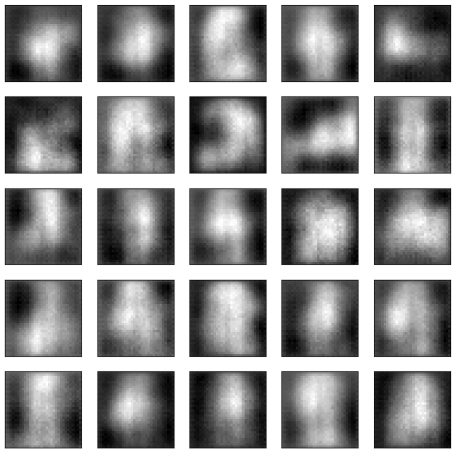


Figure 4: In this image we can see how reconstruction quality improves with the number of latent groups L.

Images generated by HRWAE-GAN with a single latent group ($L = 1$).
Average FID = 32.95 +/- 0.71



Images generated by HRWAE-GAN with two latent groups ($L = 2$).
Average FID = 30.76 +/- 2.24

Figure 5: Images generated by HRWAE-GAN. FID scores were obtained by averaging 5 runs (each run corresponds to a different instance of the model).

Table 2: Clustering results comparison between a two latent group HRWAE-GAN, the Mixture of Experts and the best performing model for each datataset - SPC [13], AE + SNNL [1] and SPICE [17].

		#clusters	%labeled (avg)	acc labeled (avg)	totall acc (avg)	%labeled (best)	acc labeled (best)	total acc (best)
MNIST	HRWAE-GAN GMM (L=2)	-	-	-	70.37% +/- 4.5%	-	-	77%
	HRWAE-GAN HDBSCAN (L=2)	10	55.2% +/- 10.3%	85.6% +/- 7%	54.9% +/- 6.6%	78.86%	94.14%	78.8%
	MoE	10	-	-	97.4%	-	-	-
	SPC	-	-	-	99.03% +/- 0.1%	-	-	-
FMNIST	HRWAE-GAN GMM (L=2)	-	-	-	60% +/- 3.4%	-	-	63.9%
	HRWAE-GAN HDBSCAN (L=2)	10	56.4% +/- 4.6%	69% +/- 5.6%	44.2% +/- 2.98%	84.3%	65.1%	56.8%
	MoE	10	-	-	68%	-	-	-
	AE + SNNL	-	-	-	84.4%	-	-	-
CIFAR10	HRWAE-GAN GMM (L=2)	-	-	-	14.7% +/-2.2%	-	-	16.48%
	HRWAE-GAN HDBSCAN (L=2)	2	2.9% +/- 18.3%	22% +/- 11.5%	15.7% +/- 2.8%	58%	21%	17.6%
	MoE	3	-	-	32.8%	-	-	-
	SPICE	-	-	-	91.7%			

2. The model suffers from training variance, meaning that two models trained with the same hyperparameters will output differently organized latent spaces and generate images of varying quality. This makes it very hard to interpret how hyperparameters and different added constraints effectively change the outcome of training.

3. The latent space regularization through adversarial training is also a hard task. While the relativistic discriminator heavily increases training stability, the loss evolution can be hard to interpret because it is not a simple minimization problem. This makes it harder to know what to change when results are sub par.

According to recent state-of-the-art works like SPICE [17] and SPC [13], semi-supervised approaches easily outperform hierarchical ones such as this one and are easier to implement and experiment with (the implementation of this model was very challenging).

5. Conclusions

In this Dissertation, I had the goal of understanding the current state-of-the-art deep clustering techniques and subsequently following a novel approach to tackle this difficult task. The result was HRWAE-GAN, a generative autoencoder with a hierarchical latent space (it contains a set of latent spaces $Z = \{z_0, ..., z_{L-1}\}$, instead of just one) that is regularized by a set of latent relativistic discriminators in an adversarial training setup.

Overall, this model is very complex and challenging to implement without the reward of state-of-the-art performance. Although the hierarchical latent structure has shown to be beneficial for data generation [22], this did not seem to be the case for clustering. Initially, there was hope that having several latent spaces could possibly be used in some advantageous way. This could happen if the model was consistent in its way of organizing each latent space (i.e, it always utilized the same logic in z_0, z_1, etc. to organize the data points), but that was not the case. It's also important to mention that the added complexity of the hierarchical architecture makes training significantly more unstable, even when using the regularization techniques recommended in NVAE[22].

Given that much better results have been achieved by recent works like SPC [13] with much simpler and more intuitive architectures, the path followed in this thesis does not seem to be recommendable.

References

[1] A. F. Agarap and A. P. Azcarraga. Improving k-means clustering performance with disentangled internal representations. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[2] M. Arjovsky and L. Bottou. Towards princi-

pled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

[3] C. M. Bishop. Pattern recognition. *Machine learning*, 128(9), 2006.

[4] R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[5] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1):1–51, 2015.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[8] W. Harchaoui, P.-A. Mattei, and C. Bouveyron. Deep adversarial gaussian mixture autoencoder for clustering. 2017.

[9] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[10] A. Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.

[13] L. Mahon and T. Lukasiewicz. Selective pseudo-label clustering, 2021.

[14] A. M. Martinez and A. C. Kak. Pca versus lda. *IEEE transactions on pattern analysis and machine intelligence*, 23(2):228–233, 2001.

[15] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock. N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5145–5152. IEEE, 2021.

[16] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[17] C. Niu and G. Wang. Spice: Semantic pseudo-labeling for image clustering, 2021.

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[19] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[20] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

[21] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[22] A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

[23] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[24] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016.

[25] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[26] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.