Representation Learning with Adversarial Latent Autoencoders

Stanislav Pidhorskyi M.S.
West Virginia University, stpidhorskyi@mix.wvu.edu

REPRESENTATION LEARNING WITH ADVERSARIAL LATENT AUTOENCODERS

Stanislav Pidhorskyi

Dissertation submitted to the
Benjamin M. Statler College of Engineering and Mineral Resources
at West Virginia University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in
Computer Science

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2020

Keywords: Computer Vision, Machine Learning, Deep Learning, Autoencoder, Generative Adversarial Network, Image Autoencoding, Novelty Detection, Anomaly Detection, Similarity Search, Similarity Retrieval

ABSTRACT

Representation Learning with Adversarial Latent Autoencoders

Stanislav Pidhorskyi

A large number of deep learning methods applied to computer vision problems require encoder-decoder maps. These methods include, but are not limited to, self-representation learning, generalization, few-shot learning, and novelty detection. Encoder-decoder maps are also useful for photo manipulation, photo editing, superresolution, etc. Encoder-decoder maps are typically learned using autoencoder networks.

Traditionally, autoencoder reciprocity is achieved in the image-space using pixel-wise similarity loss, which has a widely known flaw of producing non-realistic reconstructions. This flaw is typical for the Variational Autoencoder (VAE) family and is not only limited to pixel-wise similarity losses, but is common to all methods relying upon the explicit maximum likelihood training paradigm, as opposed to an implicit one. Likelihood maximization, coupled with poor decoder distribution leads to poor or blurry reconstructions at best.

Generative Adversarial Networks (GANs) on the other hand, perform an implicit maximization of the likelihood by solving a minimax game, thus bypassing the issues derived from the explicit maximization. This provides GAN architectures with remarkable generative power, enabling the generation of high-resolution images of humans, which are indistinguishable from real photos to the naked eye. However, GAN architectures lack inference capabilities, which makes them unsuitable for training encoder-decoder maps, effectively limiting their application space.

We introduce an autoencoder architecture that (a) is free from the consequences of maximizing the likelihood directly, (b) produces reconstructions competitive in quality with state-of-the-art GAN architectures, and (c) allows learning disentangled representations, which makes it useful in a variety of problems. We show that the proposed architecture and training paradigm significantly improves the state-of-the-art in novelty and anomaly detection methods, it enables novel kinds of image manipulations, and has significant potential for other applications.

I dedicate my dissertation work to my family and friends. I dedicate this work to my loving parents, Natalia Pidhorska and Yuriy Pidhorskyi, for their love, support, belief in me, and sacrifice. I dedicate this work to my wonderful daughter Valeria. I also dedicate this work and thank my uncle Petro Fomychov. I dedicate this work and give special thanks to Irene Derzhko for her support.

Acknowledgments

I want to especially thank my PhD advisors, Dr. Gianfranco Doretto and Dr. Donald Adjeroh, for their advice and support, as well as all my labmates. I must also acknowledge all my teachers during my student life for their encouragement and motivation.

This work was supported in part by funding from the US National Science Foundation (Award #1761792 and Award #1920920).

Contents

List of Figures

List of Tables

Chapter 1

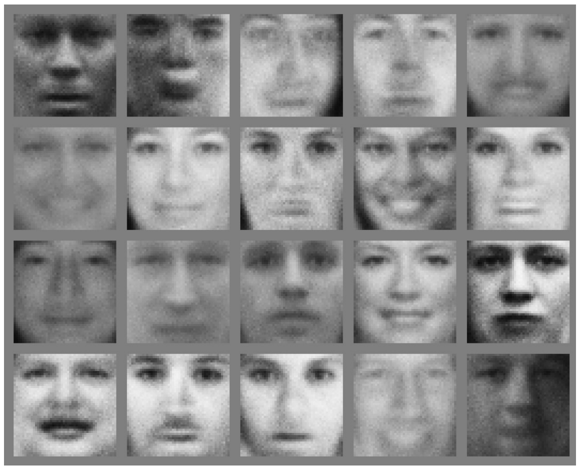
Introduction

1.1 Problem Definition

An important cornerstone of the deep learning paradigm in modern computer vision is autoencoders. Autoencoder consists of encoder-decoder maps that can map input image to some compressed, hight-level representations and then map them back to the image space. Autoencoders became the core of many methods applied to computer vision problems, which include, but are not limited to, self-representation learning, generalization, few-shot learning, and novelty detection. The modern variant of the autoencoder is also generative, meaning that the decoder network is also capable of generating samples, and is often referred to as a generator network, rather than a decoder network. Autoencoders combine generative and representational properties by learning an encoder-generator map.

However, generative power and quality of reconstructions of autoencoders lag behind those of modern generative models.

In recent years GANs [1] gained remarkable generative power, see Figure 1.1. They allow learning **generator** maps from fixed, known distributions by observing samples from the target distribution. The **generator** map is such map that maps a known, easy to sample distribution, such as normal distribution, to a distribution which resembles the target


(a) Synthetic images from original GAN paper by Goodfellow et al. [1]

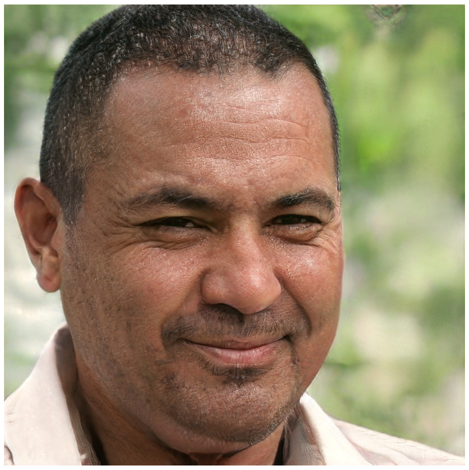

(b) Synthetic image from StyleGAN by Karras et al. [2]

Figure 1.1: **Synthetic images generated by GANs**. Comparison of images produced by vanilla GAN 1.1a and StyleGAN 1.1b

distribution.

GANs can learn this target distribution in an unbiased way and perform implicit likelihood maximization. We will discuss what implicit maximization is and why it is unbiased in the following sections.

One of the most notable steps forward in improving GANs quality was StyleGAN [2], which enabled the generation of high-resolution images of human faces, which are indistinguishable from real photos to the naked eye. With all these good qualities of GANs, they lack inference capabilities. GANs learn target distribution is an implicit way, through a the **generator** map, which makes it not possible to estimate the probability density of a given sample or find a reverse mapping of the given sample to the fixed, known distribution.

On the other hand, fully implicit likelihood maximization is not possible for autoencoders due to reciprocity requirements, and as a result, this leads to poor or blurry reconstructions at best.

Can we take the best from two worlds and develop a unified architecture that would combine generative and representational properties of autoencoder as well as the generative power of GANs?

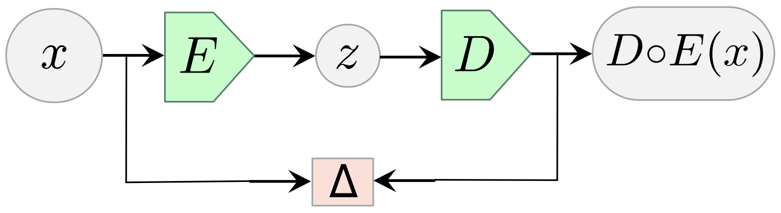


Figure 1.2: **Plain Autoencoder**. Architecture of vanila autoencoder, reciprocity enforced in data space with MSE loss. Here x is input sample, E is encoding map, D is decoding map, z - latent vector, Δ - reciprocity loss.

1.2 Motivation and Challenges

In this dissertation, we try to develop a new autoencoder architecture that would provide a combination of the generative power of GANs and representational properties of autoencoders. We also explore the application space of such architecture and demonstrate how it can be used to advance the state of the art methods in Computer Vision.

But before, let us discuss the key aspects and problems of autoencoders that prevent them from having high-quality generations, and the reasons why GANs managed to overcome them.

1.2.1 Vanila Autoencoder

Simple, feedforward autoencoders (AE) were known since late 80s [3, 4, 5], and popularized later by Salakhutdinov et al. [6]. Vanilla AE consists of encoder-decoder maps, and it is trained to replicate its input on its output, see Figure 1.2. Traditionally, reciprocity in AE is achieved in the image-space using pixel-wise similarity loss, which has a widely known flaw of producing non-realistic reconstructions.

Most commonly, MSE loss is used as a similarity loss. Autoencoder is trained to minimize expected MSE of reconstruction 1.1

$$\arg\min_{\phi,\theta} E_{\mathbf{x}\sim p_{data}} \left\|\mathbf{x} - D_\theta \circ E_\phi(\mathbf{x})\right\|^2 \tag{1.1}$$

where p_{data} is data distribution, ϕ, and θ - parameters of encoder map E and decoder map D respectively.

On the contrast, generative models are commonly trained to match target distribution using likelihood maximization paradigm: $\arg\max_{\theta} E_{\mathbf{x}\sim p_{data}} \log p_\theta(\mathbf{x})$, where $p_\theta(\mathbf{x})$ is a probability distribution describing generator.

We can rewrite AE formulation in terms of likelihood maximization paradigm:

$$\arg\max_{\phi,\theta} E_{\mathbf{x}\sim p_{data}} \log p_\theta(\mathbf{x}|\mathbf{z}), \mathbf{z} = E_\phi(\mathbf{x}) \tag{1.2}$$

And thus, if $p_\theta(\mathbf{x}|\mathbf{z})$ is a factorized Gaussian $D_\phi(\mathbf{z}) \sim \mathcal{N}(\mathbf{x}|\mu_{\theta(\mathbf{z})}, \mathbf{I})$, then be can derive that:

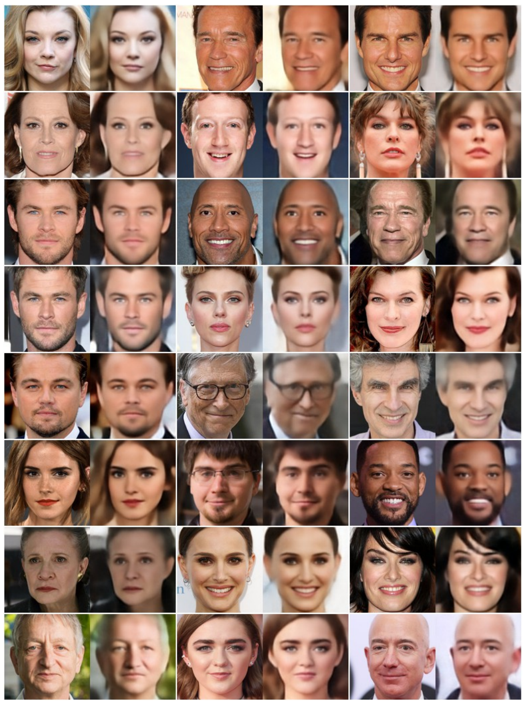
$$-\log p_\theta(\mathbf{x}|\mathbf{z}) = \tfrac{1}{2}\left\|\mathbf{x} - \mu_\theta(\mathbf{z})\right\|_2^2 \tag{1.3}$$

In such a way, we can see that MSE minimization is a particular case of the likelihood maximization paradigm when we assume decoder distribution to be a factorized Gaussian.

It is known that the explicit maximum likelihood training paradigm has a well-recognized issue [7]. Thus: (a) explicit likelihood maximization and (b) poor decoder distribution (observational model), e.g. factorized Gaussian lead to **blurry** and unrealistic results.

Virtually, AE produces *expectation of plausible samples*, but not *a plausible one*. In fact, outputs are not even blurry but noisy. That is because, the majority of implementations tend to display the mean of $p_\theta(\mathbf{x}|\mathbf{z})$, rather than drawing samples from it.

Since 1.2 is not tracktable, we need to make simplification in order to transform 1.2 into something feasible (such as 1.9). Due to the simplifications and relaxation of the original objective, the resulting method becomes biased. Even we do not use pixel-wise loss, which is an extremely bad type of loss for the image domain but replace it with perceptual loss,


(a) Reconstructions by vanilla autoencoder.

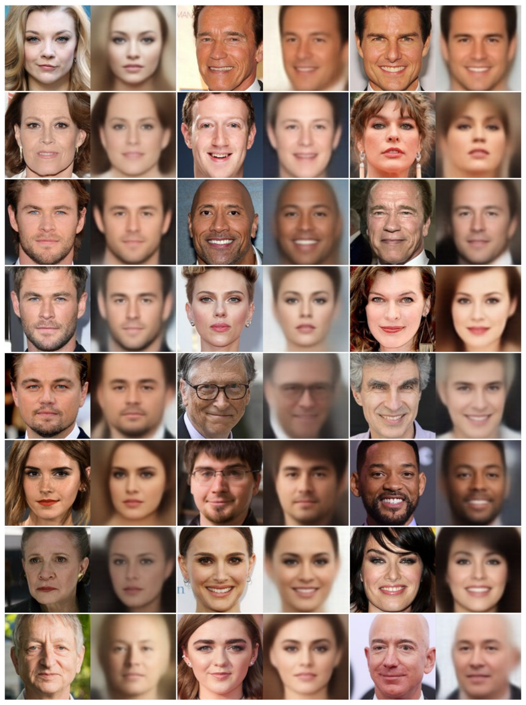

(b) Reconstructions by VAE [9].

Figure 1.3: **Reconstructions with AE and VAE**. Comparison of reconstructions done by vanilla AE 1.4a and VAE [9] 1.4b

such as LPIPS [8], that still does not solve the problem.

Explicit likelihood maximization and blurriness is typical for VAE [9], but as we see, not limited to it. It is a common trait of all methods that utilize explicit likelihood maximization, such as AAE [10], VampPrior [11], etc.

Explicit likelihood maximization problem typically arises in AE when reciprocity is achieved with a similarity criterion in data-space, even if the similarity is perceptual.

1.2.2 Variational Autoencoders

In order to support the ideas described in the previous section, let us focus on variational autoencoder. Variational AE architectures [12, 13] have been appreciated for their theoretical foundation and their stability and ease of training. On the construct to AE, VAE is first of all a generative model. The foundation of VAE starts from maximizing the likelihood, which is typical for generative models:

$$\arg\max_{\theta} E_{\mathbf{x}\sim p_{data}} \log p_\theta(\mathbf{x}) \tag{1.4}$$

The key idea of VAE is that since optimizing 1.6 is not feasible, we are going to optimize evidence lower bound (ELBO) of 1.6. This is where bias is introduced into VAEs. Optimizing lower bound is definitely a good approach to the problem, but it is unavoidably not the same as optimizing 1.6

Using ELBO, we can arrive to:

$$\arg\max_{\theta} E_{\mathbf{x}\sim p_{data}} \left[E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))\right] \tag{1.5}$$

Or, we can rewrite that into minimization of the sum of two losses:

$$\arg\min_{\theta} E_{\mathbf{x}\sim p_{data}} \left[\mathcal{L}_{REC} + \mathcal{L}_{KL}\right] \tag{1.6}$$

where

$$\mathcal{L}_{REC} = -E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) \tag{1.7}$$

$$\mathcal{L}_{KL} = KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \tag{1.8}$$

We can see now that even after using ELBO we end up with non-feasible optimization problems. The reconstruction part $\mathcal{L}_{REC}$ 1.7 is not tractable and we need to make simplifications. The majority of practical implementations of VAEs use factorized Gaussian with covariance matrix set to identity matrix for the observation model, which brings us back to pixel-wise MSE loss 1.9:

$$\mathcal{L}_{REC} = \|\mathbf{x} - D_\theta \circ E_\phi(\mathbf{x})\|_2^2 \tag{1.9}$$

As we can see, in practice, explicit likelihood maximization leads to image space similarity loss at best, and often it is implemented as pixel-wise MSE. This results into blurred images and this flaw is not specific to VAE, but is shared across all AE that approach reciprocity by enforcing image space similarity.

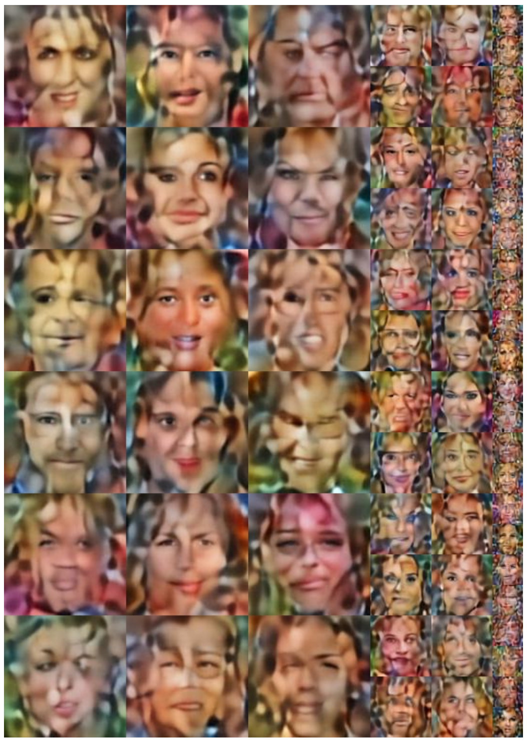
From Figure 1.3 we can see that reconstructions are not just blurry, but also shift to the mean face occurs. AE outputs an expected image, which is quite far from a realistic one. Since VAE also enforces information bottleneck [14], reconstructions are less similar to the input. One can clearly see that it preserves high frequency only for those features that are the most predictable, thus require less information to encode, such as eyes, nose, mouse. On the other hand, those parts of the image that are less predictable, such as background, hair, face contour lost the most of high frequencies.

From Figure 1.4, we can see that despite that AE produces slightly sharper results than VAE, we can not sample it's latent space to generate images. The reason is that the latent space is not compact because AE didn't need to utilize the whole capacity of the channel, in contrast to VAEs.

In this study we focus on the following three problems:

- Adversarial Latent Autoencoders.
- Novelty detection.
- Large scale similarity retrieval via hashing

We develop a new autoencoder architecture that overcomes current limitations of autoencoders and then demonstrate it's applicability to several state of the art methods in Computer Vision.


(a) Sampling vanilla autoencoder.

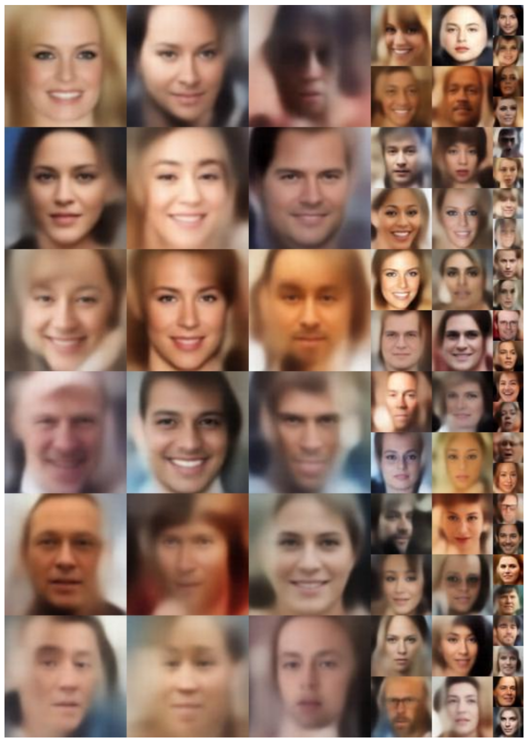

(b) Sampling VAE [9].

Figure 1.4: **Generations with AE and VAE**. Comparison of sampling samples from vanilla AE 1.4a and VAE [9] 1.4b

1.3 Related work

1.3.1 Autoencoders

Our approach builds directly on the vanilla GAN architecture [15]. Since then, a lot of progress has been made in the area of synthetic image generation. LAPGAN [16] and StackGAN [17, 18] train a stack of GANs organized in a multi-resolution pyramid to generate high-resolution images. HDGAN [19] improves by incorporating hierarchically-nested adversarial objectives inside the network hierarchy. In [20] they use a multi-scale generator and discriminator architecture to synthesize high-resolution images with a GAN conditioned on semantic label maps, while in BigGAN [21] they improve the synthesis by applying better regularization techniques. In PGGAN [22] it is shown how high-resolution images can be synthesized by progressively growing the generator and the discriminator of a GAN. The same principle was used in StyleGAN [2], the current state-of-the-art for face image generation, which we adapt it here for our StyleALAE architecture. Other recent work on GANs has focussed on improving the stability and robustness of the training [23]. New loss functions have been introduced [24], along with gradient regularization methods [25, 26], weight normalization techniques [27], and learning rate equalization [22]. Our framework is amenable to these improvements, as we explain in later sections.

Variational AE architectures [12, 13] have not only been appreciated for their theoretical foundation, but also for their stability during training, and the ability to provide insightful representations. Indeed, they stimulated research in the area of disentanglement [28], allowing learning representations with controlled degree of disentanglement between factors of variation in [14], and subsequent improvements in [29], leading to more elaborate metrics for disentanglement quantification [30, 31, 2], which we also use to analyze the properties of our approach. VAEs have also been extended to learn a latent prior different than a normal distribution, thus achieving significantly better models [32].

A lot of progress has been made towards combining the benefits of GANs and VAEs.

AAE [10] has been the precursor of those approaches, followed by VAE/GAN [33] with a more direct approach. BiGAN [34] and ALI [7] provide an elegant framework fully adversarial, whereas VEEGAN [35] and AGE [36] pioneered the use of the latent space for autoencoding and advocated the reduction of the architecture complexity. PIONEER [37] and IntroVAE [38] followed this line, with the latter providing the best generation results in this category. Section 2.3.1 describes how the proposed approach compares with those listed here.

Finally, we quickly mention other approaches that have shown promising results with representing image data distributions. Those include autoregressive [39] and flow-based methods [40]. The former forego the use of a latent representation, but the latter does not.

1.3.2 Novelty Detection

Novelty detection is a task of recognizing whether the given sample is inlier or outlier with respect to the training data. The literature in this area is sizable. Novelty detection methods can be split into four large groups: probabilistic, reconstruction-based, density estimation methods, and out-of-distribution methods.

Probabilistic methods [41, 42, 43, 44] investigate ways to compute the density function of normal, not novel data. The main difficulty concerns how to compute the densities of high-dimensional and complex data. Typically it is accomplished with different strategies of approximation of density function or lowering dimensionality of the space.

Kernel-based methods learn null space of training data and rely on distance measure to perform density estimation implicitly, [45] introduced the Kernel Null Foley-Sammon Transform (KNFST) for multi-class novelty detection, where training samples of each known category are projected onto a single point in the null space and then distances between the projection of a test sample and the class representatives are used to obtain a novelty measure. [46] improves on previous approaches by proposing an incremental procedure called Incremental Kernel Null Space-Based Discriminant Analysis (IKNDA). Modern approaches rely on deep representations typically obtained with autoencoders [47, 48].

Reconstruction-based methods. Since outliers do not have sparse representations, self-representation approaches have been proposed for outlier detection in a union of subspaces [49, 50]. Similarly, deep learning based approaches have used neural networks and leveraged the reconstruction error of encoder-decoder architectures. [51, 52] used deep learning based autoencoders to learn the model of normal behaviors and employed a reconstruction loss to detect outliers.

[53] used a GAN [54] based method. Despite the fact that an encoder is not available in GAN setup, this method is still reconstruction based. They used a generator to recover latent representation with gradient descent by optimizing reconstruction error, which then is used as a novelty score. [55] trained GANs using optical flow images to learn a representation of scenes in videos. [56] minimized the reconstruction error of an autoencoder to remove outliers from noisy data, and by utilizing the gradient magnitude of the auto-encoder they make the reconstruction error more discriminative for positive samples. In [57] they proposed a framework for one-class classification and novelty detection. It consists of two main modules learned in an adversarial fashion. The first is a decoder-encoder convolutional neural network trained to reconstruct inliers accurately, while the second is a one-class classifier made with another network that produces the novelty score.

Computing reconstruction error in image space is not ideal, in fact, l2 norm works poorly with images. [58] uses as a novelty score not only reconstruction error in image space, but also in hidden spaces. They pass the reconstructed image to the encoder and observe activations of all the intermediate layers in the encoder and compare those to activations induced by the original image. [59] extends this approach by adding adversarial loss that matches the distribution of hidden activations for real and reconstructed inliers.

Density estimation methods. State-of-the-art works on density estimation for image compression include Pixel Recurrent Neural Networks [60] and derivatives [61, 62]. These pixel-based methods allow us to sequentially predict pixels in an image along the two spatial dimensions. Because they model the joint distribution of the raw pixels along with their sequential correlation, it is possible to use them for image compression. Although they could also model the probability distribution of known samples, they work at a local scale in a patch-based fashion, which makes non-local pixels loosely correlated. Our approach, instead, does not allow modeling the probability density of individual pixels but works with the whole image. It is not suitable for image compression, and while its generative nature allows in principle to produce novel images, in this work, we focus only on novelty detection by evaluating the inlier probability distribution on test samples.

Out-of-distribution methods. These are methods that typically improve robustness of existing classification or detection systems, in order to detect erroneous sample that otherwise would be classified incorrectly. A recent line of work has focused on detecting out-of-distribution samples by analyzing the output entropy of a prediction made by a pre-trained deep neural network [63, 64, 65, 66, 67]. This is done by either simply thresholding the maximum softmax score [64], or by first applying perturbations to the input, scaled proportionally to the gradients w.r.t. to the input and then combining the softmax score with temperature scaling, as it is done in Out-of-distribution Image Detection in Neural Networks (ODIN) [66]. While these approaches require labels for the in-distribution data to train the classifier network, our method does not use label information. Therefore, it can be applied for the case when in-distribution data is represented by one class or label information is not available. Despite the fact that out-of distribution methods solve problem that is similar to the novelty detection problem, the evaluation protocol differs significantly, making it hard to compare to novelty detection methods. Under novelty detection setting, typically, one class of the dataset is used as inlier set, and all other classes are used as outlier set. Distinctively, out-of-distribution methods are evaluated typically using one dataset as inliers and other dataset as outliers [63, 64, 65, 66, 68, 67]. OOD methods can not be applied to the setting typical for novelty detection, since they require labels inside the inlier dataset. Using different datasets as inlier and outliers creates domain shift, those datasets may have different statistics, distributions, etc.. For example, ODIN [66] uses CIFAR10 and CIFAR100 as inliers, with TinyImageNet, iSUN, LSUN as outliers. [67] also includes scenarios of using CIFAR10 vs SVHN vs Caltech256, etc and [68] does MNIST vs FashionMNIST. Obviously, MNIST and FashionMNIST are extremely different, similarly CIFAR10 and SVHN are extremely different, which of course is a desirable property for a classifier to detect such cases instead of returning a random label. Sometimes, it can go to such extremes as taking images of CT scans as inliers and images of cats and dogs as outliers [69]. Indeed, resizing ImageNet images to the same resolution as in CIFAR10 may create different image statistics, unless resizing is used precisely with the same filters. On the other hand, labels of CIFAR10 and CIFAR 100 largely overlap with ImageNET. Thus, OOD methods are more targeted to detect samples coming from completely different distributions and are more associated with robustness of classification/detection systems, while novelty detection methods are more targeted at detecting samples coming from the same or similar distribution but representing novel, not previously observed class. Thus, OOD makes emphasis on reliability, while novelty detection makes emphasis on discovery of novel classes. These differences between OOD and novelty detection, and inability of OOD to run on the same benchmarks as novelty detection does, makes comparison between OOD and novelty detection methods complicated and often not fair.

Latent code density estimation. Reconstruction based methods take into account only the reconstruction error, in image or feature space, which may not be enough. Using deep autoencoder architectures also allows us to examine density of latent representation. [70] equip autoencoder with a density estimator of the latent codes that learns the probability distribution of latent codes of the inlier data. A combination of the density estimation of the latent code plus reconstruction error is used as a novelty score. In this way, this method relates to both reconstruction-based and probabilistic methods.

The proposed approach relates to the probabilistic methods and reconstruction-based methods the same way as [70]. Similarly to probabilistic methods, it aims at computing the probability distribution of test samples as novelty scores, but it does so by learning the manifold structure of the distribution with an encoder-decoder network.

The proposed method, similarly to [70] relies on the density estimation of the latent code of the sample as well as the reconstruction error. In our framework, we begin with a probabilistic approach and attempt to derive a way to estimate the probability density of the sample, and through series of assumptions, we make it possible to factorize the probability density of the sample into the probability of the latent code and the probability density of the reconstruction error. In such a way, two factorized probability densities can be united in a natural way into a single novelty score. Though the ideas of the proposed method go very close to [70], the proposed method (GPND) was initially presented as a conference paper in [71] before [70].

1.3.3 Similarity Retrieval

The existing variety of data-dependent, learning based hashing methods can be categorized into unsupervised and supervised methods [72].

Unsupervised methods [73, 74, 75, 76, 77, 78, 79, 80] use unlabeled data to learn a hashing function that preserves some metric distance between data points. This work instead falls into the supervised category, which tries to improve the quality of hashing by leveraging label information to learn compact codes. Supervised methods can be divided into those which use off-the-shelf visual features versus those that leverage deep networks. Representative examples of non-deep methods include Minimal Loss Hashing (MLH) [81], Supervised Hashing with Kernels (KSH) [82] and Latent Factor Hashing (LFH) [83].

The initial attempts to utilize deep learning [84, 85, 86, 87] for hashing included CNNH [88] and DNNH [89]. Deep Hashing Network (DHN) [90] and Deep Supervised Hashing (DSH) [91] extend DNNH by performing a continuous relaxation of the intractable discrete optimization by introducing a quantization error prior which is controlled by a quantization loss. DHN uses a cross entropy loss to link the pairwise Hamming distances with the pairwise similarity labels, while DSH uses max-margin loss. Deep Cauchy Hashing (DCH) [92] improves DHN by utilizing Cauchy distribution. Deep Pairwise-Supervised Hashing (DPSH) [93] uses pairwise similarity labels and a loss-function similar to LFH, which maximizes the log-likelihood of the pairwise similarities. The log-likelihood is modeled as a function of the Hamming distance between the corresponding data points. Deep Triplet-Supervised Hashing (DTSH) [94] extends DPSH by using triplet label information.

The learning problem of deep hashing methods (DHN, DSH, DPSH, DTSH, etc.) turns out to be NP-complete, due to the discrete nature of the codomain of the hash function being sought, which is the Hamming space. The workaround is to relax the native space into the continuous Euclidean counterpart. This makes the original learning problem ill-posed, and a regularizing prior becomes necessary, which is often chosen to encourage the sought mapping to produce a nearly binary output. While needed, such prior complicates the training and might lead to performance reduction. Discrepancy Minimizing Deep Hashing (DMDH) [95] suggests an alternating optimization approach with a series expansion of the objective. Our work instead, leverages a different relaxation, which has the advantage of maintaining a well-posed learning problem, without the need for extra priors. Besides the obvious computational advantage, the framework allows to identify a class of triplet losses, and to define new ones, tailored to seeking good hash functions.

Another line of work, like Deep Quantization Network (DQN) [96], avoid the use of relaxation by incorporating quantization methods into the approach [97, 98, 99]. DQN performs a joint learning of image representations and a product quantization for generating compact binary codes. Deep Visual-Semantic Quantization (DVSQ) [100] extends DQN by adding semantic knowledge extracted from the label space. In this way, hash codes are directly optimized. While being an interesting direction, it significantly increases the complexity of the learning process. Indeed, that might be one of the contributing factors that make our approach comparing favorably against those.

Finally, our approach also considers the quantization problem. Indeed, the proposed relaxation suggests learning a spherical embedding, which is an equivalence class of solutions, because the loss turns out to be rotation invariant with respect to the embedded spherical space. This allows picking, as a solution, a representative of the class that will affect the quantization of the spherical embedding in such a way that it directly maximizes the mean average precision. This is different from previous approaches, and it is different also from approaches like Iterative Quantization (ITQ) [74], which is unsupervised, and it aims at minimizing the quantization error. Our comparison with ITQ shows that linking the quantization directly to the retrieval metric leads to better solutions.

1.4 Contribution and Dissertation Structure

In this dissertation, we introduce algorithms for novelty detection and large scale similarity retrieval problem, as well as outline future work for missing annotation aware recognition systems. Chapter 2 proposses a new autoencoder architecture from our CVPR2020 paper [101]. Chapter 3 proposses a novelty detection method based on generative, probabilistic approach that is an updated version of our NeurIPS 2018 paper [71]. Chapter 4 proposses a deep similarity retrieval method via hashing and draws significantly from our work published at ACCV 2018 [102].

1.4.1 Chapter 2

We introduce a novel autoencoder architecture - ALAE, that is simple, flexible and affective. It bridges the gap between autoencoders and GANs in terms of the quality of generated images. It allows learning the probability distribution of the latent space while the data distribution is aligned with real data distribution in an adversarial way. We conduct extensive set of experiments that confirm that this enables learning representations that are likely less entangled. We also introduce StyleALAE which is an extension of ALAE to StyleGAN and this is the first autoencoder capable of generating and manipulating images while maintaining the same level of visual detail as StyleGAN.

1.4.2 Chapter 3

Recent approaches on novelty detection primarily leverage deep encoder-decoder network architectures to compute a reconstruction error that is used to either compute a novelty score or to train a one-class classifier. In contrust to the previous works, we take a probabilistic approach and effectively compute how likely it is that a sample was generated by the inlier distribution. We make the computation of the novelty probability feasible because we linearize the parameterized manifold capturing the underlying structure of the inlier distribution, and show how the probability factorizes and can be computed with respect to local coordinates of the manifold tangent space.

Section 3.2 introduces the *Generative Probabilistic Novelty Detection (GPND)* framework, and Section 3.3 describes the training and architecture of the adversarial autoencoder network. Section 3.5 shows a rich set of experiments showing that GPND is very effective and produces state-of-the-art results on several benchmarks.

We create a new dataset - aligned COVID-Net [103], by manually annotating samples from COVID-Net dataset and then aligning them. We demonstrate performance of our method on this new dataset and show that it performs better than state of the art methods.

1.4.3 Chapter 4

We propose *Spherical Deep Supervised Hashing (SDSH)*, a new supervised deep hashing approach to learn compact binary codes. In contrust to the previous works, we not only impose learning similarity preserving codes, but also encouraging them to be balanced. We propose a different relaxation method, that instead of using binarization priors, learns a spherical embedding, which overcomes the challenge of maintaining the learning problem well-posed. Our second contribution is formulation of a general triplet loss framework, with the introduction of the spring loss for learning balanced codes. Our third contribution is quantization of the spherical embedding that maximizes the mean average precision.

Section 4.3 introduces formulation of a general triplet loss framework, and Section 4.4 describes the spherical embedding. Section 4.5 introduces quantization algorithm, and Section 4.6 describes triplet spherical loss. Section 4.7 shows a rich set of experiments showing that SDSH is very effective and produces state-of-the-art results on several benchmarks.

Chapter 2

Adversarial Latent Autoencoders

2.1 Introduction

Generative Adversarial Networks (GAN) [1] have emerged as one of the dominant unsupervised approaches for computer vision and beyond. Their strength relates to their remarkable ability to represent complex probability distributions, like the face manifold [104], or the bedroom images manifold [105], which they do by learning a *generator* map from a known distribution onto the data space. Just as important are the approaches that aim at learning an *encoder* map from the data to a latent space. They allow learning suitable representations of the data for the task at hand, either in a supervised [106, 107, 108, 109, 110], or unsupervised [111, 112, 14, 29, 31, 21] manner.

Autoencoder (AE) [12, 13] networks are unsupervised approaches aiming at combining the "generative" as well as the "representational" properties by learning simultaneously an *encoder-generator* map. General issues subject of investigation in AE structures are whether they can: (a) have the same generative power as GANs; and, (b) learn disentangled representations [28]. Several works have addressed (a) [10, 33, 34, 7, 38]. An important testbed for success has been the ability for an AE to generate face images as rich and sharp as those produced by a GAN [22]. Progress has been made but victory has not been declared.

A sizable amount of work has addressed also (b) [14, 29, 30], but not jointly with (a).

We introduce an AE architecture that is general, and has generative power comparable to GANs while learning a less entangled representation. We observed that every AE approach makes the same assumption: *the latent space should have a probability distribution that is fixed* a priori *and the autoencoder should match it.* On the other hand, it has been shown in [2], the state-of-the-art for synthetic image generation with GANs, that an intermediate latent space, far enough from the imposed input space, tends to have improved disentanglement properties.

The observation above has inspired the proposed approach. We designed an AE architecture where we allow the latent distribution to be learned from data to address entanglement (A). The output data distribution is learned with an adversarial strategy (B). Thus, we retain the generative properties of GANs, as well as the ability to build on the recent advances in this area. For instance, we can seamlessly include independent sources of stochasticity, which have proven essential for generating image details, or can leverage recent improvements on GAN loss functions, regularization, and hyperparameters tuning [24, 113, 27, 114, 26, 21]. Finally, to implement (A) and (B) we impose the AE reciprocity in the latent space (C). Therefore, we can avoid using reconstruction losses based on simple ℓ_2 norm that operate in data space, where they are often suboptimal, like for the image space. We regard the unique combination of (A), (B), and (C) as the major techical novelty and strength of the approach. Since it works on the latent space, rather than autoencoding the data space, we named it *Adversarial Latent Autoencoder (ALAE)*.

We designed two ALAEs, one with a multilayer perceptron (MLP) as encoder with a symmetric generator, and another with the generator derived from a StyleGAN [2], which we call *StyleALAE*. For this one, we designed a companion encoder and a progressively growing architecture. We verified qualitatively and quantitatively that both architectures learn a latent space that is more disentangled than the imposed one. In addition, we show qualitative and quantitative results about face and bedroom image generation that are comparable with

StyleGAN at the highest resolution of 1024×1024. Since StyleALAE learns also an encoder network, we are able to show at the highest resolution, face reconstructions as well as several image manipulations based on real images rather then generated.

2.2 Preliminaries

A Generative Adversarial Network (GAN) [1] is composed of a generator network $\mathtt{G}$ mapping from a space $\mathcal{Z}$ onto a *data space* $\mathcal{X}$, and a discriminator network $\mathtt{D}$ mapping from $\mathcal{X}$ onto $\mathbb{R}$. The $\mathcal{Z}$ space is characterized by a known distribution $p(z)$. By sampling from $p(z)$, the generator $\mathtt{G}$ produces data representing a *synthetic* distribution $q(x)$. Given training data $\mathcal{D}$ drawn from a *real* distribution $p_{\mathcal{D}}(x)$, a GAN network aims at learning $\mathtt{G}$ so that $q(x)$ is as close to $p_{\mathcal{D}}(x)$ as possible. This is achieved by setting up a zero-sum two-player game with the discriminator $\mathtt{D}$. The role of $\mathtt{D}$ is to distinguish in the most accurate way data coming from the real versus the synthetic distribution, while $\mathtt{G}$ tries to fool $\mathtt{D}$ by generating synthetic data that looks more and more like real.

Following the more general formulation introduced in [25], the GAN learning problem entails finding the minimax with respect to the pair $(\mathtt{G}, \mathtt{D})$ (i.e., the Nash equilibrium), of the value function defined as

$$V(\mathtt{G}, \mathtt{D}) = E_{p_{\mathcal{D}}(x)}[f(\mathtt{D}(x))] + E_{p(z)}[f(-\mathtt{D}(\mathtt{G}(z)))] \; , \tag{2.1}$$

where $E[\cdot]$ denotes expectation, and $f : \mathbb{R} \to \mathbb{R}$ is a concave function. By setting $f(t) = -\log(1 + \exp(-t))$ we obtain the original GAN formulation [1]; instead, if $f(t) = t$ we obtain the Wasserstein GAN [24].

2.3 Adversarial Latent Autoencoders

We introduce a novel autoencoder architecture by modifying the original GAN paradigm. We begin by decomposing the generator $\mathtt{G}$ and the discriminator $\mathtt{D}$ in two networks: F, G, and E, D, respectively. This means that

$$\mathtt{G} = G \circ F \;, \quad \text{and} \quad \mathtt{D} = D \circ E \;, \tag{2.2}$$

see Figure 2.1. In addition, we assume that the *latent* spaces at the interface between F and G, and between E and D are the same, and we indicate them as $\mathcal{W}$. In the most general case we assume that F is a deterministic map, whereas we allow E and G to be stochastic. In particular, we assume that G might optionally depend on an independent noisy input η, with a known fixed distribution $p_\eta(\eta)$. We indicate with $G(w, \eta)$ this more general stochastic generator.

Under the above conditions we now consider the distributions at the output of every network. The network F simply maps $p(z)$ onto $q_F(w)$. At the output of G the distribution can be written as

$$q(x) = \int_w \int_\eta q_G(x|w, \eta) q_F(w) p_\eta(\eta) \, \mathrm{d}\eta \, \mathrm{d}w \;, \tag{2.3}$$

where $q_G(x|w, \eta)$ is the conditional distribution representing G. Similarly, for the output of E the distribution becomes

$$q_E(w) = \int_x q_E(w|x) q(x) \mathrm{d}x \;, \tag{2.4}$$

where $q_E(w|x)$ is the conditional distribution representing E. In (2.4) if we replace $q(x)$ with $p_{\mathcal{D}}(x)$ we obtain the distribution $q_{E,\mathcal{D}}(w)$, which describes the output of E when the real data distribution is its input.

Since optimizing (2.1) leads toward the synthetic distribution matching the real one, i.e., $q(x) = p_{\mathcal{D}}(x)$, it is obvious from (2.4) that doing so also leads toward having $q_E(w) =$

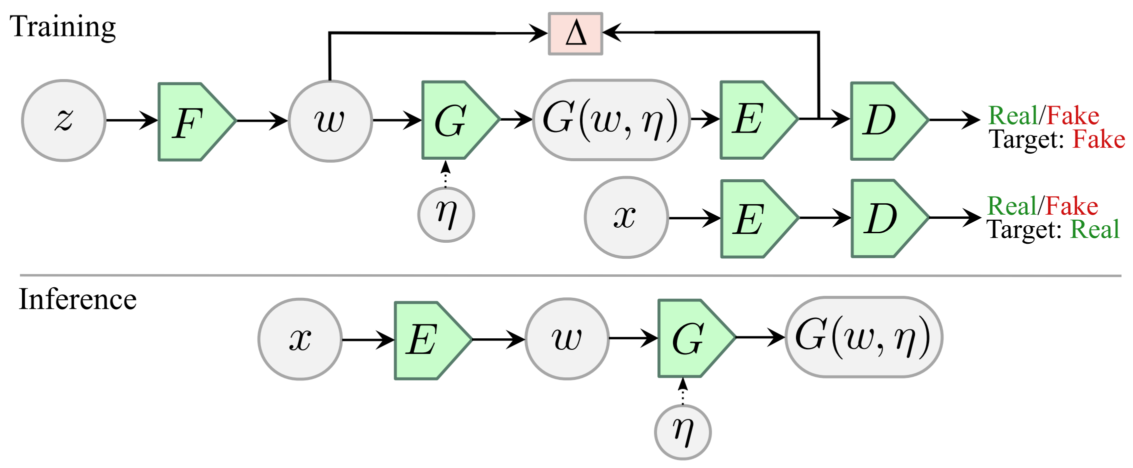


Figure 2.1: **ALAE Architecture.** Architecture of an Adversarial Latent Autoencoder.

$q_{E,\mathcal{D}}(w)$.

In addition to that, to achieve reciprocity, we want learn such $q_E\left(w|x\right)$ and $q_G\left(x|w\right)$ that maximize marginal likelihood $q_E(w)$. We do so by maximizing the expectation of $q_E\left(w|x\right)$ over the distribution of $q_G\left(x|w\right)$ in expectation over the distribution $q_F(w)$, where $q_G\left(x|w\right)$ is marginal distribution over the $p_\eta(\eta)$, i.e. $q_G\left(x|w\right) = \int_\eta q_G(x|w,\eta)p_\eta(\eta)$.

$$\underset{E,G}{\arg\max}\, E_{w\sim q_F}\left[E_{x\sim q_G(x|w)} \log q_E\left(w|x\right)\right] \tag{2.5}$$

Computing marginal distribution $q_E(w)$ is generally intractable, so we make several approximations discussed later in 2.3.1.

In this way we could interpret the pair of networks (G, E) as a *generator-encoder* network that autoencodes the *latent space* $\mathcal{W}$.

If we incorporate likelihood maximization into the learning process by regularizing the GAN loss (2.1) with the goal (2.5) via alternating the following two optimizations

$$\min\nolimits_{F,G} \max\nolimits_{E,D} V(G\circ F, D\circ E) \tag{2.6}$$

$$\underset{E,G}{\arg\max}\, E_{w\sim q_F}\left[E_{x\sim q_G(x|w)} \log q_E\left(w|x\right)\right] \tag{2.7}$$

Table 2.1: Autoencoder criteria used: (a) for matching the real to the synthetic data distribution; (b) for setting/learning the latent distribution; (c) for which space reciprocity is achieved.

Autoencoder	(a) Data Distribution	(b) Latent Distribution	(c) Reciprocity Space
VAE [12, 13]	similarity	imposed/divergence	data
AAE [10]	similarity	imposed/adversarial	data
VAE/GAN [33]	similarity	imposed/divergence	data
VampPrior [32]	similarity	learned/divergence	data
BiGAN [34]	adversarial	imposed/adversarial	adversarial
ALI [7]	adversarial	imposed/adversarial	adversarial
VEEGAN [35]	adversarial	imposed/divergence	latent
AGE [36]	adversarial	imposed/adversarial	latent
IntroVAE [38]	adversarial	imposed/adversarial	data
ALAE (ours)	adversarial	learned/divergence	latent

We refer to a network optimized according to (2.6) (2.7) as an *Adversarial Latent Autoencoder (ALAE)*. The building blocks of an ALAE architecture are depicted in Figure 2.1.

2.3.1 Relation with other autoencoders

Data distribution. In architectures composed by an *encoder* network and a *generator* network, the task of the encoder is to map input data onto a space characterized by a *latent distribution*, whereas the generator is tasked to map latent codes onto a space described by a *data distribution*. Different strategies are used to learn the data distribution. For instance, some approaches impose a similarity criterion on the output of the generator [12, 13, 10, 32], or even learn a similarity metric [33]. Other techniques instead, set up an adversarial game to ensure the generator output matches the training data distribution [34, 7, 35, 36, 38]. This latter approach is what we use for ALAE.

Latent distribution. For the latent space instead, the common practice is to set a desired target latent distribution, and then the encoder is trained to match it either by minimizing a divergence type of similarity [12, 13, 33, 35, 32], or by setting up an adversarial game [10, 34, 7, 36, 38]. Here is where ALAE takes a fundamentally different approach. Indeed, we do not impose the latent distribution, i.e., $q_E(w)$, to match a target distribution.

The only condition we set, is given by (2.5). In other words, we do not want F to be the identity map, and are very much interested in letting the learning process decide what F should be.

Reciprocity. Another aspect of autoecoders is whether and how they achieve reciprocity. This property relates to the ability of the architecture to reconstruct a data sample x from its code w, and vice-versa. Clearly, this requires that $x = G(E(x))$, or equivalently that $w = E(G(w))$. In the first case, the network must contain a reconstruction term that operates in the data space. In the latter one, the term operates in the latent space. While most approaches follow the first strategy [12, 13, 10, 33, 38, 32], there are some that implement the second [35, 36], including ALAE.

Indeed, we can make (2.7) tracktable by assuming distribution $q_E(w|x)$ to be factorized Gaussian. We can compute integral over $p_\eta(\eta)$ using Monte Carlo integration with one sample, like it is done in [12] leading to minimization of the expected coding reconstruction error, as follows

$$\arg\min_{E,G} E_{p(z)}\left[\|F(z) - E \circ G \circ F(z)\|_2^2\right] \tag{2.8}$$

Imposing reciprocity in the latent space gives the significant advantage that simple ℓ_2, ℓ_1 or other norms can be used effectively, regardless of whether they would be inappropriate for the data space. For instance, it is well known that element-wise ℓ_2 norm on image pixel differences does not reflect human visual perception. On the other hand, when used in latent space its meaning is different. For instance, an image translation by a pixel could lead to a large ℓ_2 discrepancy in image space, while in latent space its representation would hardly change at all. Ultimately, using ℓ_2 in image space has been regarded as one of the reasons why autoencoders have not been as successful as GANs in reconstructing/generating sharp images [33]. Another way to address the same issue is by imposing reciprocity adversarially, as it was shown in [34, 7]. Table 2.1 reports a summary of the main characteristics of most of the recent generator-encoder architectures.

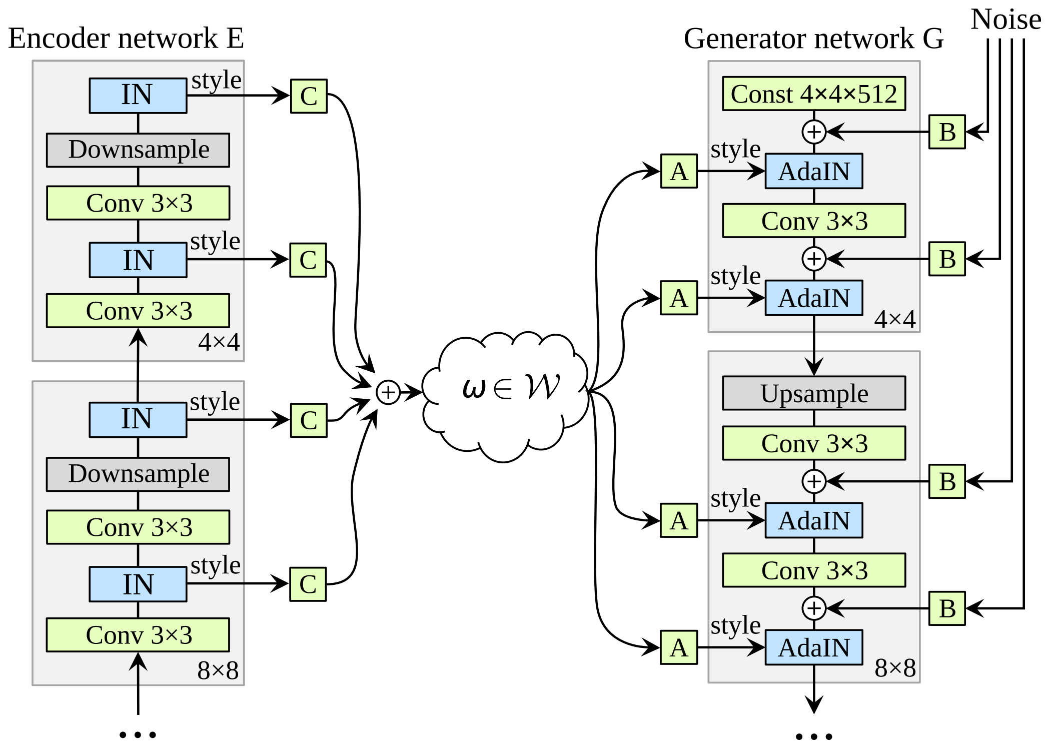


Figure 2.2: **StyleALAE Architecture.** The StyleALAE encoder has Instance Normalization (IN) layers to extract multiscale style information that is combined into a latent code w via a learnable multilinear map.

2.4 StyleALAE

We use ALAE to build an autoencoder that uses a StyleGAN based generator. For this we make our latent space $\mathcal{W}$ play the same role as the intermediate latent space in [2]. Therefore, our G network becomes the part of StyleGAN depicted on the right side of Figure 2.2. The left side is a novel architecture that we designed to be the encoder E.

Since at every layer, G is driven by a style input, we design E symmetrically, so that from a corresponding layer we extract style information. We do so by inserting Instance Normalization (IN) layers [115], which provide instance averages and standard deviations for every channel. Specifically, if y_i^E is the output of the i-th layer of E, the IN module extracts the statistics $\mu(y_i^E)$ and $\sigma(y_i^E)$ representing the style at that level. The IN module also provides as output the normalized version of the input, which continues down the pipeline with no more style information from that level. Given the information flow between E and G, the architecture is effectively mimicking a multiscale style transfer from E to G, with the difference that there is not an extra input image that provides the content [115, 116].

The set of styles that are inputs to the Adaptive Instance Normalization (AdaIN) layers [115] in G are related linearly to the latent variable w. Therefore, we propose to combine the styles output by the encoder, and to map them onto the latent space, via the following multilinear map

$$w = \sum_{i=1}^{N} C_i \begin{bmatrix} \mu(y_i^E) \\ \sigma(y_i^E) \end{bmatrix} \tag{2.9}$$

where the C_i's are learnable parameters, and N is the number of layers.

Similarly to [22, 2] we use progressive growing. We start from low-resolution images (4×4 pixels) and progressively increase the resolution by smoothly blending in new blocks to E and G. For the F and D networks we implement them using MLPs. The $\mathcal{Z}$ and $\mathcal{W}$ spaces, and all layers of F and D have the same dimensionality in all our experiments. Moreover, for StyleALAE we follow [2], and chose F to have 8 layers, and we set D to have 3 layers.

Algorithm 1 ALAE Training

```
1: $\theta_F, \theta_G, \theta_E, \theta_D \leftarrow$ Initialize network parameters
2: **while** not converged **do**
3:     Step I. Update $E$, and $D$
4:     $x \leftarrow$ Random mini-batch from dataset
5:     $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
6:     $L_{adv}^{E,D} \leftarrow \text{softplus}(D \circ E \circ G \circ F(z))) + \text{softplus}(-D \circ E(x)) + \frac{\gamma}{2} \mathrm{E}_{p_{\mathcal{D}}(x)} \left[ \|\nabla D \circ E(x)\|^2 \right]$
7:     $\theta_E, \theta_D \leftarrow \textsc{Adam}(\nabla_{\theta_D, \theta_E} L_{adv}^{E,D}, \theta_D, \theta_E, \alpha, \beta_1, \beta_2)$
8:     Step II. Update $F$, and $G$
9:     $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
10:    $L_{adv}^{F,G} \leftarrow \text{softplus}(-D \circ E \circ G \circ F(z)))$
11:    $\theta_F, \theta_G \leftarrow \textsc{Adam}(\nabla_{\theta_F, \theta_G} L_{adv}^{F,G}, \theta_F, \theta_G, \alpha, \beta_1, \beta_2)$
12:    Step III. Update $E$, and $G$
13:    $z \leftarrow$ Samples from prior $\mathcal{N}(0, I)$
14:    $L_{error}^{E,G} \leftarrow \|F(z) - E \circ G \circ F(z)\|_2^2$
15:    $\theta_E, \theta_G \leftarrow \textsc{Adam}(\nabla_{\theta_E, \theta_G} L_{error}^{E,G}, \theta_E, \theta_G, \alpha, \beta_1, \beta_2)$
16: **end while**
```

2.5 Implementation

Adversarial losses and regularization. We use a non-saturating loss [1, 26], which in (2.1) we introduce by setting $f(\cdot)$ to be a SoftPlus function [117]. This is a smooth version of the rectifier activation function, defined as $f(t) = \mathrm{softplus}(t) = \log(1 + \exp(t))$. In addition, we use gradient regularization techniques [118, 26, 119]. We utilize R_1 [23, 26], a zero-centered gradient penalty term which acts only on real data, and is defined as $\frac{\gamma}{2}\,\mathrm{E}_{p_{\mathcal{D}}(x)}\left[\|\nabla D \circ E(x)\|^2\right]$, where the gradient is taken with respect to the parameters θ_E and θ_D of the networks E and D, respectively.

This regularization prevents the discriminator from creating non-zero gradients orthogonal to the data manifold, which would cause a deviation from the Nash-equilibrium when the generator produces a distribution close to the real data distribution. With the non-saturating loss and R_1 regularization we observe better convergence and stability compared to the standard GAN loss [1]. This is in line with what was reported in [2].

Training. In order to optimizate (2.6) (2.7) we use alternating updates. One iteration is composed of three updating steps: two for (2.6) and one for (2.7). Step I updates the discriminator (i.e., networks E and D). Step II updates the generator (i.e., networks F and G). Step III updates the latent space autoencoder (i.e., networks G and E). The procedural details are summarized in Algorithm 1. For updating the weights we use the Adam optimizer [120] with $\beta_1 = 0.0$ and $\beta_2 = 0.99$, coupled with the learning rate equalization technique [22] described below. For non-growing architectures (i.e., MLPs) we use a learning rate of 0.002, and batch size of 128. For growing architectures (i.e., StyleALAE) learning rate and batch size depend on the resolution.

Learning rate equalization. We use learning rate equalization as opposed to batch normalization. The idea is to compensate for different dynamic ranges of the network parameters. However, we implement it differently than in [22]. There the weights are initialized with $\mathcal{N}(0, 1)$ and scaled dynamically, which means that the scaling operation is part of the computational graph and participates in the gradient computation. The weights are scaled according to the He's initializer [121], setting $\hat{w}_i = w_i\sqrt{2/n_i}$, where w_i are the weights of the i-th layer, and n_i is the number of connections of a response in that layer (i.e., the fan-in). The benefit here is that the dynamic range of all weights is the same, while providing a good initialization. The downside is that a scaled copy of all weights should be computed each forward pass and all gradients should be scaled back each backward pass. In order to reduce the amount of computation, we take a slightly different route and initialize the weights with $\mathcal{N}(0, \sqrt{2/n_i})$, while we correct the learning rate for that scale factor as $\hat{\eta}_i = \eta\sqrt{2/n_i}$, where η here is a global learning rate, and $\hat{\eta}_i$ is the learning rate corrected for the i-th layer. In this way the same behaviour is achieved without the need to explicitly store weights with a normalized dynamic range.

2.6 Experiments

2.6.1 Training details

We implemented our approach with PyTorch [122], code and uncompressed images are available at https://github.com/wvuvl/ALAE. As a byproduct of our implementation, we have also reimplemented StyleGAN [2] and PGGAN [22]. To verify that our implementation matches the original, we have implemented methods for loading the original StyleGAN weights, and we were able to reproduce the same FID score reported in [2].

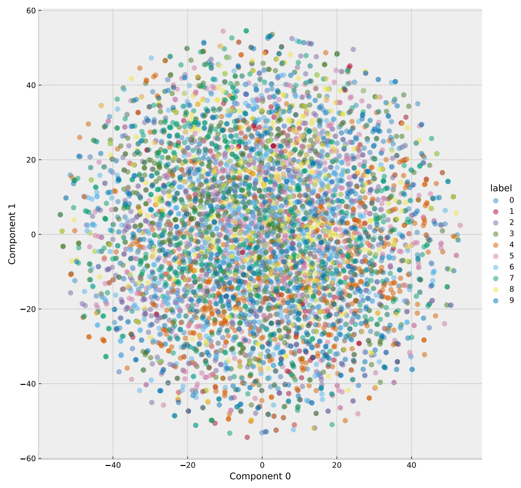
Most of the experiments were conducted on a machine with 8$\times$ GPU Titan RTX. We trained StyleALAE on 60000 training samples from the FFHQ [2] dataset for 147 epoch, from which 18 epochs were spent at resolution 1024×1024. Starting from resolution 4×4 we have grown the StyleALAE up to 1024×1024. When growing to a new resolution level we used 500k training samples during the transition, and another 500k samples for training stabilization. Once reached the maximum resolution of 1024×1024, we continued training for 1M images. Thus, the total training time measured in images was 10M.

We contrast our training time with the one reported for StyleGAN [2]. At every resolution level they used 600k images during the transition, and 600k images for stabilization. Their total training time was 25M images, and 15M of them were used for training at resolution 1024×1024. Note that at the same resolution we trained StyleALAE with only 1M images, so, 15 times less. Unfortunately, our reduced budget did not allow us to train for a longer time, and we reported all the scores, including the FID score, at 1M training time, while observing that it was still improving. We regard the large training time difference between StyleALAE and StyleGAN (1M vs 15M) as likely to be the cause of discrepancy between the corresponding FID scores.

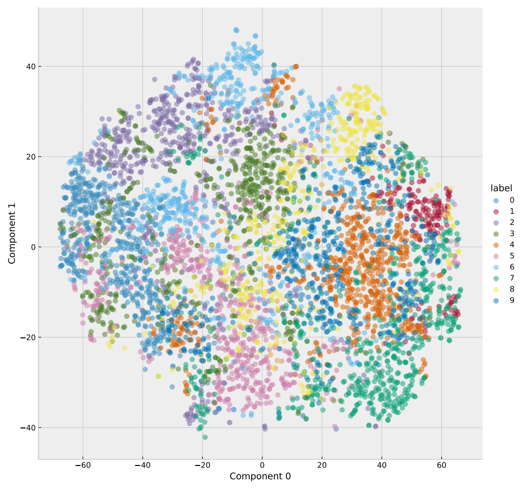
On the FID score. We would like to point out a major limitation of the FID score as a measurement for how good are the generations of a GAN. As mentioned above, we verified that the FID score of StyleGAN is 4.4, when computed as described in [2]. In addition to that, we computed the FID score between 50000 training samples from FFHQ, and the 10000 images from FFHQ that we use for testing. In this case the FID score is 4.5. This result suggests that the generations are "closer" to the training distribution than the real images, which is not true. On the other hand, this result highlights how limited the FID score is as a metric for generation, and that more research should be done in the future to come up with more meaningful metrics in this domain.

2.7 Latent space projections

After training ALAE on the permutation-invariant MNIST dataset [124, 125], we projected the $\mathcal{Z}$ space and the $\mathcal{W}$ space representations of the training dataset based on t-SNE [123]. The result is depicted in Figure 2.3, where the digit labels are color coded. Since the distribution of the $\mathcal{Z}$ space is a multivariate Gaussian, as expected t-SNE does not unwrap any manifold structure. However, it can be observed how randomly distributed are the labels. On the other hand, the t-SNE projection of the $\mathcal{W}$ space shows that labels are much more aggregated in clusters. This seems to suggest that a traversal in the $\mathcal{W}$ space should



(a) Projection of $\mathcal{Z}$ space



(b) Projection of $\mathcal{W}$ space

Figure 2.3: **Projections of $\mathcal{Z}$ and $\mathcal{W}$ spaces**. Qualitatively shows that labels in $\mathcal{Z}$ space are arbitrary entangled, while $\mathcal{W}$ is more disentangled. Samples in $\mathcal{W}$ were project to 2D space using t-SNE [123]

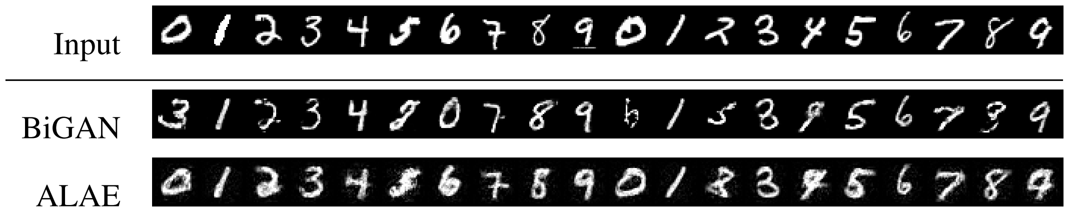


Figure 2.4: **MNIST reconstruction.** Reconstructions of the permutation-invariant MNIST. Top row: real images. Middle row: BiGAN reconstructions. Bottom row: ALAE reconstructions. The same MLP architecture is used in both methods.

lead to a smoother transition in image space, as opposed to a transition performed in the $\mathcal{Z}$ space. This indeed is observed also in Figure 2.5, where we show two type of traversals: one obtained by interpolating in the $\mathcal{Z}$ space, and the other obtained by interpolating the representation of the same samples, but in the $\mathcal{W}$ space.

2.7.1 Representation learning with MLP

We train ALAE with MNIST [124], and then use the feature representation for classification, reconstruction, and analyzing disentanglement. We use the permutation-invariant setting, where each 28×28 MNIST image is treated as a 784D vector without spatial structure, which requires to use a MLP instead of a CNN. We follow [125] and use a three layer MLP

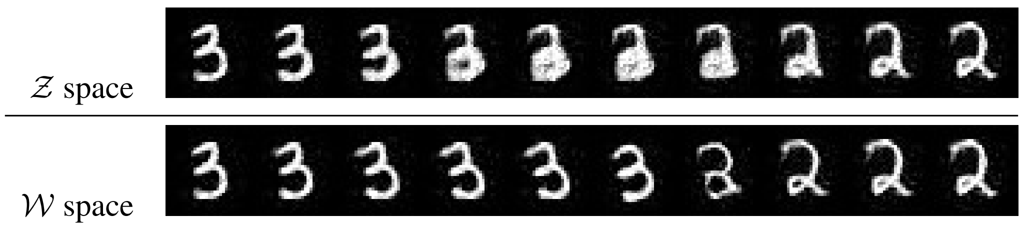


Figure 2.5: **MNIST traversal.** Reconstructions of the interpolations in the $\mathcal{Z}$ space, and the $\mathcal{W}$ space, between the same digits. The latter transition appears to be smoother.

Table 2.2: **MNIST classification.** Classification accuracy (%) on the permutation-invariant MNIST [124] using 1NN and linear SVM, with *same writers* (SW) and *different writers* (DW) settings, and short features (sf) vs. long features (lf), indicated as sf/lf.

	1NN SW	**Linear SVM SW**	**1NN DW**	**Linear SVM DW**
AE(ℓ_1)	97.15/97.43	88.71/97.27	96.84/96.80	89.78/97.72
AE(ℓ_2)	**97.52**/97.37	88.78/97.23	**97.05**/96.77	89.78/97.72
LR	92.79/97.28	89.74/97.56	91.90/96.69	90.03/97.80
JLR	92.54/97.02	89.23/97.19	91.97/96.45	90.82/97.62
BiGAN [125]	95.83/97.14	90.52/97.59	95.38/96.81	91.34/97.74
ALAE (ours)	93.79/**97.61**	**93.47/98.20**	94.59/**97.47**	**94.23/98.64**

with a latent space size of 50D. Both networks, E and G have two hidden layers with 1024 units each. In [125] the features used are the activations of the layer before the last of the encoder, which are 1024D vectors. We refer to those as *long features*. We also use, as features, the 50D vectors taken from the latent space, $\mathcal{W}$. We refer to those as *short features*.

MNIST has an official split into training and testing sets of sizes 60000 and 10000 respectively. We refer to it as *different writers* (DW) setting since the human writers of the digits for the training set are different from those who wrote the testing digits. We consider also a *same writers* (SW) setting, which uses only the official training split by further splitting it in two parts: a train split of size 50000 and a test split of size 10000, while the official testing split is ignored. In SW the pools of writers in the train and test splits overlap, whereas in DW they do not. This makes SW an easier setting than DW.

Results. We report the accuracy with the 1NN classifier as in [125], and extend those results by reporting also the accuracy with the linear SVM, because it allows a more direct analysis of disentanglement. Indeed, we recall that a disentangled representation [126, 127, 28] refers to a space consisting of linear subspaces, each of which is responsible for one factor of variation. Therefore, a linear classifier based on a disentangled feature space should lead to better performance compared to one working on an entangled space. Table 2.2 summarizes the average accuracy over five trials for ALAE, BiGAN, as well as the following baselines proposed in [125]: Latent Regressor (LR), Joint Latent Regressor (JLR), Autoencoders trained to minimize the ℓ_2 (AE(ℓ_2)) or the ℓ_1 (AE(ℓ_1)) reconstruction error.

The most significant result of Table 2.2 is drawn by comparing the 1NN with the corresponding linear SVM columns. Since 1NN does not presume disentanglement in order to be effective, but linear SVM does, larger performance drops signal stronger entanglement. ALAE is the approach that remains more stable when switching from 1NN to linear SVM, suggesting a greater disentanglement of the space. This is true especially for short features, whereas for long features this effect fades away because linear separability grows.

We also note that ALAE does not always provide the best accuracy, and the baseline AE (especially AE(ℓ_2)) does well with 1NN, and more so with short features. This might be explained by the baseline AE learning a representation that is closer to a discriminative one. Other approaches instead focus more on learning representations for drawing synthetic random samples, which are likely richer, but less discriminative. This effect also fades for longer features.

Another observation is about SW vs. DW. 1NN generalizes less effectively for DW, as expected, but linear SVM provides a small improvement. This is unclear, but we speculate that DW might have fewer writers in the test set, and potentially slightly less challenging.

Figure 2.4 shows qualitative reconstruction results. It can be seen that BiGAN reconstructions are subject to semantic label flipping much more often than ALAE. Finally, Figure 2.5 shows two traversals: one obtained by interpolating in the $\mathcal{Z}$ space, and the other by interpolating in the $\mathcal{W}$ space. The second shows a smoother image space transition, suggesting a lesser degree of entanglement.

Metric	**ALAE**	**ALAE no style**	**ALAE LR**	**ALAE $\mathcal{Z}$**
FID [128]	15.15	19.28	7.55	17.78
LPIPS [8]	0.3266	0.3226	0.3923	0.7700

Table 2.3: **Ablation study.** FID scores (lower is better) measured on CelebA [129] at resolution 128×128.

2.8 Ablation

We perform an ablation study using 128×128 crops from the CelebA [104] dataset. We analyze the behavior of StyleALAE for the cases when:

- The style-based encoder is replaced by a regular encoder, with an architecture similar to the discriminator network of StyleGAN (denoted as **ALAE no-style**).
- The encoder E is not part of the discriminator pipeline and is not included in the minimax game. This architecture is similar to the Latent Regressor defined in [34]; however, in our case it still uses he $\mathcal{W}$ space (denoted as **ALAE LR**).
- The latent space distribution is imposed. This means that the $\mathcal{Z}$ space is the latent space, not $\mathcal{W}$. Consequently, also the reciprocity is imposed in the $\mathcal{Z}$ space (denoted as **ALAE** $\mathcal{Z}$).

Table 2.5 shows the FID [128] and LPIPS [8] scores for the four configurations of the ALAE architecture. As it can be seen, the style-based encoder helps with the FID score, compared with ALAE no-style, indicating that it better allows to capture the training distribution. Similarly, the FID score also increases when the latent space distribution is imposed, as shown in ALAE $\mathcal{Z}$. On the other hand, the FID score decreases for ALAE LR. This can be explained because that architecture is more directly focussed on learning the distribution of the training set, just like a regular GAN architecture is. In ALAE instead, the encoder participates in the minimax game, making it harder for the generator to learn the training distribution. However, ALAE LR pays a price: the encoder is fully trained with generated images only, and never sees real images. Therefore, it never learns how to encode

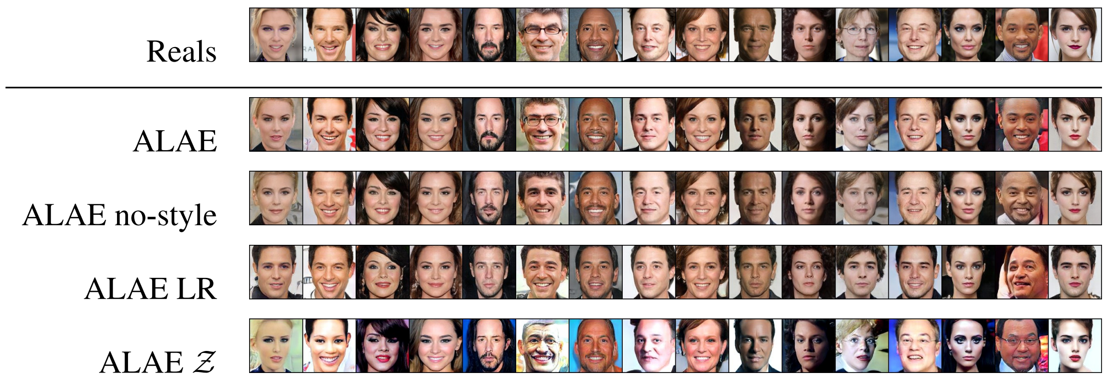


Figure 2.6: **CelebA reconstructions.** Qualitative results for CelebA reconstructions. Top row: real images. Second row: ALAE reconstructions. Third row: ALAE no-style reconstructions. Fourth row: ALAE LR reconstructions. Last row: ALAE $\mathcal{Z}$ reconstructions. In can be noted how ALAE LR produces reconstructions that look fairly good, but resemble less the original subject. For ALAE $\mathcal{Z}$ instead, besides diminished resemblance with the original subject, the reconstructions definitely look more degraded.

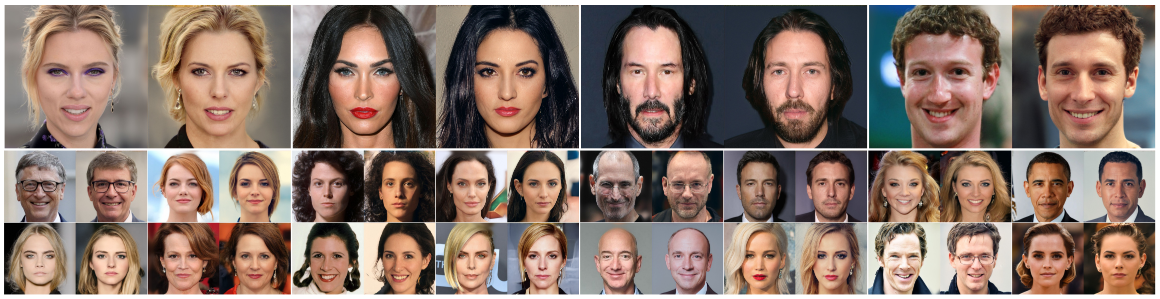


Figure 2.7: **FFHQ reconstructions.** Reconstructions of unseen images with StyleALAE trained on FFHQ [2] at 1024×1024.

real images well. This is reflected in the higher LPIPS score, and in the reconstructions in Figure 2.10. About ALAE $\mathcal{Z}$ instead, again, imposing the latent distribution does not help even with the LPIPS score, as it is also visible in the reconstructions in Figure 2.10.

In ALAE no-le we show how in the case of MNIST, even if we do not use the learning equalization we obtain comparable results. On the other hand, in our experiments with larger networks like for StyleALAE, we have observed that adding learning equalization generally provides for greater performance increases. For ALAE $\mathcal{Z}$ instead, we witness a clear deterioration of the performance. In particular, with short features the linear SVM performs much worse, highlighting a much more entangled space.

Table 2.4: **FID scores.** FID scores (lower is better) measured on FFHQ [2] and LSUN Bedroom [130].

Method	**FFHQ**	**LSUN Bedroom**
StyleGAN [2]	4.40	2.65
PGGAN [22]	-	8.34
IntroVAE [38]	-	8.84
Pioneer [131]	-	18.39
Balanced Pioneer [132]	-	17.89
StyleALAE Generation	13.09	17.13
StyleALAE Reconstruction	16.52	15.92

2.8.1 Learning style representations

FFHQ. We evaluate StyleALAE with the FFHQ [2] dataset. It is very recent and consists of 70000 images of people faces aligned and cropped at resolution of 1024×1024. In contrast to [2], we split FFHQ into a training set of 60000 images and a testing set of 10000 images. We do so in order to measure the reconstruction quality for which we need images that were not used during training.

We implemented our approach with PyTorch. Most of the experiments were conducted on a machine with $4\times$ GPU Titan X, but for training the models at resolution 1024×1024 we used a server with $8\times$ GPU Titan RTX. We trained StyleALAE for 147 epochs, 18 of which were spent at resolution 1024×1024. Starting from resolution 4×4 we grew StyleALAE up to 1024×1024. When growing to a new resolution level we used 500k training samples during the transition, and another 500k samples for training stabilization. Once reached the maximum resolution of 1024×1024, we continued training for 1M images. Thus, the total training time measured in images was 10M. In contrast, the total training time for StyleGAN [2] was 25M images, and 15M of them were used at resolution 1024×1024. At the same resolution we trained StyleALAE with only 1M images, so, 15 times less. It takes slightly over two days to reach resolution 512×512 and slightly over two days to continue training and reach resolution 1024×1024 on a machine with 8 X Titan RTX.

Table 2.4 reports the FID score [128] for generations and reconstructions. Source images

Table 2.5: **PPL.** Perceptual path lengths on FFHQ measured in the $\mathcal{Z}$ and the $\mathcal{W}$ spaces (lower is better).

Method		**Path length**	
		full	**end**
StyleGAN	$\mathcal{Z}$	412.0	415.3
StyleGAN no mixing	$\mathcal{W}$	200.5	160.6
StyleGAN	$\mathcal{W}$	231.5	182.1
StyleALAE	$\mathcal{Z}$	300.5	292.0
StyleALAE	$\mathcal{W}$	**134.5**	**103.4**

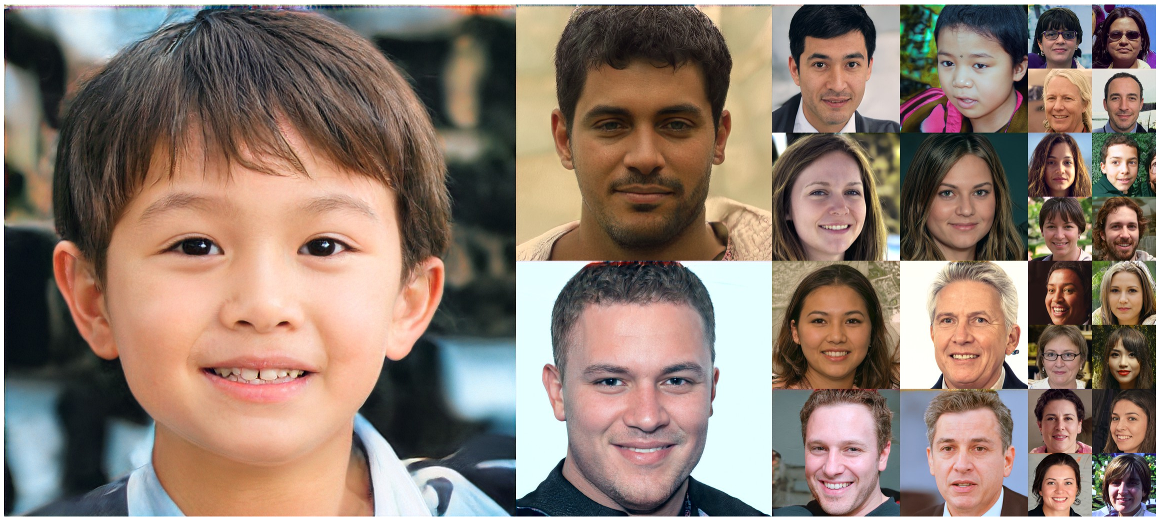


Figure 2.8: **FFHQ generations.** Generations with StyleALAE trained on FFHQ [2] at 1024×1024.

for reconstructions are from the test set and were not used during training. The scores of StyleALAE are higher, and we regard the large training time difference between StyleALAE and StyleGAN (1M vs 15M) as the likely cause of the discrepancy.

Table 2.5 reports the perceptual path length (PPL) [2] of SyleALAE. This is a mea-

Table 2.6: Comparison of FID and PPL scores for CelebA-HQ images at 256×256 (lower is better). FID is based on 50,000 generated samples compared to training samples.

	FID	**PPL full**
PGGAN [22]	**8.03**	229.2
GLOW [40]	68.93	219.6
PIONEER [131]	39.17	155.2
Balanced PIONEER [132]	25.25	146.2
StyleALAE (ours)	19.21	**33.29**

surement of the degree of disentanglement of representations. We compute the values for representations in the $\mathcal{W}$ and the $\mathcal{Z}$ space, where StyleALAE is trained with style mixing in both cases. The StyleGAN score measured in $\mathcal{Z}$ corresponds to a traditional network, and in $\mathcal{W}$ for a style-based one. We see that the PPL drops from $\mathcal{Z}$ to $\mathcal{W}$, indicating that $\mathcal{W}$ is perceptually more linear than $\mathcal{Z}$, thus less entangled. Also, note that for our models the PPL is lower, despite the higher FID scores.

Figure 2.8 shows a random collection of generations obtained from StyleALAE. Figure 2.7 instead shows a collection of reconstructions.

Figure 2.11 instead shows traversals based on principal directions along CelebA-HQ attributes. The principal directions are obtained in this way. Given an attribute (e.g., old/young), we train a binary linear SVM classifier that distinguishes old vs young people, based on the latent space representation. We then use the direction orthogonal to the SVM separating boundary as the principal direction of variation for that attribute. Then, given an image, we compute the latent representation, and add to it a variation along the principal direction of variation. We can then draw noise and use the generator network to generate face images.

In Figure 2.12 instead, we repeat the style mixing experiment in [2], but with real images as sources and destinations for style combinations. We note that the original images are faces of celebrities that we downloaded from the internet. Therefore, they are not part of FFHQ, and come from a different distribution. Indeed, FFHQ is made of face images obtained from Flickr.com depicting non-celebrity people. Often the faces do not wear any makeup, neither have the images been altered (e.g., with Photoshop). Moreover, the imaging conditions of the FFHQ acquisitions are very different from typical photoshoot stages, where professional equipment is used. Despite this change of image statistics, we observe that StyleALAE works effectively on both reconstruction and mixing.

LSUN. We evaluated StyleALAE with LSUN Bedroom [130]. Figure 2.9 shows generations and reconstructions from unseen images during training. Table 2.4 reports the FID

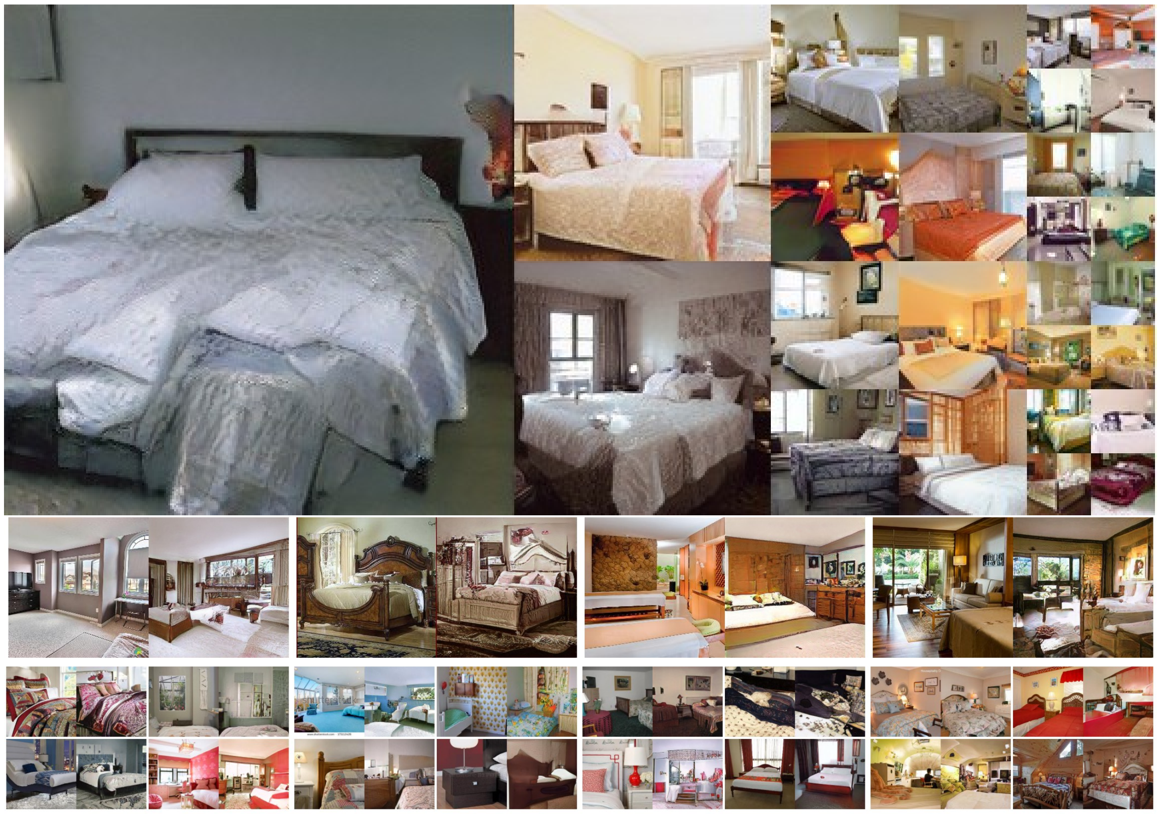


Figure 2.9: **LSUN generations and reconstructions.** Generations (first row), and reconstructions using StyleALAE trained on LSUN Bedroom [130] at resolution 256×256.

scores on the generations and the reconstructions.

CelebA-HQ. CelebA-HQ [22] is an improved subset of CelebA [104] consisting of 30000 images at resolution 1024×1024. We follow [131, 132, 40, 22] and use CelebA-HQ downscaled to 256×256 with training/testing split of 27000/3000. Table 2.6 reports the FID and PPL scores, and Figure 2.10 compares StyleALE reconstructions of unseen faces with two other approaches.

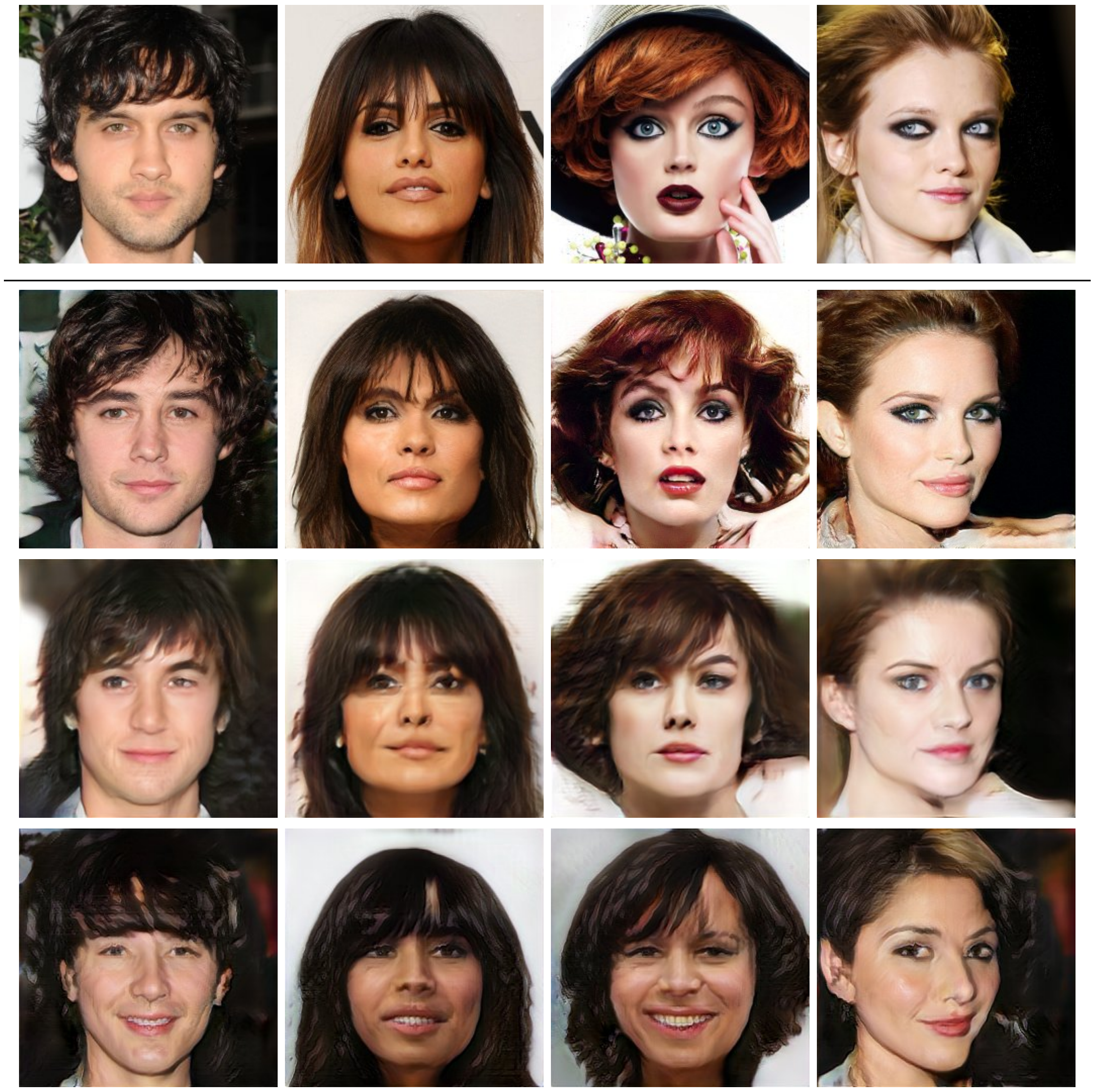


Figure 2.10: **CelebA-HQ reconstructions.** CelebA-HQ reconstructions of unseen samples at resolution 256×256. Top row: real images. Second row: StyleALAE. Third row: Balanced PIONEER [132]. Last row: PIONEER [131]. StyleALAE reconstructions look sharper and less distorted.

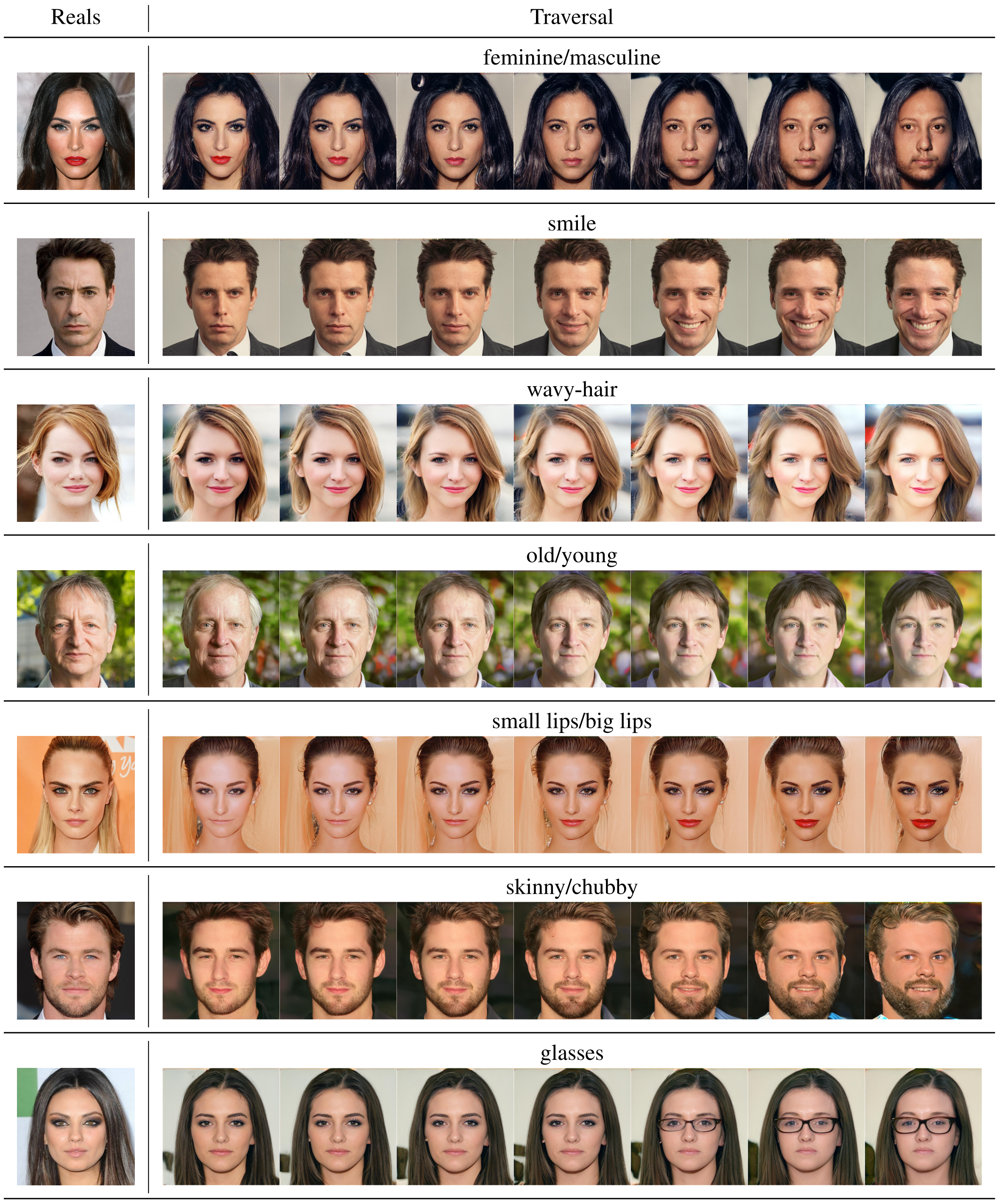


Figure 2.11: **Attribute traversals.** Qualitative results for reconstructions and attribute traversals on FFHQ dataset at resolution 1024×1024. Reconstructions from images that are not part of FFHQ. Left column: real images. Columns from the second to the last one: StyleALAE reconstructions of the source image with feature corresponding to an attribute being modified. The $\mathcal{W}$ representation of the input image was modified by adding/subtracting a vector that was identified as principal direction for the selected attribute. All manipulations on the space $\mathcal{W}$ are linear. All directions were determined as perpendiculars to the decision boundary of a Linear SVM fitted to detect a particular attribute on the space $\mathcal{W}$.

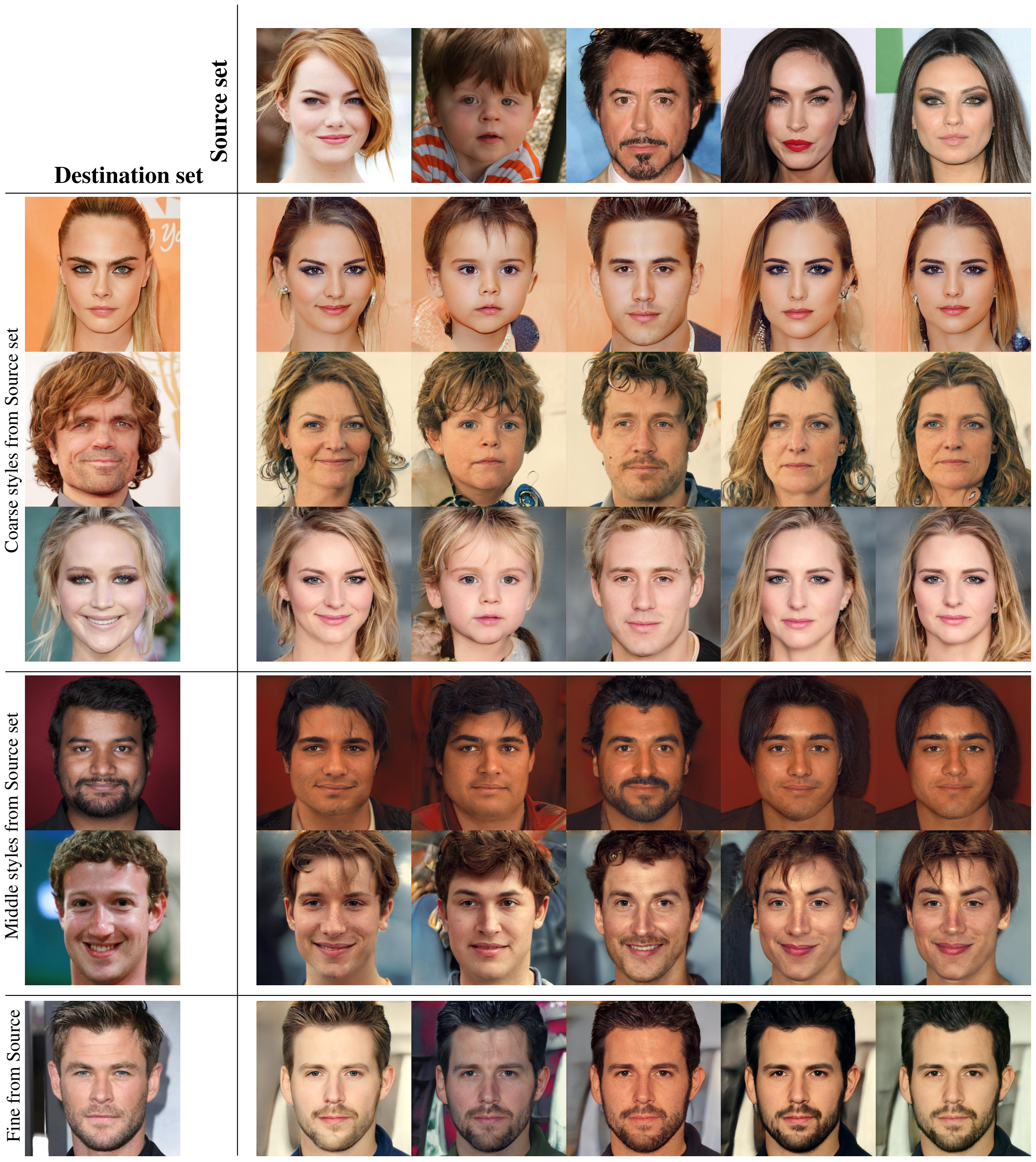


Figure 2.12: Two sets of real images were picked to form the Source set and the Destination set. The rest of the images were generated by copying specified subset of styles from the Source set into the Destination set. This experiment repeats the one from [2], but with real images. Copying the coarse styles brings high-level aspects such as pose, general hair style, and face shape from Source set, while all colors (eyes, hair, lighting) and finer facial features resemble the Destination set. Instead, if we copy middle styles from the Source set, we inherit smaller scale facial features like hair style, eyes open/closed from Source, while the pose, and general face shape from Destination are preserved. Finally, copying the fine styles from the Source set brings mainly the color scheme and microstructure.

Chapter 3

Generative Probabilistic Novelty Detection with Adversarial Autoencoders

3.1 Introduction

Novelty detection is the problem of identifying whether a new data point is considered to be an *inlier* or an *outlier*. From a statistical point of view this process usually occurs while prior knowledge of the distribution of inliers is the only information available. This is also the most difficult and relevant scenario because outliers are often very rare, or even dangerous to experience (e.g., in industry process fault detection [133]), and there is a need to rely only on inlier training data. Novelty detection has received significant attention in application areas such as medical diagnoses [134], drug discovery [135], and among others, several computer vision applications, such as anomaly detection in images [49, 136], videos [137], and outlier detection [56, 138]. We refer to [139] for a general review on novelty detection. The most recent approaches are based on learning deep network architectures [55, 57], and they tend to either learn a one-class classifier [140, 57], or to somehow leverage as novelty score, the reconstruction error of the encoder-decoder architecture they are based on [141, 56].

In this work, we introduce a new encoder-decoder architecture as well, which is based on adversarial autoencoders [142]. However, we do not train a one-class classifier, instead, we learn the probability distribution of the inliers. Therefore, the novelty test simply becomes the evaluation of the probability of a test sample, and rare samples (outliers) fall below a given threshold. We show that this approach allows us to effectively use the decoder network to learn the parameterized manifold shaping the inlier distribution, in conjunction with the probability distribution of the (parameterizing) latent space. The approach is made computationally feasible because for a given test sample we linearize the manifold, and show that with respect to the local manifold coordinates the data model distribution factorizes into a component dependent on the manifold (decoder network plus latent distribution), and another one dependent on the noise, which can also be learned offline.

We named the approach *generative probabilistic novelty detection (GPND)* because we compute the probability distribution of the full model, which includes the signal plus noise portion, and because it relies on being able to also generate data samples. We are mostly concerned with novelty detection using images, and with controlling the distribution of the latent space to ensure good generative reproduction of the inlier distribution. This is essential not so much to ensure good image generation, but for the correct computation of the novelty score. This aspect has been overlooked by the deep learning literature so far, since the focus has been only on leveraging the reconstruction error. We do leverage that as well, but we show in our framework that the reconstruction error affects only the noise portion of the model. In order to control the latent distribution and image generation we learn an adversarial autoencoder network with two discriminators that address these two issues.

Section 3.2 introduces the GPND framework, and Section 3.3 describes the training and architecture of the adversarial autoencoder network. Section 3.5 describe a rich set of experiments showing that GPND is very effective and produces state-of-the-art results on

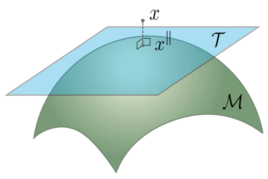


Figure 3.1: **Manifold schematic representation.** This figure shows connection between the parametrized manifold $\mathcal{M}$, its tangent space $\mathcal{T}$, data point x and its projection $x^{\parallel}$.

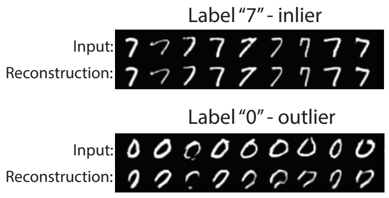


Figure 3.2: **Reconstruction of inliers and outliers.** This figure showns reconstructions for the autoencoder network that was trained on inlier of label "7" of MNIST [143] dataset. First line is input of inliers of label "7", the second line shows corresponding reconstructions. The third line corresponds to input of outlier of label "0" and the forth line, corresponding reconstructions.

several benchmarks.

3.2 Generative Probabilistic Novelty Detection

We assume that training data points $x_1, \dots, x_N$, where $x_i \in \mathbb{R}^m$, are sampled, possibly with noise ξ_i, from the model

$$x_i = f(z_i) + \xi_i \qquad i = 1, \cdots, N \; , \tag{3.1}$$

where $z_i \in \Omega \subset \mathbb{R}^n$. The mapping $f : \Omega \rightarrow \mathbb{R}^m$ defines $\mathcal{M} \equiv f(\Omega)$, which is a parameterized manifold of dimension n, with $n < m$. We also assume that the Jacobi matrix of f is full rank at every point of the manifold. In addition, we assume that there is another mapping $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$, such that for every $x \in \mathcal{M}$, it follows that $f(g(x)) = x$, which means that g acts as the inverse of f on such points.

Given a new data point $\bar{x} \in \mathbb{R}^m$, we design a novelty test to assert whether $\bar{x}$ was sampled from model (3.1). We begin by observing that $\bar{x}$ can be non-linearly projected onto $\bar{x}^{\parallel} \in \mathcal{M}$ via $\bar{x}^{\parallel} = f(\bar{z})$, where $\bar{z} = g(\bar{x})$. Assuming f to be smooth enough, we perform a

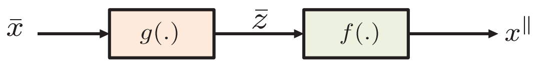


Figure 3.3: **Projection of the sample datapoint.** This figure shows that projection of the input data point can be represented as a sequence of applying functions f and g.

linearization based on its first-order Taylor expansion

$$f(z) = f(\bar{z}) + J_f(\bar{z})(z - \bar{z}) + O(\|z - \bar{z}\|^2) \ , \tag{3.2}$$

where $J_f(\bar{z})$ is the Jacobi matrix computed at $\bar{z}$, and $\| \cdot \|$ is the L$_2$ norm. We note that $\mathcal{T} = \text{span}(J_f(\bar{z}))$ represents the tangent space of f at $\bar{x}^{\|}$ that is spanned by the n independent column vectors of $J_f(\bar{z})$, see Figure 3.1. Also, we have $\mathcal{T} = \text{span}(U^{\|})$, where $J_f(\bar{z}) = U^{\|}SV^{\top}$ is the singular value decomposition (SVD) of the Jacobi matrix. The matrix $U^{\|}$ has rank n, and if we define $U^{\perp}$ such that $U = [U^{\|}U^{\perp}]$ is a unitary matrix, we can represent the data point $\bar{x}$ with respect to the local coordinates that define the tangent space $\mathcal{T}$, and its orthogonal complement $\mathcal{T}^{\perp}$.

This is done by computing

$$\bar{w} = U^{\top}\bar{x} = \begin{bmatrix} {U^{\|}}^{\top}\bar{x} \\ {U^{\perp}}^{\top}\bar{x} \end{bmatrix} = \begin{bmatrix} \bar{w}^{\|} \\ \bar{w}^{\perp} \end{bmatrix} \ , \tag{3.3}$$

where the rotated coordinates $\bar{w}$ are decomposed into $\bar{w}^{\|}$, which are parallel to $\mathcal{T}$, and $\bar{w}^{\perp}$ which are orthogonal to $\mathcal{T}$.

We now indicate with $p_X(x)$ the probability density function describing the random variable X, from which training data points have been drawn. Also, $p_W(w)$ is the probability density function of the random variable W representing X after the change of coordinates. The two distributions are identical. However, we make the assumption that the coordinates $W^{\|}$, which are parallel to $\mathcal{T}$, and the coordinates $W^{\perp}$, which are orthogonal to $\mathcal{T}$, are statistically independent. This means that the following holds

$$p_X(x) = p_W(w) = p_W(w^{\|}, w^{\perp}) = p_{W^{\|}}(w^{\|}) p_{W^{\perp}}(w^{\perp}) \ . \tag{3.4}$$

This is motivated by the fact that in (3.1) the noise ξ is assumed to predominantly deviate the point x away from the manifold $\mathcal{M}$ in a direction orthogonal to $\mathcal{T}$. This means that $W^{\perp}$ is primarely responsible for the noise effects, and since noise and drawing from the manifold are statistically independent, so are $W^{\|}$ and $W^{\perp}$.

From (3.4), given a new data point $\bar{x}$, we propose to perform novelty detection by executing the following test

$$\boxed{p_X(\bar{x}) = p_{W^{\|}}(\bar{w}^{\|}) p_{W^{\perp}}(\bar{w}^{\perp}) = \begin{cases} \geq \gamma & \Longrightarrow \quad \text{Inlier} \\ < \gamma & \Longrightarrow \quad \text{Outlier} \end{cases}} \tag{3.5}$$

where γ is a suitable threshold.

3.2.1 Computing the distribution of data samples

The novelty detector (3.5) requires the computation of $p_{W^{\|}}(w^{\|})$ and $p_{W^{\perp}}(w^{\perp})$. Given a test data point $\bar{x} \in \mathbb{R}^m$ its non-linear projection onto $\mathcal{M}$ is $\bar{x}^{\|} = f(g(\bar{x}))$. Therefore, $\bar{w}^{\|}$ can be written as $\bar{w}^{\|} = {U^{\|}}^{\top} \bar{x} = {U^{\|}}^{\top} (\bar{x} - \bar{x}^{\|}) + {U^{\|}}^{\top} \bar{x}^{\|} = {U^{\|}}^{\top} \bar{x}^{\|}$, where we have made the approximation that ${U^{\|}}^{\top} (\bar{x} - \bar{x}^{\|}) \approx 0$. Since $\bar{x}^{\|} \in \mathcal{M}$, then in its neighborhood it can be parameterized as in (3.2), which means that $w^{\|}(z) = {U^{\|}}^{\top} f(\bar{z}) + SV^{\top}(z - \bar{z}) + O(\|z - \bar{z}\|^2)$. Therefore, if Z represents the random variable from which samples are drawn from the parameterized manifold, and $p_Z(z)$ is its probability density function, then it follows that

$$p_{W^{\|}}(w^{\|}) = |\text{det} S^{-1}| \, p_Z(z) \ , \tag{3.6}$$

since V is a unitary matrix. We note that $p_Z(z)$ is a quantity that is independent from the linearization (3.2), and therefore it can be learned offline, as explained in Section 3.4.

In order to compute $p_{W^\perp}(w^\perp)$, we approximate it with its average over the hypersphere $\mathcal{S}^{m-n-1}$ of radius $\|w^\perp\|$, giving rise to

$$p_{W^\perp}(w^\perp) \approx \frac{\Gamma\left(\frac{m-n}{2}\right)}{2\pi^{\frac{m-n}{2}}\|w^\perp\|^{m-n}} p_{\|W^\perp\|}(\|w^\perp\|) \ , \tag{3.7}$$

where $\Gamma(\cdot)$ represents the gamma function. This is motivated by the fact that noise of a given intensity will be equally present in every direction. Moreover, its computation depends on $p_{\|W^\perp\|}(\|w^\perp\|)$, which is the distribution of the norms of $w^\perp$, and which can easily be learned offline by histogramming the norms of $\bar{w}^\perp = {U^\perp}^\top \bar{x}$.

3.3 Manifold learning with adversarial autoencoders

In this section we describe the network architecture and the training procedure for learning the mapping f that define the parameterized manifold $\mathcal{M}$, and also the mapping g. The mappings g and f represent and are modeled by an *encoder* network, and a *decoder* network, respectively. Similarly to previous work on novelty detection [144, 145, 146, 56, 57, 141], such networks are based on autoencoders [147, 3].

The autoencoder network and training should be such that they reproduce the manifold $\mathcal{M}$ as closely as possible. For instance, if $\mathcal{M}$ represents the distribution of images depicting a certain object category, we would want the estimated encoder and decoder to be able to generate images as if they were drawn from the real distribution. Differently from previous work, we require the latent space, represented by z, to be close to a known distribution, preferably a normal distribution, and we would also want each of the components of z to be maximally informative, which is why we require them to be independent random variables. Doing so facilitates learning a distribution $p_Z(z)$ from training data mapped onto the latent space Ω. This means that the autoencoder has generative properties because by sampling

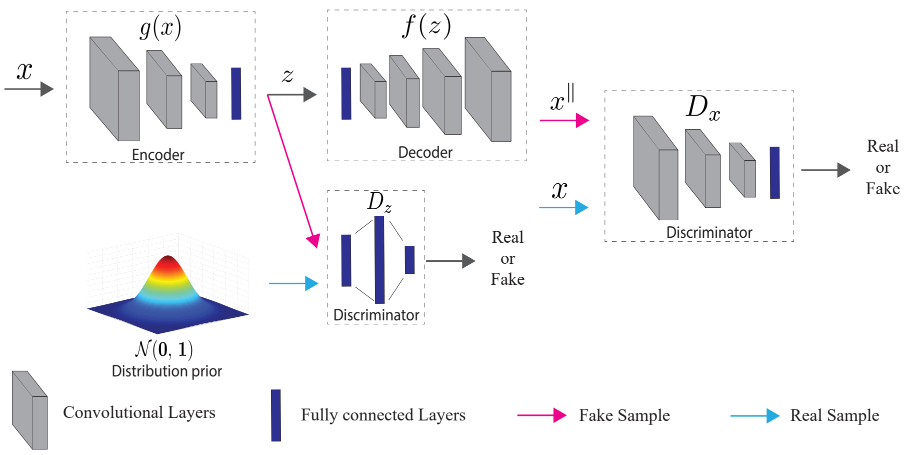


Figure 3.4: **AAE based architecture of the network for manifold learning** as presented in our NeurIPS 2018 paper [71]. It is based on Adversarial Autoencoders (AAE) [142] and additionaly has a discriminator that adds adversarial component in image space.

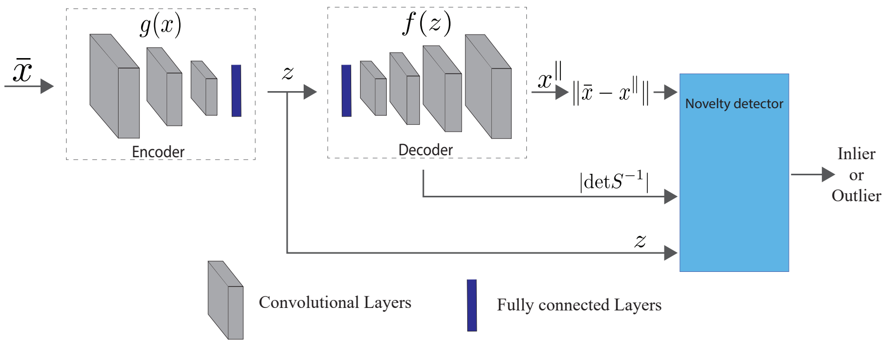


Figure 3.5: **Inference time architecture.** Architecture for performing novelty test as presented in our NeurIPS 2018 paper [71]. Only two networks remained from the training step: encoder g and decoder f.

from $p_Z(z)$ we would generate data points $x \in \mathcal{M}$. Note that differently from GANs [54] we also require an encoder function g.

Variational Auto-Encoders (VAEs) [9] are known to work well in the presence of continuous latent variables, and they can generate data from a randomly sampled latent space. VAEs utilize stochastic variational inference and minimize the Kullback-Leibler (KL) divergence penalty to impose a prior distribution on the latent space that encourages the encoder to learn the modes of the prior distribution. Adversarial Autoencoders (AAEs) [142], in contrast to VAEs, use an adversarial training paradigm to match the posterior distribution of the latent space with the given distribution. One of the advantages of AAEs over VAEs is that the adversarial training procedure encourages the encoder to match the whole distribution of the prior.

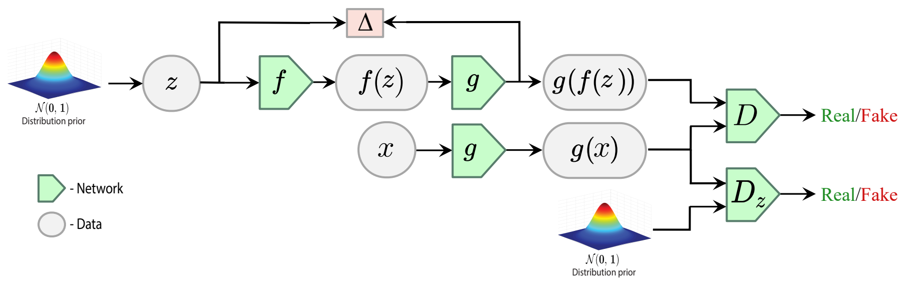
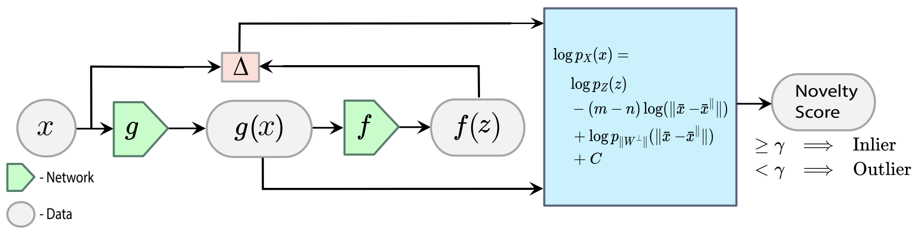


Figure 3.6: **Architecture of the network for manifold learning.** It is based on Adversarial Latent Autoencoders (ALAE) [101] and similarly to Adversarial Autoencoders (AAE) [142] has additional discriminator that aims to bring latent distribution close to normal.



x
g
$g(x)$
f
$f(z)$
Δ
$\log p_X(x) =$
$\log p_Z(z)$
$-(m-n)\log(\|\bar{x} - \bar{x}^{\|}\|)$
$+\log p_{\|W^{\perp}\|}(\|\bar{x} - \bar{x}^{\|}\|)$
$+C$
Novelty Score
$\geq \gamma \implies$ Inlier
$< \gamma \implies$ Outlier
- Network
- Data

Figure 3.7: **Inference time architecture.** Architecture for performing novelty test. Only two networks remained from the training step: encoder g and decoder f.

Similarly to AAEs, PixelGAN autoencoders [148] introduce the adversarial component to impose a prior distribution on the latent code, but the architecture is significantly different since it is conditioned on the latent code.

Our initial version of GPND, as in our NeurIPS 2018 paper [71], uses architecture based on AAE presented in 3.4 and 3.5.

Unfortunately, since we are concerned with working with images, both AAEs and VAEs tend to produce examples that are often far from the real data manifold.

3.3.1 GPND with Adversarial Autoencoders

We start from a basic AAE architecture and similarly to [149, 57] we add an adversarial training criterion to match the output of the decoder with the distribution of real data. This allows us to reduce blurriness and add more local details to the generated images.

Our full objective consists of three terms. First, we use an adversarial loss for matching the distribution of the latent space with the prior distribution, which is a normal with 0 mean, and standard deviation 1, $\mathcal{N}(0,1)$. Second, we use an adversarial loss for matching the distribution of the decoded images from z and the known, training data distribution. Third, we use an autoencoder loss between the decoded images and the encoded input image. Figure 3.4 shows the architecture configuration.

Adversarial losses

For the discriminator D_z, we use the following adversarial loss:

$$\mathcal{L}_{adv-d_z}(x, g, D_z) = E[\log(D_z(\mathcal{N}(0,1)))] + E[\log(1 - D_z(g(x)))] \ , \tag{3.8}$$

where the encoder g tries to encode x to a z with distribution close to $\mathcal{N}(0,1)$. D_z aims to distinguish between the encoding produced by g and the prior normal distribution. Hence, g tries to minimize this objective against an adversary D_z that tries to maximize it.

Similarly, we add the adversarial loss for the discriminator D_x:

$$\mathcal{L}_{adv-d_x}(x, D_x, f) = E[\log(D_x(x))] + E[\log(1 - D_x(f(\mathcal{N}(0,1))))] \ , \tag{3.9}$$

where the decoder f tries to generate x from a normal distribution $\mathcal{N}(0,1)$, in a way that x is as if it was sampled from the real distribution. D_x aims to distinguish between the decoding generated by f and the real data points x. Hence, f tries to minimize this objective against an adversary D_x that tries to maximize it.

Autoencoder loss

We also optimize jointly the encoder g and the decoder f so that we minimize the reconstruction error for the input x that belongs to the known data distribution.

$$\mathcal{L}_{error}(x, g, f) = -E_z[\log(p(f(g(x))|x))] \ , \tag{3.10}$$

where $\mathcal{L}_{error}$ is minus the expected log-likelihood, i.e., the reconstruction error. This loss does not have an adversarial component but it is essential to train an autoencoder. By minimizing this loss we encourage g and f to better approximate the real manifold.

Full objective

The combination of all the previous losses gives

$$\mathcal{L}(x, g, D_z, D_x, f) = \mathcal{L}_{adv-d_z}(x, g, D_z) + \mathcal{L}_{adv-d_x}(x, D_x, f) + \lambda \mathcal{L}_{error}(x, g, f) \ , \quad (3.11)$$

where λ is a parameter that strikes a balance between the reconstruction and the other losses. The autoencoder network is obtained by minimizing (3.11), giving:

$$\hat{g}, \hat{f} = \arg \min_{g,f} \max_{D_x, D_z} \mathcal{L}(x, g, D_z, D_x, f) \ . \quad (3.12)$$

The model is trained using stochastic gradient descent by doing alternative updates of each component as follows

- Maximize $\mathcal{L}_{adv-d_x}$ by updating weights of D_x;
- Minimize $\mathcal{L}_{adv-d_x}$ by updating weights of f;
- Maximize $\mathcal{L}_{adv-d_z}$ by updating weights of D_z;
- Minimize $\mathcal{L}_{error}$ and $\mathcal{L}_{adv-d_z}$ by updating weights of g and f.

3.3.2 GPND with Adversarial Latent Autoencoders

Unfortunately, the presented modified AAE approach still tends to produce examples that are often far from the real data manifold.

This is because the reciprocity of the network is achieved only from a reconstruction loss that is typically a pixel-wise similarity loss between input and output image. Such loss often causes the generated images to be blurry, which has a negative effect on the proposed approach. The blurriness of reconstructions is typical for VAEs, AAE, and other autoencoders that achieve reciprocity with a similarity criterion in image space. The main reason for this is explicit likelihood training paradigm [150] as apposed to implicit. Likelihood maximization combined with poor decoder distribution results in blurry reconstructions. Practically, the only feasible decoder distribution for VAEs is factorized Gaussian distribution, where all pixels are assumed independent, which makes likelihood maximization equivalent to MSE minimization.

We updated the GPND architecture and we utilize paradigm, where the reciprocity is achieved in latent space [151, 152, 101], as opposed to image space. We adopt architecture from [101], which is distinctive from other autoencoder architectures by enforcing reciprocity in latent space, not in image space, which eliminates the need for pixel-wise reconstruction loss. Moreover, we also combine the adversarial training criterion with AAEs, which results in having two adversarial losses: one to impose a prior on the latent space distribution, and the second one to impose a prior on the output distribution.

Our full objective consists of three terms. First, we use an adversarial loss for matching the distribution of the latent space with the prior distribution, which is a normal with 0 mean, and standard deviation 1, $\mathcal{N}(0, 1)$. Second, we use an adversarial loss for matching the distribution of the images decoded from normal distribution and the known training data distribution. Third, we impose reciprocity on the latent space using ℓ_2 loss on latent vector reconstruction error. Figure 3.6 shows the architecture configuration.

Adversarial Latent Autoencoders

We use the architecture of Adversarial Latent Autoencoders, introduced in [101], which we modify in two ways. First, we eliminate intermediate space $\mathcal{W}$ and function F, for the following reasons. Distribution of the intermediate space is unconstrained, thus computing $p_w(w),\ w \in \mathcal{W}$ would require as to use chain rule and find the inverse of F function.

Function F is a normalizing flow and can be reversed, but unfortunately, in practice, that gives highly unstable results. Computing $F^{-1}(E(x))$ of an input x results in values that have an extremely low probability of being sampled from the normal distribution. We think that it is likely the case is $E(x)$ returns point that does not lie perfectly on the manifold $F(z),\ z \in \mathcal{N}(0,1)$, and the error gets amplified a lot due to the computing of the inverse mapping F^{-1}. Eliminating the intermediate space $\mathcal{W}$ and function F means that we train the encoder E to regress back to $\mathcal{Z}$ space directly. The proposed architecture consist of four networks: generator f, encoder g, discriminator D and z-space discriminator D_z. Discriminator D together with encoder g work on aligning distribution of the output of the generator with real data distribution, while z-space discriminator D_z aims at aligning distribution of the encoder output with normal distribution.

Adversarial losses

According to generalized GAN formulation [25], the value function $V(\mathtt{G}, \mathtt{D})$ of the minimax game for a generic GAN can be defined as

$$V(\mathtt{G}, \mathtt{D}) = E_{p_{\mathcal{D}}(x)}[\eta(\mathtt{D}(x))] + E_{p(z)}[\eta(-\mathtt{D}(\mathtt{G}(z)))] \ , \tag{3.13}$$

where $\eta : \mathbb{R} \to \mathbb{R}$ is a concave function.

Following [101], we decompose the discriminator network $\mathtt{D}$ into two other networks: g and D, see Figure 3.6. This means that $\mathtt{D} = D \circ g$. We assume that the *latent* space on the input of generator network f and interface between g and D are the same and we indicate them as $\mathcal{Z}$.

Consequently, the two alternating minimax games of the proposed architecture can be defined as followes using the generalized value function $V(\cdot, \cdot)$:

$$\min_f \max_{g,D} V(f, D \circ g) \tag{3.14}$$

$$\min_g \max_{D_z} V(g \circ f, D_z) \tag{3.15}$$

When training g and D, real data is data coming from the image samples of the dataset. When training g and D_z - real data is data sampled from the normal distribution.

Distribution alignment

The output distribution of f can be expressed as

$$q(x) = \int_w q_f(x|z) p_Z(z) \mathrm{d}z \ , \tag{3.16}$$

where distribution $q_f(x|z)$ represents generator network f and $p_Z(z)$ is probability density of the normal distribution $\mathcal{N}(0, 1)$. Thus, the output of the encoder network g, when fed with output from generator will be:

$$q_g(z) = \int_x q_g(z|x) q(x) \mathrm{d}x \ , \tag{3.17}$$

where distribution $q_g(z|x)$ represents encoder network g. Adversarial training aims to bring close distributions of $q(x)$ and distribution of real data $p_{\mathcal{D}}(x)$, and consequently that lead to $q_g(z) = q_{g,\mathcal{D}}(z)$, where $q_{g,\mathcal{D}}(z)$ is marginal distribution of the output of encoder network, when fed with real data, e.g. when in (2.4) $q(x)$ is replaced with $p_{\mathcal{D}}(x)$. Additionaly to that, we also try to bring both distributions $q_g(z)$ and $q_{g,\mathcal{D}}(z)$ close to normal distribution with additional discriminator D_z.

Reciprocity

In order to achieve reciprocity, we learn such $q_g\left(z|x\right)$ and $q_f\left(x|z\right)$ that maximize marginal likelihood $q_g(z)$. Simultaneously to optimizing two GAN losses, we also maximize the expectation of $q_g\left(z|x\right)$ over the distribution of $q_f\left(x|z\right)$.

$$\arg\max_{g,f} E_{z\sim\mathcal{N}(0,1)}\left[E_{x\sim q_f(z|w)}\log q_g\left(z|x\right)\right] \tag{3.18}$$

Assuming the encoder distribution $q_g\left(z|x\right)$ to be factorised Gaussian, equation (3.18) becomes MSE loss in latent space $\mathcal{Z}$:

$$\arg\min_{g,f} E_{z\sim\mathcal{N}(0,1)}\|z - g\circ f(z)\|_2^2 \tag{3.19}$$

Autoencoder training

Following [153, 101] we use non-saturating loss [54] and set $\eta(\cdot)$ to a softplus function $\eta(t) = \log(1+\exp(t))$. In addition, we utilize R_1 gradient regularization techniques [118, 26, 119], which is a zero-centered gradient penalty term which is computed only on real data applied only to the networks g, D, Dz.

Following [22, 153, 101] we use learning rate equalization we find it a crucial component for stable GAN training. We implement learning rate equalization, described in [22] by changing the learning rate for each layer individually, ensuring that all layers are trained at the same speed. This slightly differs from learning rate equalization implemented in [22], because we eliminate intermediate weight scaling operation, more details are explained in [101]

The model is trained using stochastic gradient descent by doing alternations of three steps:

- Adversarial step. Updating encoder g and discriminator D. Simultaneously maximize $V(f, D\circ g)$ and minimize $V(g\circ f, D_z)$ with respect to weights of g and D;

- Adversarial step. Updating generator f and discriminator D_z. Simultaneously minimize $V(f, D \circ g)$ and maximize $V(g \circ f, D_z)$ with respect to weights of f and D_z;

- Reciprocal step. Updating encoder-generator pair. Minimize $\|z - g \circ f(z)\|_2^2$ by updating weights of f and g.

See Figure 3.6 for details.

3.3.3 Performing inference

During the inference, the input data is fed into encoder network g and then to generator network f, hence the other two discriminator networks are dropped. The resulting latent space and the reconstruction space is used to perform a novelty test, see Figure 3.7.

3.4 Implementation Details and Complexity

The novelty detector (3.5) requires the computation of $p_{W^{\|}}(w^{\|})$ and $p_{W^{\perp}}(w^{\perp})$. Given a test data point $\bar{x} \in \mathbb{R}^m$ its non-linear projection onto $\mathcal{M}$ is $\bar{x}^{\|} = f(g(\bar{x}))$.

As reflected in 3.3, $\bar{w}^{\|}$ is a component of $\bar{w}$ that is parallel to the tangent space $\mathcal{T}$, that is defined as $\bar{w}^{\|} = U^{\|^\top}\bar{x}$. Similarly, as reflected in 3.3, $\bar{w}^{\perp}$ is a component of $\bar{w}$ that is orthogonal to the tangent space $\mathcal{T}$, that is defined as $\bar{w}^{\perp} = U^{\perp^\top}\bar{x}$.

We note that since $U^{\perp}$ is a null space of the linearization of $f(z)$ at the point $\bar{z}$, s.t. $\bar{x}^{\|} = f(\bar{z})$, thus:

$$U^{\perp^\top}\bar{x}^{\|} = 0 \ , \tag{3.20}$$

Eq. 3.20 means that distance between $x^{\|}$ and the tangent plane $\mathcal{T}$ is zero.

We introduce $\bar{x}^{\perp} = \bar{x} - \bar{x}^{\|}$ - which is a vector connecting $\bar{x}$ and its projection onto the manifold $\mathcal{M}$ - $\bar{x}^{\|}$. Thus, $\bar{x}^{\|}$ lies in the span of $U^{\|}$ and $\bar{x}^{\perp}$ lies in the span of $U^{\perp}$. Since bases $U^{\|}$ and $U^{\perp}$ are orthogonal, points $\bar{x}^{\|}$ and $\bar{x}^{\perp}$ are also orthogonal. Thus, the difference between $\bar{x}$ and its projection $\bar{x}^{\|}$ fully lies in the complement to the tangent space $\mathcal{T}^{\top}$, so that projection of the difference onto the tangent spaces is zero:

$$U^{\|\top}(\bar{x} - \bar{x}^{\|}) = U^{\|\top}\bar{x}^{\perp} = 0 \ , \tag{3.21}$$

Using 3.21 we can rewrite $\bar{w}^{\|}$ in terms of the $\bar{x}^{\|}$:

$$\bar{w}^{\|} = U^{\|\top}\bar{x} = U^{\|\top}(\bar{x}^{\perp} + \bar{x}^{\|}) = U^{\|\top}\bar{x}^{\|} \tag{3.22}$$

Similarly, we can rewrite $\bar{w}^{\perp}$ in terms of the reconstruction difference: $\bar{x} - \bar{x}^{\|}$:

$$\bar{w}^{\perp} = U^{\perp\top}\bar{x} = U^{\perp\top}\bar{x} - U^{\perp\top}\bar{x}^{\|} = U^{\perp\top}(\bar{x} - \bar{x}^{\|}) \tag{3.23}$$

On the other hand, as discussed in 3.2.1, in order to compute $p_{W^{\perp}}(w^{\perp})$, we approximate it with its average over the hypersphere $\mathcal{S}^{m-n-1}$ of radius $\|w^{\perp}\|$, thus we are interested only in euclidean norm of $\bar{w}^{\perp}$.

Taking into account that bases $U^{\|}$ and $U^{\perp}$ are orthogonal and according to pythagoras theorem:

$$\|U^{\top}(\bar{x} - \bar{x}^{\|})\|^2 = \|U^{\|\top}(\bar{x} - \bar{x}^{\|})\|^2 + \|U^{\perp\top}(\bar{x} - \bar{x}^{\|})\|^2 \tag{3.24}$$

Using 3.21 we can further conclude that:

$$\|U^{\top}(\bar{x} - \bar{x}^{\|})\| = \|U^{\perp\top}(\bar{x} - \bar{x}^{\|})\| \tag{3.25}$$

Since U is a unitary matrix an thus it does not change length of the vector:

$$\|\bar{x} - \bar{x}^{\|}\| = \|U^{\perp\top}(\bar{x} - \bar{x}^{\|})\| \tag{3.26}$$

Thus from 3.23 and 3.26 we derive that euclidian norm of $\bar{w}^{\perp}$ equals to reconstruction error:

$$\|\bar{w}^{\perp}\| = \|\bar{x} - \bar{x}^{\|}\| \tag{3.27}$$

Thus, expression 3.7 for $p_{W^{\perp}}(w^{\perp})$ can be rewritten as:

$$p_{W^{\perp}}(w^{\perp}) \approx \frac{\Gamma\left(\frac{m-n}{2}\right)}{2\pi^{\frac{m-n}{2}}\|\bar{x} - \bar{x}^{\|}\|^{m-n}} p_{\|W^{\perp}\|}(\|\bar{x} - \bar{x}^{\|}\|) \ , \tag{3.28}$$

This means that we don't need to compute Full-SVD in order to obtain $U^{\perp}$, neither we need to compute $\bar{w}^{\perp}$ or $\bar{w}^{\|}$.

Taking into account 3.22 we transform 3.2 by multiplying both sides by $U^{\|^{\top}}$ and defining $U^{\|^{\top}} f(z)$ as function $w^{\|}(z)$:

$$\begin{aligned} w^{\|}(z) &= U^{\|^{\top}} f(\bar{z}) + U^{\|^{\top}} U^{\|} S V^{\top}(\bar{z})(z - \bar{z}) \\ &\quad + U^{\|^{\top}} O(\|z - \bar{z}\|^2) \ , \\ &= U^{\|^{\top}} f(\bar{z}) + S V^{\top}(\bar{z})(z - \bar{z}) \\ &\quad + U^{\|^{\top}} O(\|z - \bar{z}\|^2) \ , \end{aligned} \tag{3.29}$$

Therefore, if Z represents the random variable from which samples are drawn from the parameterized manifold, and $p_Z(z)$ is its probability density function, then it follows that

$$p_{W^{\|}}(w^{\|}) = |\text{det} S^{-1}| \, p_Z(z) \ , \tag{3.30}$$

Since V is a unitary matrix. We note that $p_Z(z)$ is a quantity that is independent of the linearization (3.2), and therefore it can be learned offline.

After learning the autoencoder network, by mapping the training set onto the latent space through g, we fit to the data a generalized Gaussian distribution and estimate $p_Z(z)$.

Computing a derivative, i.e. the Jacoby matrix J_f, can be done numerically or using autograd capabilities of the deep learning frameworks. In the case of numerical computation with central differences it will require $2 \cdot n$ of forward passes. If autograd is used, then it will require m backward passes, since all practical implementations of autograd in deeplearning frameworks can only compute derivative of a scalar. Thus both cases are very slow and in $m+1$ or $2 \cdot n+1$ times slower compared to single forward pass. If we ignore the component $|\text{det} S^{-1}|$, single forward pass is enough for the rest of the computations.

Experimentally we found that component $|\text{det} S^{-1}|$ has minor contribution, and we demonstrate that in the ablation study subsection 3.5.2. For the rest of the experimental study we make a decision to ignore the Jacobian component completly and approximate $p_{W^{\|}}(w^{\|})$ as:

$$p_{W^{\|}}(w^{\|}) \approx p_Z(z) \ , \tag{3.31}$$

Thus, this approximation completely eliminates need for computing Jacoby matrix J_f and SVD. For the detection test 3.5 we compute log likelihood of the test data point as follows:

$$\begin{aligned} \log p_X(x) = &\log p_Z(z) - (m-n)\log(\|\bar{x} - \bar{x}^{\|}\|) \\ &+ \log p_{\|W^{\perp}\|}(\|\bar{x} - \bar{x}^{\|}\|) \\ &+ C \ . \end{aligned} \tag{3.32}$$

Where $C = \log \Gamma\left(\frac{m-n}{2}\right) - \frac{m-n}{2} \log 2\pi$. We compute log-likelihood up to a constant C that we ignore because of the tuneable threshold in the detection test.

In addition, by histogramming the quantities $\|x - x^{\|}\|$ we estimate $p_{\|W^{\perp}\|}(x - x^{\|})$. The entire training procedure takes about 4 hours with a high-end PC with one NVIDIA TITAN X for a MNIST-like dataset.

When a sample is tested, the procedure entails mainly computing a forward pass of the autoencoder. The computational cost of evaluating 3.32 is neglectable compared to the cost of the forward pass.

Table 3.1: Results of tuning α and β parameters on validation set of MNIST dataset.

	Digits										Mean	SD
	0	1	2	3	4	5	6	7	8	9		
α	11.73	11.02	9.495	17.19	4.333	17.45	17.55	17.34	15.84	13.05	13.45	4.68
β	0.3807	0.4994	0.1765	0.4937	0.5665	0.4327	0.6939	0.4295	0.2848	0.4273	0.4397	0.1517

3.4.1 Model correction

Experimentally we found that performance of the method can be significantly improved by utilizing model correction described below. In 3.2.1, one of the assumptions was that noise will be equally present in every direction, which is a strong assumption that generally would not hold, leading to wrong exponents in 3.7. To compensate for that, we suggest to modify 3.32 with tunable parameters: α, and β.

$$\begin{aligned}\log p_X(x) =& \alpha \log p_Z(z) - \beta(m-n)\log(\|\bar{x} - \bar{x}^{\|}\|) \\ & + \log p_{\|W^{\perp}\|}(\|\bar{x} - \bar{x}^{\|}\|) \\ & + C\ .\end{aligned} \tag{3.33}$$

We found that those parameters are not very sensible to the data, and we use $\alpha = 13$, and $\beta = 0.4$. This pair of parameters was found by taking the average of optimals among all classes of MNIST dataset, but was also used for experiments on CIFAR-10, FashionMNIST, and Coil-100 datasets. Table 3.5 and Table 3.6 show also results obtained by tuning α, and β on validation set and results obtained without tuning, by setting $\alpha = 13$, and $\beta = 0.4$. For each digit, using grid search, we found optimal pair of α, and β on validation set. The mean values of the optimal parameters α, and β are 13.45 and 0.4397 with standard deviation of 4.68 and 0.1517, see Table 3.1.

3.5 Experiments

We evaluate our novelty detection approach, which we call *Generative Probabilistic Novelty Detection (GPND)*, against several state-of-the-art approaches and with several performance measures. Our initial version of GPND, as in our NeurIPS 2018 paper [71] as described in 3.3.1, we call GPND AAE.

We evaluate performance in one-class novelty detection setting using F_1 measure and the area for the ROC curve (AUROC). In addition to that, under the out-of-distribution setting we also use the FPR at 95% TPR (i.e., the probability of an outlier to be misclassified as inlier), the Detection Error (i.e., the misclassification probability when TPR is 95%), and the area under the precision-recall curve (AUPR) when inliers (AUPR-In) or outliers (AUPR-Out) are specified as positives. All reported results are from our publicly available implementation[1] and the initial version of GPND, as in our NeurIPS 2018 paper [71] is available at[2], based on the deep machine learning framework PyTorch [122].

For one class novelty detection we implement Protocol 1 and Protocol 2 setting [154].

Protocol 1 : Dataset is split into training, validation, and testing sets at random. Testing set contains 20$\%$ of all samples, while training and validation sets contain the remaining 80$\%$. We use 60$\%$ of all samples as training set, 20$\%$ as validation set for all cases except for the Coil-100 dataset [155], where we do not use validation set due to the small size of the dataset. Outlier samples from training set are removed and they are not used during testing or training. The area under the ROC curve (AUROC) is computed for the case when outlier samples constitute half of the testing set. F_1 measure is computed for a tabulated outlier-inlier proportions.

Protocol 2 : For MNIST and CIFAR-10 official splits for training and testing sets are used. Outlier samples from training set are removed and they are not used during testing or training. The area under the ROC curve (AUROC) is computed for the case when outlier samples constitute half of the testing set.

[1]https://github.com/wvuvl/GPND2
[2]https://github.com/wvuvl/GPND

3.5.1 Datasets

We evaluate GPND on the following datasets.

MNIST [143] contains $70,000$ handwritten digits from 0 to 9 labeled accordingly with ten classes. Each of the ten classes is used as inlier class and the rest of the categories are used as outliers. Has official split into training and testing sets constituting $60,000$ and $10,000$ samples

CIFAR-10 [156] contains $60,000$ 32×32 colour images in 10 classes. Each of the ten classes is used as inlier class and the rest of the categories are used as outliers. Has official split into training and testing sets constituting $50,000$ and $10,000$ samples

The Coil-100 dataset [155] contains $7,200$ images of 100 different objects. Each object has 72 images taken at pose intervals of 5 degrees. We downscale the images to size 32×32. We take randomly n categories, where $n \in 1, 4, 7$ and randomly sample the rest of the categories for outliers. We repeat this procedure 30 times.

Fashion-MNIST [157] is a new dataset comprising of 28×28 grayscale images of $70,000$ fashion products from 10 categories, with $7,000$ images per category. The training set has $60,000$ images and the test set has $10,000$ images. Fashion-MNIST shares the same image size, data format and the structure of training and testing splits with the original MNIST.

COVID-Net [103] is a new dataset comprising of large set of chest x-ray that combine covid patients, pneumonia cases and normal chest x-rays. We made custom annotation for this dataset in order to align images for easier training of autoencoder. After alignment is done, we downscale the images to resolution 128×128. Training split of the dataset consists of: normal cases - 7956, pneumonia - 5373, covid - 425 images. Testing split of the dataset consists of: normal cases - 880, pneumonia - 589, covid - 98 images. Annotation sample is shown in Figure 3.8. Samples of the aligned dataset are shown in Figure 3.9

Others. We compare GPND with ODIN [66] using their protocol. For inliers we used samples from CIFAR-10(CIFAR-100) [156], which is a publicly available dataset of small images of size 32×32, which have each been labeled to one of 10 (100) classes. Each class

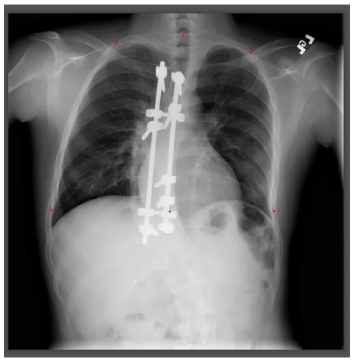


Figure 3.8: **COVID-Net**. Sample of the annotation of the source image from COVID-Net dataset.

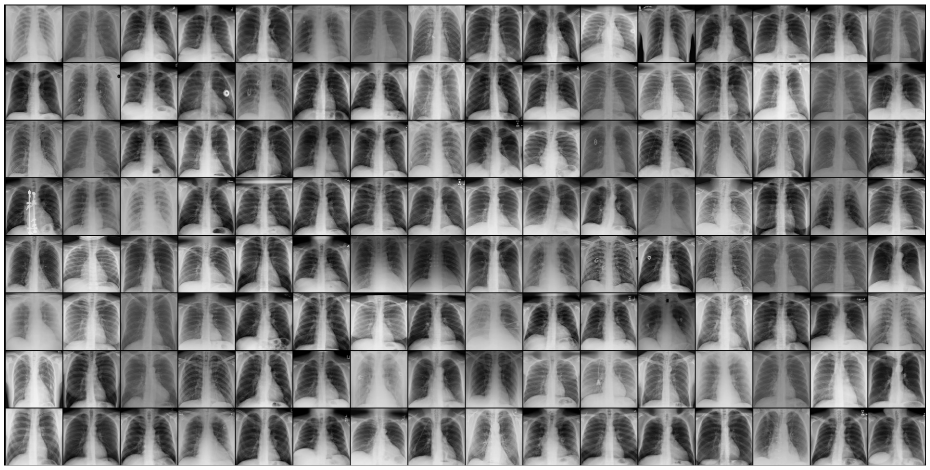


Figure 3.9: **COVID-Net**. Samples of the aligned COVID-Net dataset.

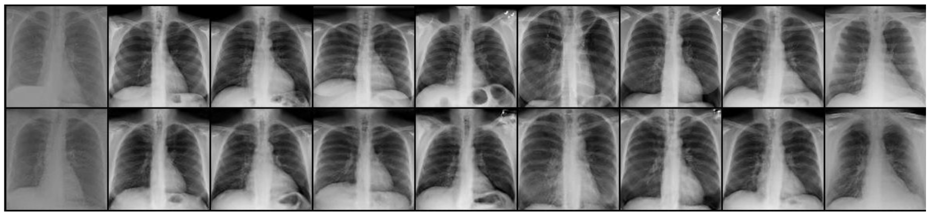


Figure 3.10: **COVID-Net Reconstructions** . First row - real input images. Second row - reconstructed images.

is represented by $6,000$ (600) images for a total of $60,000$ samples. For outliers we used samples from TinyImageNet [158], LSUN [159], and iSUN [160]. For more details please refer to [66]. We reuse the prepared datasets of outliers provided by the ODIN GitHub project page.

3.5.2 Results

MNIST dataset. We perform two experiments that correspond to Protocol 1 and Protocol 2, the results are displayed in tables 3.2, 3.2, and 3.5. We follow the Protocol 1 described in [57, 56] with some differences discussed below. The dataset is randomly divided into three sets: 60% of each class is used for training, 20% for validation, and 20% for testing. The same way of splitting the dataset is also adopted by [154]. Once $p_X(\bar{x})$ is computed for each validation sample, we search for the γ that gives the highest F_1 measure. For each class of digit, we train the proposed model and simulate outliers as randomly sampled images from other categories with proportion equal 50% (Table 3.2) or ranging from 10% to 50% (Table 3.3). Results for $\mathcal{D}(\mathcal{R}(X))$ and $\mathcal{D}(X)$ reported in [57] correspond to the protocol for which data is not split into separate training, validation and testing sets, meaning that the same inliers used for training were also used for testing. We diverge from this protocol and do not reuse the same inliers for training and testing. We follow the $60\%/20\%/20\%$ splits for training, validation and testing. This makes our testing harder, but more realistic, while we still compare our numbers against those obtained by others with easier settings. Results reporting AUC measure on the MNIST dataset using Protocol 1 are shown in Table 3.2, where we compare with [154]. Results reporting F_1 measure on the MNIST dataset using Protocol 1 are shown in Table 3.3, where we compare with [57, 161, 56].

We report results on COVID-Net dataset by training novelty detector on normal only samples and then detecting covid cases. Areas for the ROC curve (AUROC) on COVID-Net dataset for plain AE, OCGAN, and our methods are reported in the Table 3.4. Reconstructions of the dataset samples are shown in Figure 3.10

Table 3.2: AUROC results for novelty detection on MNIST using Protocol 1.

	MNIST	COIL	fMNIST
ALOCC DR [57]	0.88	0.809	0.753
ALOCC D [57]	0.82	0.686	0.601
DCAE [146]	0.899	0.949	0.908
OCGAN [154]	0.977	0.995	0.924
GPND (ours)	**0.984**	**0.9994**	**0.930**

Table 3.3: F_1 scores on MNIST [143]. Inliers are taken to be images of one category, and outliers are randomly chosen from other categories.

% of outliers	$\mathcal{D}(\mathcal{R}(X))$ [57]	$\mathcal{D}(X)$ [57]	LOF [161]	DRAE [56]	GPND AAE (Ours)	GPND w. t.(Ours)	GPND (Ours)
10	0.97	0.93	0.92	0.95	**0.983**	$\underline{0.981}$	$\underline{0.981}$
20	0.92	0.90	0.83	0.91	**0.971**	$\underline{0.970}$	$\underline{0.970}$
30	0.92	0.87	0.72	0.88	$\underline{0.961}$	$\underline{0.961}$	**0.962**
40	0.91	0.84	0.65	0.82	$\underline{0.950}$	**0.953**	**0.953**
50	0.88	0.82	0.55	0.73	**0.939**	**0.946**	$\underline{0.945}$

Table 3.4: AUROC results for novelty detection on COVID dataset.

	AUROC
AE [154]	0.637
OCGAN [154]	0.692
GPND (ours)	**0.835**

Table 3.5: AUROC results for novelty detection on MNIST dataset. Each row represents a different class on which baselines and our model are trained.

	OC SVM	KDE	DAE	VAE	Pix CNN	GAN	AND	OCGAN	GPND(our)
0	0.988	0.885	0.991	0.998	0.531	0.926	0.993	0.998	0.997
1	0.999	0.996	0.999	0.999	0.995	0.995	0.999	0.999	0.999
2	0.902	0.710	0.891	0.962	0.476	0.805	0.959	0.942	0.985
3	0.950	0.693	0.935	0.947	0.517	0.818	0.966	0.963	0.986
4	0.955	0.844	0.921	0.965	0.739	0.823	0.956	0.975	0.977
5	0.968	0.776	0.937	0.963	0.542	0.803	0.964	0.980	0.983
6	0.978	0.861	0.981	0.995	0.592	0.890	0.994	0.991	0.996
7	0.965	0.884	0.964	0.974	0.789	0.898	0.980	0.981	0.984
8	0.853	0.669	0.841	0.905	0.340	0.817	0.953	0.939	0.970
9	0.955	0.825	0.960	0.978	0.662	0.887	0.981	0.981	0.987
avg	0.951	0.814	0.942	0.969	0.618	0.866	0.975	0.975	**0.986**

Table 3.6: AUROC results for novelty detection on CIFAR10 dataset. Each row represents a different class on which baselines and our model are trained.

	OC SVM	KDE	DAE	VAE	Pix CNN	GAN	AND	OCGAN	GPND(our)
0	0.630	0.658	0.718	0.688	0.788	0.708	0.735	0.757	0.767
1	0.440	0.520	0.401	0.403	0.428	0.458	0.580	0.531	0.653
2	0.649	0.657	0.685	0.679	0.617	0.664	0.690	0.640	0.679
3	0.487	0.497	0.556	0.528	0.574	0.510	0.542	0.620	0.545
4	0.735	0.727	0.740	0.748	0.511	0.722	0.761	0.723	0.749
5	0.500	0.496	0.547	0.519	0.571	0.505	0.546	0.620	0.548
6	0.725	0.758	0.642	0.695	0.422	0.707	0.751	0.723	0.740
7	0.533	0.564	0.497	0.500	0.454	0.471	0.535	0.575	0.647
8	0.649	0.680	0.724	0.700	0.715	0.713	0.717	0.820	0.701
9	0.508	0.540	0.389	0.398	0.426	0.458	0.548	0.554	0.670
avg	0.586	0.610	0.590	0.586	0.551	0.592	0.641	0.657	**0.670**

We also follow Protocol 2, described in [70], where official splits for training and testing sets are used. We use 80% of official training split for training, the rest is used for validation. Similarly to Protocol 1, once $p_X(\bar{x})$ is computed for each validation sample, we search for the γ that gives the highest score. For each class of digit, we train the proposed model and simulate outliers as randomly sampled images from other categories with proportion equal 50% and report AUC measure. We report AUC measure separately for each digit and also the average value in Table 3.5 where we compare with [70].

Coil-100 dataset. We follow the protocol described in [138] with some differences dis-

Table 3.7: Results on Coil-100. Inliers are taken to be images of one, four, or seven randomly chosen categories, and outliers are randomly chosen from other categories (at most one from each category).

	OutRank [164]	CoP [165]	REAPER [166]	OutlierPursuit [167]	LRR [168]	DPCP [169]	ℓ_1 thresholding [50]	R-graph [138]	GPND AAE(our)	GPND (our)
Inliers: **one** category of images , Outliers: 50%										
AUC	0.836	0.843	0.900	0.908	0.847	0.900	0.991	0.997	0.968	0.9994
F1	0.862	0.866	0.892	0.902	0.872	0.882	0.978	**0.990**	0.979	0.982
Inliers: **four** category of images , Outliers: 25%										
AUC	0.613	0.628	0.877	0.837	0.687	0.859	0.992	**0.996**	0.945	0.983
F1	0.491	0.500	0.703	0.686	0.541	0.684	0.941	**0.970**	0.960	**0.970**
Inliers: **seven** category of images , Outliers: 15%										
AUC	0.570	0.580	0.824	0.822	0.628	0.804	0.991	**0.996**	0.919	0.973
F1	0.342	0.346	0.541	0.528	0.366	0.511	0.897	0.955	0.941	**0.959**

Table 3.8: Results on Fashion-MNIST [157]. Inliers are taken to be images of one category, and outliers are randomly chosen from other categories.

$\%$ of outliers	10	20	30	40	50
GPND AAE					
F1	0.968	0.945	0.917	0.891	0.864
AUC	0.928	0.932	0.933	0.933	0.933
GPND					
F1	0.971	0.948	0.926	0.903	0.877
AUC	0.930	0.930	0.930	0.930	0.930

cussed below. Results are averages from 5-fold cross-validation. Each fold takes 20% of each class. Because the count of samples per category is very small, we use 80% of each class for training, and 20% for testing. We find the optimal threshold γ on the training set. Results reported in [138] correspond to not splitting data into separate training, validation and testing sets, because it is not essential, since they leverage a VGG [162] network pre-trained on ImageNet [163]. We diverge from that protocol and do not reuse inliers and follow $80\%/20\%$ splits for training and testing.

Results on Coil-100 are shown in Table 3.7. We do not outperform R-graph [138], however as mentioned before, R-graph uses a pretrained VGG network, while we train an autoencoder from scratch on a very limited number of samples, which is on average only 70 per category.

Fashion-MNIST dataset. We repeat the same experiment with the same protocol that we

Table 3.9: Comparison with ODIN [66]. ↑ indicates larger value is better, and ↓ indicates lower value is better.

	Outlier dataset	**FPR(95%TPR)↓**	**Detection↓**	**AUROC↑**	**AUPR in↑**	**AUPR out↑**
		ODIN-WRN-28-10 / ODIN-Dense-BC / GPND				
CIFAR-10	TinyImageNet (crop)	23.4/**4.3**/29.1	14.2/**4.7**/15.7	94.2/**99.1**/90.1	92.8/**99.1**/84.1	94.7/99.1/**99.5**
	TinyImageNet (resize)	25.5/**7.5**/11.8	15.2/**6.3**/8.3	92.1/**98.5**/96.5	89.0/**98.6**/95.0	93.6/98.5/**99.8**
	LSUN (resize)	17.6/**3.8**/4.9	11.3/**4.4**/4.9	95.4/**99.2**/98.7	93.8/**99.3**/98.4	96.1/99.2/**99.7**
	iSUN	21.3/**6.3**/11.0	13.2/**5.7**/7.8	93.7/**98.8**/96.9	91.2/**98.9**/96.1	94.9/98.8/**99.7**
	Uniform	**0.0/0.0/0.0**	2.5/2.5/**0.1**	**100.0**/99.9/99.9	**100.0/100.0/100.0**	**100.0**/99.9/99.5
	Gaussian	**0.0/0.0/0.0**	2.5/2.5/**0.0**	**100.0/100.0/100.0**	**100.0/100.0/100.0**	**100.0/100.0**/99.8
CIFAR-100	TinyImageNet (crop)	43.9/**17.3**/33.2	24.4/**11.2**/17.2	90.8/**97.1**/89.1	91.4/**97.4**/83.8	90.0/96.8/**98.7**
	TinyImageNet (resize)	55.9/44.3/**15.0**	30.4/24.6/**9.5**	84.0/90.7/**95.9**	82.8/91.4/**94.6**	84.4/90.1/**99.4**
	LSUN (resize)	56.5/44.0/**6.8**	30.8/24.5/**5.8**	86.0/91.5/**98.3**	86.2/92.4/**98.0**	84.9/90.6/**99.6**
	iSUN	57.3/49.5/**14.3**	31.1/27.2/**9.3**	85.6/90.1/**96.2**	85.9/91.1/**95.6**	84.8/88.9/**99.3**
	Uniform	0.1/0.5/**0.0**	2.5/2.8/**0.0**	99.1/99.5/**100.0**	99.4/99.6/**100.0**	97.5/99.0/**99.7**
	Gaussian	1.0/0.2/**0.0**	3.0/2.6/**0.0**	98.5/99.6/**100.0**	99.1/99.7/**100.0**	95.9/99.1/**100.0**

Table 3.10: **F1 and AUROC results on MNIST using Protocol 1.** Ablation study on MNIST dataset using Protocol 1 under the following conditions: a) *GPND* – the proposed approach, uses both parallel and perpendicular components, b) *GPND+J* – parallel and perpendicular components, where parallel part is includes Jacobian computation, c) *GPND-parallel* – parallel component only, d) *GPND-perpendicular* – perpendicular component only.

	F1	AUROC
GPND	0.945	0.985
GPND+J	0.954	0.987
GPND-parallel	0.901	0.961
GPND-perpendicular	0.925	0.969

have used for MNIST, but on Fashion-MNIST. Results are provided in Table 3.8.

CIFAR-10 (CIFAR-100) dataset. We follow the protocol described in [66], where for inliers and outliers are used different datasets. ODIN relies on a pretrained classifier and thus requires label information provided with the training samples, while our approach does not use label information. The results are reported in Table 3.9. Despite the fact that ODIN relies upon powerful classifier networks such as Dense-BC and WRN with more than 100 layers, the much smaller network of GPND competes well with ODIN. Note that for CIFAR-100, GPND significantly outperforms both ODIN architectures. We think this might be due to the fact that ODIN relies on the perturbation of the network classifier output, which becomes less accurate as the number of classes grows from 10 to 100. On the other hand, GPND does not use class label information and copes much better with the additional complexity induced by the increased number of classes.

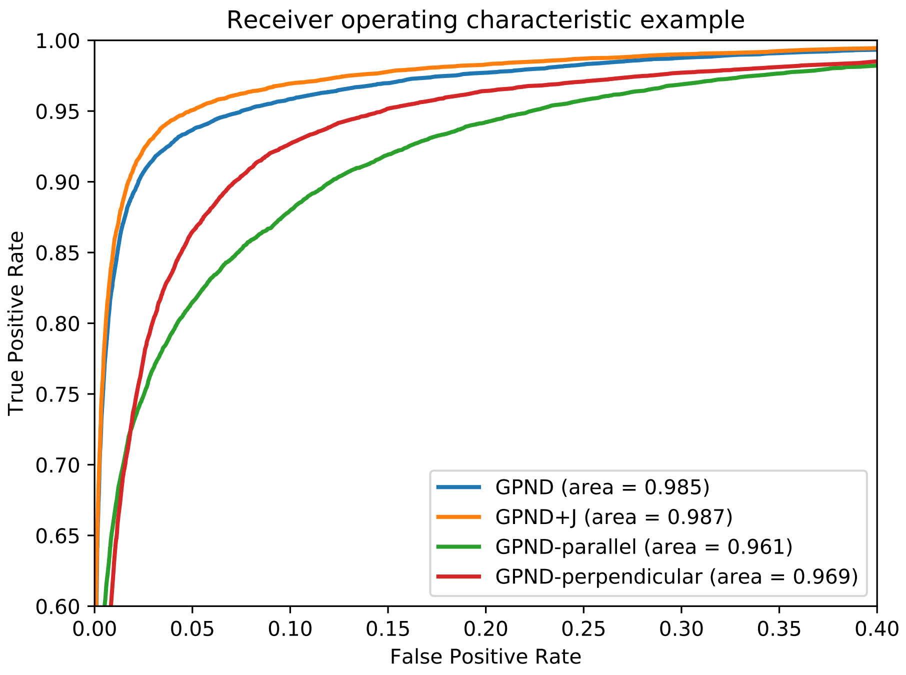


Figure 3.11: **ROC curves**. Ablation study on MNIST dataset using Protocol 1 under the following conditions: a) *GPND* – the proposed approach, uses both parallel and perpendicular components, b) *GPND+J* – parallel and perpendicular components, where parallel part is includes Jacobian computation, c) *GPND-parallel* – parallel component only, d) *GPND-perpendicular* – perpendicular component only.

Table 3.11: **F1 and AUROC results on MNIST using Protocol 1.**. Comparison with baselines.

	10%	**20%**	**30%**	**40%**	**50%**
			F1		
GPND	0.981	0.970	0.962	0.953	0.945
AE	0.848	0.796	0.795	0.776	0.756
P-VAE	0.976	0.958	0.942	0.924	0.905
P-AAE	0.973	0.955	0.940	0.920	0.902
			AUROC		
GPND	0.984	0.984	0.984	0.984	0.984
AE	0.934	0.938	0.934	0.929	0.928
P-VAE	0.952	0.957	0.956	0.958	0.959
P-AAE	0.952	0.956	0.953	0.952	0.953

In the ablation study we motivate the importance of both, parallel and perpendicular components of $p_X(\bar{x})$ in (3.5) and also demonstrate the effect of the approximation in 3.31.

To demonstrate the importance of parallel and perpendicular components, we repeat the experiment with MNIST using Protocol 1 (as in Table 3.2) under the following conditions: a) *GPND* is the proposed approach where $p_X(\bar{x})$ is computed as in 3.5 and uses the approximation 3.31 (jacobian is not computed) and implemented as in 3.33, b) *GPND+J* is the proposed approach as in 3.5, but without approximation 3.31, e.i. jacobian is computed, c) *GPND-parallel* – parallel component only, drops $p_{W^\perp}$ and assumes $p_X(\bar{x}) = p_{W^\parallel}(\bar{w}^\parallel)$; d) *GPND-perpendicular* – perpendicular component only, drops $p_{W^\parallel}$ and assumes $p_X(\bar{x}) = p_{W^\perp}(\bar{w}^\perp)$. For this ablation study we report F1 and AUROC measures, see Table 3.10 and we show the ROC curves see Figure 3.11. The ablation study demonstrates that both, parallel and perpendicular components of $p_X(\bar{x})$ in (3.5) significantly contribute to the overall performance of the proposed approach. The Jacobian component also contributes to performance improvement, however the computational cost outweight its role thus justifing the approximation in 3.31.

Next, we compare GPND with some baselines to better appreciate the improvement provided by the architectural choices, see Table 3.11.

The baselines are: i) vanilla AE with thresholding of the reconstruction error and same pipeline (AE); ii) proposed approach where the AAE is replaced by a VAE (P-VAE); iii) proposed approach where the AAE is without the additional adversarial component induced by the discriminator applied to the decoded image (P-AAE).

Additional implementation details include the choice of hyperparameters. For MNIST and COIL-100 the latent space size was chosen to maximize F_1 on the validation set. It is 32, and we varied it from 16 to 64 without significant performance change. For CIFAR-10 and CIFAR-100, the latent space size was set to 64. We use the Adam optimizer with betas equal to $\beta_1 = 0$ and $\beta_2 = 0.99$, learning rate of 0.002, and learning rate equalization [170]. We use batch size of 32, and train the network for 160 epochs. On MNIST, CIFAR-10 and

CIFAR-100 it takes approximately 3 hours to train the network on Titan X GPU. It takes approximately 10 hours to train on COVID-Net dataset, longer time is due to the larger size of the images.

3.6 Conclusion

We introduced GPND, an approach and a network architecture for novelty detection that is based on learning mappings f and g that define the parameterized manifold $\mathcal{M}$ which captures the underlying structure of the inlier distribution. Unlike prior deep learning based methods, GPND detects that a given sample is an outlier by evaluating its inlier probability distribution. We have shown how each architectural and model components are essential to the novelty detection. In addition, with a relatively simple architecture we have shown how GPND provides state-of-the-art performance using different measures, different datasets, and different protocols, demonstrating to compare favorably also with the out-of-distribution literature.

Chapter 4

Deep Supervised Hashing with Spherical Embedding

4.1 Introduction

Indexing and searching large-scale image databases leverage heavily hashing based approximate nearest neighbor search technology. The goal of hashing is to map high dimensional data, such as images, into compact codes in a way that visually, or semantically similar images are mapped into similar codes, according to the Hamming distance. Given a query image, hierarchically structured based methods can then be used for the rapid retrieval of the neighbors within a certain distance from the query [171].

Recent data-dependent methods for hashing images (as opposed to data-independent methods [172]) leverage deep neural networks due to their ability to integrate the image feature representation learning with the objective of the hashing task [173, 89, 88, 94, 100], leading to a superior efficiency and compactness of the codes. However, so far the focus of these approaches has been on designing architectures only with the similarity preserving goal of mapping similar images to similar codes. On the other hand, hashing methods based on hand-crafted image features [174], have improved performance also by requiring codes to have certain properties, for example, to be balanced, uncorrelated, or to be obtained with a small quantization error [73, 175, 176, 74, 177, 82, 178].

Balanced codes are such that bits partition data in equal portions [73, 175]. It is a desirable property because it increases the variance of bits since they approach 50% chance of being one or zero. Also code bits that are uncorrelated, or even better independent, increase their information content. In addition, learning hash functions generally require solving intractable optimization problems that are made tractable with the continuous relaxation of the codomain of the function. This means that quantization is required to discretize the continuous embedding into codes, and controlling the quantization error has been shown to improve performance [74].

In this work, we propose a deep supervised hashing approach that goes beyond similarity preserving, and that aims at learning hashing functions that map onto codes with good quality, by encouraging them to be *balanced*, and to *maximize the mean average precision*. We do so by advocating the use of a different continuous relaxation strategy that has several advantages, incto overcome the challenge of maintaining the learning problem well-posed, without the addition of costly priors that typically encourage a binary output, and that have led deep hashing approaches to focus, so far, only on similarity preserving codes. This approach allows learning a hashing function composed of a *spherical embedding* followed by an optimal *quantization*. Specifically, the embedding is based on a convolutional neural network, learned with a triplet loss framework [179], that has the advantage of being more general, because it allows the use of different triplet losses, and it allows introducing a new *triplet spring loss* that aims at learning balanced codes. Moreover, the loss framework is rotation invariant in the embedded space, which allows optimizing the quantization for a rotation that provides the highest mean average precision. We call the resulting model *Spherical Deep Supervised Hashing (SDSH)*, and provides state-of-the-art performance on standard benchmarks, including a significantly greater performance at more compact code dimensions.

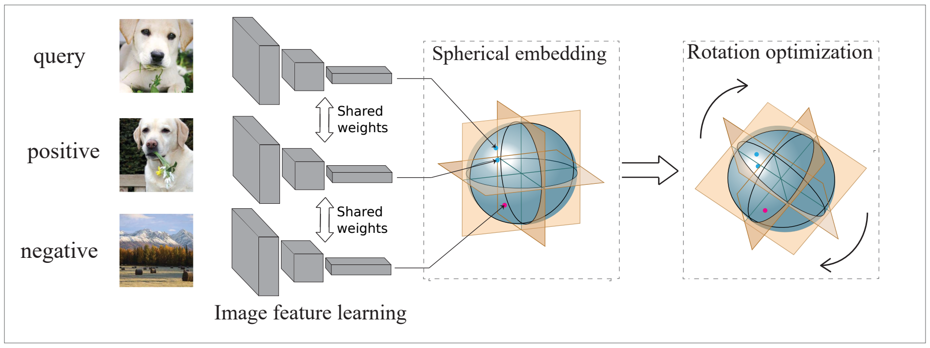


Figure 4.1: **Overview.** Overview of the approach, highlighting the stages of the hash embedding learning, and the optimal quantization.

4.2 Problem Overview

Given a training set of N images $\mathcal{I} = \{I_1, \cdots, I_N\}$, with labels $\mathcal{Y} = \{y_1, \cdots, y_N\}$, we are interested in learning a *hash function* h that maps an image I onto a compact binary code $\mathbf{b} = h(I) \in \{+1, -1\}^B$ of length B. The typical approach based on deep learning assumes that given three images I_i, I_j, I_k, with labels y_i, y_j, y_k, such that $y_i = y_j$, and $y_i \neq y_k$, then the hash function should be such that the corresponding binary codes $\mathbf{b}_i$ and $\mathbf{b}_j$ should be close, while $\mathbf{b}_i$ and $\mathbf{b}_k$ should be far away in the Hamming space. If $d_H(\cdot, \cdot)$ indicates the Hamming distance, this means that $d_H(\mathbf{b}_i, \mathbf{b}_j)$ should be as small as possible, while $d_H(\mathbf{b}_i, \mathbf{b}_k)$ should be as large as possible.

In addition to that, ideally, we would want to encourage hash codes to be maximally informative, where bits are independent and balanced, and to ultimately maximize the mean average precision (mAP). These aspects have been of secondary importance thus far, because deep learning approaches have to use binarizing priors to regularize loss functions that are already computationally intensive to optimize.

We overcome the major hurdle of the binarizing prior by advocating the use of a different relaxation method, which does not require additional priors, and learns a *spherical embedding*. This modeling choice has ripple effects. Besides simplifying the learning by eliminating the binarizing prior, it enables a unified formulation of a class of triplet losses [179] that are *rotation invariant*, and it allows to introduce one, which we name

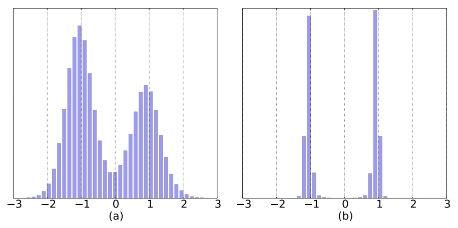


Figure 4.2: **Embedding distribution.** Bimodal distribution of the hash embedding components (a) early on during training, and (b) at advanced training stage, when modes are separated.

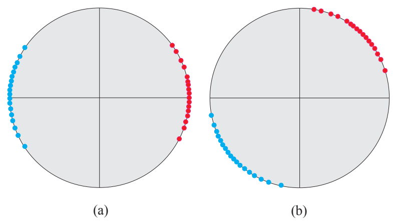


Figure 4.3: **Quantization.** (a) Distribution of hash embeddings on the unit circle for two classes. The sign quantization assigns different hash codes to samples in the same class. (b) Rotated distribution of hash embeddings. The sign quantization assigns same hash codes to samples in the same class, thus increasing the mAP.

spring loss, that encourages balanced hash codes. In addition, the rotation invariance allows us to look for a rotation of the embedding hypersphere that leads to its optimal *quantization* for producing hash codes that directly maximize the retrieval mAP. This two-stage approach is depicted in Figure 4.1.

4.3 Hash Function Learning

The desired hash function should ensure that the code $\mathbf{b}_i$ of image I_i would be closer to all other codes $\mathbf{b}_j$ of I_j because $y_i = y_j$, than it would be to any code $\mathbf{b}_k$ of I_k since $y_i \neq y_k$. Therefore, if $\mathcal{T} = \{(i,j,k) | y_i = y_j \neq y_k\}$ is the set of the allowed triplet labels, then we certainly desire this condition to be verified

$$d_H(\mathbf{b}_i, \mathbf{b}_j) < d_H(\mathbf{b}_i, \mathbf{b}_k) \quad \forall (i,j,k) \in \mathcal{T} \; . \tag{4.1}$$

A loss function that aims at achieving condition (4.1) could simply be written as

$$L(h) = \sum_{(i,j,k)\in\mathcal{T}} \ell(d_H(\mathbf{b}_i, \mathbf{b}_j) - d_H(\mathbf{b}_i, \mathbf{b}_k)) \tag{4.2}$$

where $\ell(\cdot) : \mathbb{R} \to [0, +\infty)$ is the cost for a triplet that does not satisfy (4.1). Equation (4.2) is a more general version of the well known triplet loss [179].

As pointed out in [94], it is easy to realize that

$$d_H(\mathbf{b}_i, \mathbf{b}_j) - d_H(\mathbf{b}_i, \mathbf{b}_k) = \frac{1}{2}\mathbf{b}_i^\top \mathbf{b}_k - \frac{1}{2}\mathbf{b}_i^\top \mathbf{b}_j \ . \tag{4.3}$$

In particular, the Hamming space where codes are defined, and through which depends the estimation of the hash function h, makes the optimization of (4.2) intractable [93]. Therefore, the typical approach is to relax the domain of $\mathbf{b}$ from the Hamming space to the continuous space $\mathbb{R}^B$ [93, 94]. However, this method has severe drawbacks. The first and most important one is that optimizing (4.2) becomes an ill-posed problem, with trivial solutions corresponding to pulling infinitely apart relaxed codes with label mismatch. This forces the introduction of a regularizing prior to the loss, which typically is designed to encourage the relaxed code $\tilde{\mathbf{b}}$ to be also "as binary as possible", or in other words, to stay close to one of the vertices of the Hamming space.

Computationally, adding a prior is a major setback because it increases the number of hyperparameters at best, with all the consequences. In addition, if we look at the distribution of the values of the components of $\tilde{\mathbf{b}}$, we have observed experimentally that as the two main modes around $+1$ and -1 become separated, the corresponding hash codes, obtained simply by taking $\mathbf{b} = \mathrm{sgn}(\tilde{\mathbf{b}})$, stop changing during the training procedure. See Figure 4.2. This "locking" behavior might prevent from learning a hash function that could potentially be more efficient if it still had room to adjust the outputs. Finally, we note that (4.3) does not hold in the relaxed space, meaning that

$$\left\| \frac{\tilde{\mathbf{b}}_i - \tilde{\mathbf{b}}_j}{2} \right\|^2 - \left\| \frac{\tilde{\mathbf{b}}_i - \tilde{\mathbf{b}}_k}{2} \right\|^2 \neq \frac{1}{2}\tilde{\mathbf{b}}_i^\top \tilde{\mathbf{b}}_k - \frac{1}{2}\tilde{\mathbf{b}}_i^\top \tilde{\mathbf{b}}_j \ . \tag{4.4}$$

even though there are approaches that rely on the left-hand-side of (4.4) being approximately equal to the right-hand-side [94].

The following section addresses the drawbacks outlined above by advocating the use of a different relaxation for hash function learning.

4.4 Spherical Embedding

The hash function learning problem can be summarized as learning a function $\tilde{h}$ such that $h(I) = \text{sgn}[\tilde{h}(I)]$, where $\tilde{h}$ optimizes a relaxed version of (4.2). Differently from previous hashing work, we propose to use a relaxation where $\tilde{h}$ is a *spherical embedding*, meaning that we constrain the output $\mathbf{s} = \tilde{h}(I)$ to be defined on the $(B-1)$-dimensional unit sphere. This means that, using the previous notation, $\mathbf{s} \doteq \tilde{\mathbf{b}}/\|\tilde{\mathbf{b}}\|$, and the meaning of Equations (4.1), (4.2), and (4.3) remain valid by simply substituting $\mathbf{b}$ with $\mathbf{s}$, and $d_H(\mathbf{b_i}, \mathbf{b_j})$ with $\|(\mathbf{s}_i - \mathbf{s}_j)/2\|^2$. Therefore, we advocate the end-to-end learning of a function $\tilde{h}$, given by minimizing the loss

$$L(\tilde{h}) = \sum_{(i,j,k)\in\mathcal{T}} \ell(\mathbf{s}_i^\top \mathbf{s}_k - \mathbf{s}_i^\top \mathbf{s}_j) \ . \tag{4.5}$$

This approach, also used in [179] to regularize the spreading of the embedding, is leveraged here to address the limitations of the continuous relaxation described in Section 4.3. Indeed, a spherical embedding makes the optimization of the relaxed version of (4.2) (which is (4.5)) a well-posed problem, and this was the main reason why previous works required a regularizing prior. In addition, previous priors encouraged the embedding space to be "as binary as possible" by moving $\tilde{\mathbf{b}}$ closer to a vertex of the Hamming cube, without direct evidence that this was producing better hash codes. On the other hand, we observed this practice to encourage a "locking" behavior, wich we do not have with the spherical embedding because there are no forces pushing towards the Hamming cube, and $\mathbf{s}$ is free to move on the unit sphere. Moreover, if $\tilde{h}$ is an optimal spherical embedding, so is $R\tilde{h}$, where R is a rotation matrix, since it still minimizes (4.5). Therefore, since (4.5) is *rotation invariant*, $\tilde{h}$ is found modulo a rotation, which can be estimated at a later stage to optimize other hash code properties (Section 4.5).

In Section 4.6 we design different triplet loss functions $\ell(\cdot)$, leading to different spherical embeddings. In practice, the spherical embedding comprises a convolutional neural network with a number of layers aiming at learning visual features from images, followed by fully connected layers with an output dimension equal to the number of bits B of the hash code. We adopt the VGG-F architecture [180], and replace the last fully connected layer, but other architectures can also be used [84, 181].

4.5 Quantization

Given an image I, its hash code is computed with a *sign quantization* as $\mathbf{b} = \text{sgn}(\tilde{h}(I))$. Since $\tilde{h}$ minimizes (4.5), we observed that also a rotated version $R\tilde{h}$ does, so it is important to analyze the difference between the two solutions. Figure 4.3(a) shows a case where the spherical embeddings of two classes along the unit circle are such that the sign quantization assigns different hash codes to samples of the same class. Therefore, a rotation R could be applied to the embeddings as in Figure 4.3(b), where the sign quantization would now produce the expected results.

We propose to use the extra degrees of freedom due to the *rotation invariance* of (4.5) for learning a rotation matrix R that finds the quantization that produces the best hash function. We do so by estimating the rotation R that maximizes the mean average precision (mAP), which is the metric that we value the most for retrieval

$$\hat{R} = \arg\max_{R} \text{mAP}(R) \ . \tag{4.6}$$

Since mAP$(\cdot)$ is not a smooth function, and has zero gradient almost everywhere, (4.6) is not easy to optimize, even with derivative-free methods. On the other hand, we found that a standard random search optimization with a linear annealing schedule allows to achieve good results quickly. At the i-th iteration we apply a random perturbation $Q^{(i)}$ to the current rotation matrix $R^{(i)}$ to obtain the update $R^{(i+1)} = Q^{(i)}R^{(i)}$, which we retain if it improves the mAP. Since the perturbation should be random, uniform, and with a controllable magnitude, we generate it by setting $Q^{(i)} = P^{(i)}E(\theta)P^{(i)\top}$, where $P^{(i)}$ is a random unitary matrix, generated with a simplified approach based on the SVD decomposition of a matrix with normally sampled elements [182, 183]. $E(\theta)$ instead represents a rotation by θ on the plane identified by the first two basis vectors. The angle θ defines the perturbation magnitude and varies linearly with the iteration number, starting with $\theta_0 = 1.0$ down to 0 when the maximum number of iterations is reached, which was 800 in our experiments. Algorithm 2 summarizes the steps. We compute the mAP with a C++ implementation, where we take 1000 samples from the training set as queries and 16000 samples as database, or a smaller number if the training set is smaller. With a PC workstation with CPU Core i7 5820K 3.30GHz the running time for one iteration update is around 0.4s, keeping the time for the random search optimization very small, if compared with the time for training the deep network of the spherical embedding.

We now note that if R is an optimal solution, by swapping two columns of R we obtain a new solution, corresponding to swapping two bits in all hash codes. Since there are $B!$ of these kind of changes, it means that the order of growth of the solution space is $O(B!)$. Therefore, as B increases, estimating R according to (4.6) becomes less important because the likelihood that a random R is not far from an optimal solution has increased accordingly. The experimental section supports this observation.

4.6 Triplet Spherical Loss

In this section we give three examples of *rotation invariant* triplet loss ℓ that can be used in (4.5), namely the *margin loss*, the *label likelihood loss*, and the new *spring loss*. To shorten the notation, we define the quantity $d_{i,j,k} \doteq \mathbf{s}_i^\top \mathbf{s}_k - \mathbf{s}_i^\top \mathbf{s}_j$.

4.6.1 Margin Loss

The first loss that we consider is well known, and stems from requiring condition (4.1) to be verified with a certain margin α, in combination with using the standard hinge loss. This

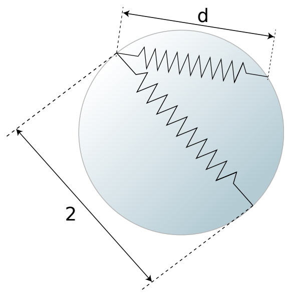


Figure 4.4: **Spring loss.** Unit sphere where two points with different class labels are pulled apart by an elastic force proportional to the displacement $2 - d$, while constrained to remain on the sphere.

translates into the following *margin loss*

$$\ell(d_{i,j,k}) = \max\{0, d_{i,j,k} + \alpha\} \ . \tag{4.7}$$

4.6.2 Label Likelihood Loss

The second loss has been originally proposed in [94], where it was used with the Hamming space relaxed into the continuous space $\mathbb{R}^B$ for learning a hashing function. Here we extend it for learning a spherical embedding. The loss is derived from a probabilistic formulation of the likelihood of the triplet labels $\mathcal{T}$, where triplets that verify condition (4.1) by bigger margins have a bigger likelihood, and where the margin parameter α can affect the speed of the training process. This *label likelihood loss*, adapted to our framework becomes

$$\ell(d_{i,j,k}) = d_{i,j,k} + \alpha + \log(1 + e^{-d_{i,j,k} - \alpha}) \ . \tag{4.8}$$

4.6.3 Spring Loss

While both the margin loss and the label likelihood loss produce remarkable results, none of them make explicit efforts towards clustering samples in the spherical embedding space, according to their classes, and in a way that classes cover the sphere in a spatially uniform manner. This last property is very important because if we assume an equal number of samples per class, for each bit b, balanced codes satisfy the property

$$\sum_{i=1}^{N} h_b(I_i) = 0, \quad b = 1, \cdots, B \; . \tag{4.9}$$

Therefore, we note that condition (4.9) is satisfied whenever the spherical embedding distributes the classes uniformly on the unit sphere, thus producing balanced codes, where code bits have higher variance and are more informative. This has motivated the design of the loss that we introduce.

Algorithm 2 Random search for the optimal rotation R.

```
Result: Returns the optimal matrix $R$ according to a random search
 1: $R^{(0)} = I$                          ▷ Initialize rotation matrix with identity matrix
 2: $i = 0$
 3: while $i <$ number of iterations do
 4:     $\theta^{(i)} \leftarrow f(i)$;      ▷ $f(i)$ is a linear annealing schedule to update the rotation magnitude
 5:                                        ▷ $\theta^{(i)}$ is the magnitude of perturbational rotation for iteration $i$
 6:     $E^{(i)} \leftarrow E(\theta^{(i)})$;   ▷ $E(\theta^{(i)})$ Returns rotation around some fixed axis, with magnitude $\theta^{(i)}$
 7:     $P^{(i)} \leftarrow$ random unitary matrix
 8:     $Q^{(i)} \leftarrow P^{(i)} E^{(i)} P^{(i)\top}$   ▷ $Q^{(i)}$ is a random perturbational rotation matrix with magnitude $\theta^{(i)}$
 9:     $R' \leftarrow Q^{(i)} R^{(i)}$           ▷ Update rotation matrix $R$ with perturbation
10:     if mAP$(R')$ >mAP$(R^{(i)})$ then       ▷ If updated version of $R$ is better, then keep it
11:         $R^{(i+1)} \leftarrow R'$
12:     else
13:         $R^{(i+1)} \leftarrow R^{(i)}$
14:     end if
15:     $i \leftarrow i + 1$
16: end while
17: return $R^i$
```

Let us consider two points $\mathbf{s}_i$ and $\mathbf{s}_k$ on the $(B-1)$-dimensional sphere of unit radius, and let us assume that a spring is connecting them. The Euclidean distance $d = \|\mathbf{s}_i - \mathbf{s}_k\| \doteq \sqrt{d_{i,k}}$, between the points varies in the range $[0, 2]$. At distance 2, we consider the spring unstretched, while at distance $d < 2$, the spring will have accumulated an elastic potential energy proportional to $(2-d)^2$. See Figure 4.4. This suggests that we could train the hash embedding by minimizing (4.5), with $\ell(d_{i,k}) = (2 - \sqrt{d_{i,k}})^2$, where we would limit the summation to the pairs $(i,k) \in \mathcal{Q} = \{(i,j) | y_i \neq y_j\}$. In this way, the training would aim at minimizing the total elastic potential energy of the system of springs. The effect is that samples from different classes would be mapped on the sphere, but as far apart as

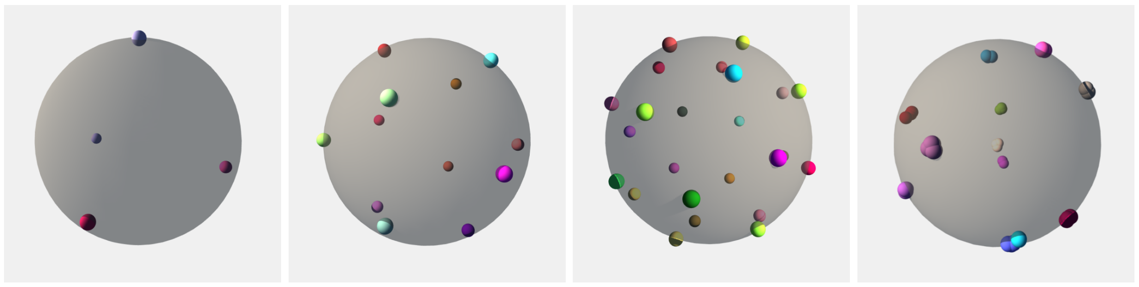


Figure 4.5: **Regular polyhedrons.** Three left-most images: Three unit spheres with $5 \times n$ points at minimum elastic potential, where n is the number of classes. From left to right n is equal to 4, 12, 24. The 5 points per class coincide with the class centroid at equilibrium. Right-most image: For $n = 12$, the points at minimum margin loss do not reach a uniform distribution on the sphere.

possible. We note that minimizing this loss is equivalent to solving a first order linear approximation to the Thomson's Problem [184], which concerns the determination of the minimum electrostatic potential energy configuration of N electrons, constrained to the surface of a unit sphere. The solution has been rigorously identified in only a handful of cases, which mostly correspond to spatial configurations forming regular polyhedrons, which we have observed also in our simulations using the spring loss described before, as it can be seen in Figure 4.5. If we were to perform a Voronoi tessellation on the sphere based on the class centroids, we clearly would obtain a pretty uniform partition of the sphere, which is our main goal. For comparison, the right-most image in Figure 4.5 is obtained with the margin loss, which stops pulling apart query and negative samples once they are more far apart than the margin. This leads to a less uniform distribution of the classes.

We have experimented with the loss described above and indeed, it provides fairly good results. However, we argue that it can be improved because it does not explicitly pull closer samples that belong to the same class, besides pulling apart samples with different labels. We address that issue with the *triplet spring loss*, which we define as follows

$$\ell(d_{i,j,k}) = (2 - \sqrt{2 - d_{i,j,k}})^2 \ . \tag{4.10}$$

Note that $d_{i,j,k}$ varies in the range $[-2, 2]$, thus the square root varies in the range $[0, 2]$, and the loss varies in the range $[0, 4]$. The loss is minimized when $d_{i,j,k}$ approaches -2. Since

$d_{i,j,k}$ is proportional to $\|\mathrm{s}_i - \mathrm{s}_j\|^2 - \|\mathrm{s}_i - \mathrm{s}_k\|^2$, convergence is approached by maximally pulling closer s_i and s_j, while maximally pushing apart s_i and s_k. Note that, differently than before, now even when s_i and s_k have reached a distance of 2, the loss still works to pull s_i and s_j closer.

4.7 Experiments

We tested our approach on the most relevant datasets for deep hashing applications, namely *CIFAR-10* [156], *NUS_WIDE* [185], and we also tested on *MNIST* [124] to compare with some older methods. For each experimental setting, we report the average mAP score over 5 runs for comparison against the previous works for hashes of size as low as 4 and up to 48 bits.

4.7.1 Experimental setup

Similar to other deep hashing methods we use raw image pixels as input. Following [94, 186, 93], for the spherical embedding we adopt VGG-F [187] pre-trained on ImageNet. We replace the last layer with our own, initialized with normal distribution. The output layer doesn't have activation function and the number of outputs matches the needed number of bits - B. The input layer of VGG-F is 224x224, so we crop and resize images of the *NUS_WIDE* dataset and upsample images of the *CIFAR-10* dataset to match the input size.

CIFAR-10: CIFAR-10 [156] is a publicly available dataset of small images of size 32x32 which have each been labeled to one of ten classes. Each class is represented by 6,000 images for a total of 60,000 available samples. In terms of evaluation in the CIFAR domain, two images are counted as relevant to each other if their labels match. In order for our experiments to be comparable to as many works as possible, including [94] and [100], we use two different experimental settings, which are labeled "Full" and "Reduced".

"Full" setting: For this setting, 1,000 images are first selected randomly from each class

of the dataset to make up the test images. Which, by extension, results in 10,000 query images. The remaining 50,000 images are used as the database images and as the images used in training.

"Reduced" setting: For this setting, 100 images are selected randomly from each class for use as 1,000 total test images. From the remaining 59,000 samples we randomly sample 500 images per category to form the reduced training set with only 5,000 images. The database is composed of all 59,000 samples which were not selected for testing.

The mAP for CIFAR-10, full and reduced setting, is computed based on all samples from the database set. We have used for training purposes a system with only one NVIDIA Titan X GPU, in this configuration training takes about four hours for the Cifar Full setting.

NUS_WIDE: NUS_WIDE [185] is another publicly available dataset, but unlike CIFAR-10 each sample image is multi-labeled from a set of 81 possible labels across all 269,643. This is reduced slightly, as [94] and [100] have also done in their experiments, by first removing every image which does not have any of the 21 most common labels associated with it. This is done as many of the less common labels have very few samples associated with them, but prepared in this way each of the 21 labels are represented by at least 5,000 samples. In terms of evaluation in the NUS_WIDE domain two images are counted as relevant to each other if any of their labels match. Note, that despite the usage of samples associated with the 21 most frequent labels, all 81 labels are used for determining similarity between samples. To compare with previous work we use three different settings, labeled "Full", "Reduced A", and "Reduced B".

"Full" setting: For this setting, 100 samples from each of the 21 most frequent labels are reserved for the test set. And the remaining images are used both as the database and as the training set. The mAP is computed based on the top 50000 returned neighbors.

"Reduced A" setting: For this setting, the 2100 test samples are selected as in the Full setting. From the remaining samples, 500 were sampled from the 21 most frequent labels to compose the training set. The remaining were used for the database. The mAP is computed

Table 4.1: Mean Average Precision (MAP) Results for Different Number of Bits of CIFAR-10: In the case of DTSH and DVSQ we have filled some additional results which were not presented by the original papers by using the authors' respective released source code to replicate their experiments such that we may compare with them across more hash sizes.

	CIFAR-10 Full setting: Number of Bits							CIFAR-10 Reduced setting: Number of Bits						
Method	4	8	12	16	24	32	48	4	8	12	16	24	32	48
DQN[96]	-	-	-	-	-	-	-	-	-	0.554	-	0.558	0.564	0.580
DSH[91]	-	-	0.616	-	0.651	0.661	0.676	-	-	-	-	-	-	-
DPSH[93]	-	-	0.763	-	0.781	0.795	0.807	-	-	0.763	-	0.727	0.744	0.757
DMDH [95]	-	-	-	-	-	-	-	-	-	-	0.704	-	0.719	0.732
DTSH[94]	-	0.814	0.859	0.915	0.923	0.925	0.926	-	0.641	0.710	0.723	0.750	0.765	0.774
DVSQ[100]	-	0.839	-	0.839	0.843	0.840	0.842	-	**0.715**	-	0.727	0.730	0.733	0.764
BL[188]	0.870							-						
SDSH-ML	-	0.839	0.882	0.886	0.939	0.880	0.878	-	0.657	0.712	0.756	0.747	0.765	0.764
SDSH-LL	0.481	0.763	0.854	**0.942**	**0.945**	**0.944**	**0.947**	0.407	0.673	**0.757**	0.782	0.799	**0.815**	**0.822**
SDSH-S	**0.755**	**0.911**	**0.939**	0.938	0.939	0.939	0.934	**0.569**	0.697	0.723	**0.783**	**0.801**	0.810	0.813

based on top 5000 returned neighbors.

"Reduced B" setting: For this setting, the training set was chosen by sampling the available images uniformly 5,000 times. From the remaining samples the training set was uniformly sampled 10,000 times. All of the remaining samples from these two operations were used as the database. The mAP is computed based on top 5000 returned neighbors.

4.7.2 Results

Our method is abbreviated as SDSH, and the combination with the margin loss, label likelihood loss, and spring loss are indicated as SDSH-ML, SDSH-LL, and SDSH-S respectively. All reported results are from our publicly available implementation[1]. The average mAP scores for all methods on CIFAR-10 are listed in Table 4.1, which includes also the baseline (BL) classification accuracy. As pointed out in [188], the BL value can be interpreted as mAP attainable by a supervised system retrieving samples, one class at a time, with classes ranked according to the class probability of the query. Therefore, a retrieval approach should surpass the BL threshold to be effective. Table 4.1 shows that the proposed SDSH-S is above BL starting from 8-bits, and it always outperforms the state-of-the-art methods. In full setting and $B = 8$, and 12, SDSH-S shows large improvements, and outperforms DVSQ by 7.2% and 5.7%. SDSH-LL performs slightly better than SDSH-S for $B > 12$, but has worse

[1]https://github.com/wvuvl/SDSH

Table 4.2: Mean Average Precision (MAP) Results for Different Number of Bits on NUS_WIDE

(a) Reduced A settings

	Number of Bits					
Method	8	12	16	24	32	48
DTSH[94]	-	0.773	-	**0.808**	**0.812**	**0.824**
SDSH-ML	0.758	0.770	0.784	0.798	0.802	0.810
SDSH-LL	0.751	0.780	0.792	0.805	0.810	0.817
SDSH-S	**0.774**	**0.789**	**0.796**	0.807	**0.812**	0.820

(b) Full settings

	Number of Bits			
Method	16	24	32	48
DTSH[94]	0.756	0.776	0.785	0.799
DPSH[93]	0.715	0.722	0.736	0.741
SDSH-ML	0.794	0.800	0.797	0.805
SDSH-LL	0.452	0.808	0.810	0.812
SDSH-S	**0.812**	**0.817**	**0.821**	**0.821**

(c) Reduced B settings

	Number of Bits			
Method	8	16	24	32
DVSQ[100]	**0.780**	**0.790**	**0.792**	**0.797**
DMDH [95]	-	0.751	-	0.781
SDSH-ML	0.739	0.771	0.785	0.791
SDSH-LL	0.750	0.771	0.782	0.789
SDSH-S	0.755	0.783	0.786	0.790

Table 4.3: Mean Average Precision (MAP) Results for Different Number of Bits of MNIST

	Number of Bits			
Method	16	24	32	48
CNNH[88]	0.957	0.963	0.956	0.960
CNNH+[88]	0.969	0.975	0.971	0.975
DSCH[189]	0.965	0.966	0.972	0.975
DRSCH[189]	0.969	0.974	0.979	0.979
SDSH-ML	0.993	**0.995**	0.994	0.995
SDSH-LL	**0.994**	0.994	**0.995**	**0.996**
SDSH-S	**0.994**	**0.995**	**0.995**	0.995

performance for lower number of bits. For reduced setting, SDSH-S and SDSH-LL perform about the same, and always outperforms the state-of-the-art methods with an exception for the 8-bit case. Figure 4.6 presents the data of Table 4.1 in the form of plots.

Figure 4.7 shows the comparison of the precision-recall curves of SDSH-S with those produced by two state-of-the-art approaches, namely DTSH [94] and DVSQ [100], highlighting the promising performance of the proposed approach.

Tables 4.2b, 4.2a, and 4.2c show average mAP scores on NUS_WIDE for Full, Reduced A, and Reduced B settings respectively. Unfortunately, the protocol of this experiment does not allow to compare against the BL value. Although [188] describes two additional

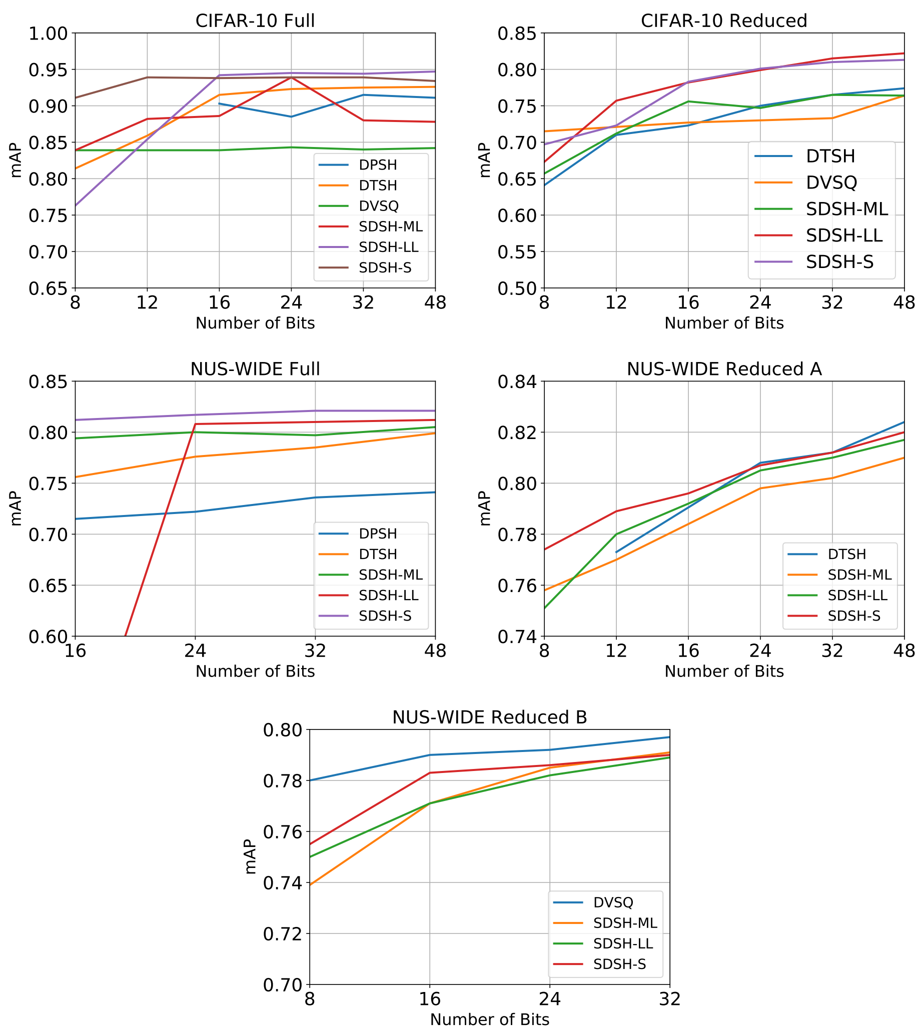


Figure 4.6: **Average mAP scores**. Comparison of mAP values w.r.t. bit number for our method (SDSH-ML, SDSH-LL, SDSH-S) with DPSH [93], DTSH[94] and DVSQ[100].

protocols, since they are suitable for tasks other than supervised hashing, for comparison with the state-of-the-art, here we follow protocols that have been in use by the widest majority of the literature. In particular, SDSH-S, SDSH-LL, and SDSH-ML always outperform the state-of-the-art methods on Full setting except for SDSH-LL at 8-bit case, where it converged poorly. On full setting, SDSH-S outperforms other type of losses for all bit numbers. On Reduced A setting, SDSH-S outperforms the state-of-the-art methods for $B < 24$ and for higher B it stays about the same as DTSH and slightly below for $B = 48$. On Reduced B setting our method is outperformed by DVSQ. It is important to note that NUS_WIDE has 81 labels but only 500 samples from the 21 most frequent labels are used for training.

Therefore, even though the training set for NUS-WIDE's reduced setting is still about twice as large as the training set for CIFAR-10's reduced setting, the ratio of samples per label for CIFAR is 500, while the ratio for this NUS_WIDE setting is on average 129.6 per label. Therefore, for NUS_WIDE reduced setting, the network is more prone to overfitting. As for CIFAR-10, the bottom row of Figure 4.6 presents the data of Tables 4.2b, 4.2a, and 4.2c in the form of plots.

In Table 4.3 we show a comparison between CNNH, CNNH+ [88], and DSCH, DRSCH [189] on the MNIST [124] dataset, noticing that all the three losses provide a performance increase.

4.7.3 Ablation Study

The proposed approach requires two steps for learning a hash function. The first step learns a spherical embedding that identifies an equivalence class of solutions because of the rotation invariance of the loss, and then a rotation needs to be identified to pick a representative of the equivalence class. Here we analyze what happens if use the identity as rotation, versus using the proposed method, versus using an off-the-shelve method like ITQ [74]. The summary of the results is shown in Figure 4.9, where the three approaches have been applied to SDSH-S, which has been tested on CIFAR-10, and NUS_WIDE. The first observation is that estimating the optimal rotation becomes more important at lower number of bits. As we suggested in Section 4.5, when B grows, the solution space grows significantly, so a random solution is more likely to do well. In addition, we note that ITQ tends to underperform our approach, and often decreases the performance of the identity solution. We note that ITQ differs from the proposed approach at least in two important aspects. First, ITQ is an unsupervised method, whereas our random search leverages the label information. In addition, ITQ aims at minimizing the quantization error, whereas the proposed method looks for the best quantization that maximizes the mAP. Finally, Figure 4.8 shows the performance improvement that adds rotation optimisation for different loss types, highlighting, as expected, that each of them can benefit from that step.

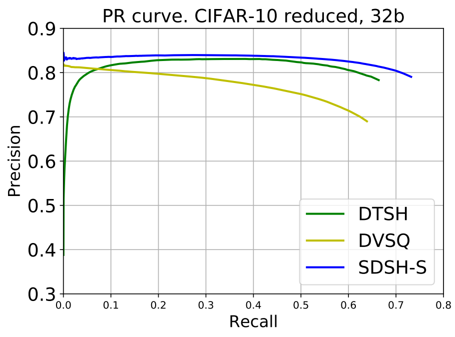


Figure 4.7: **Precision-Recall Curves** Comparison of P-R curves from our method, DVSQ [100] and DTSH [94] on CIFAR-10 reduced, top 5000 samples @ 32 bits.

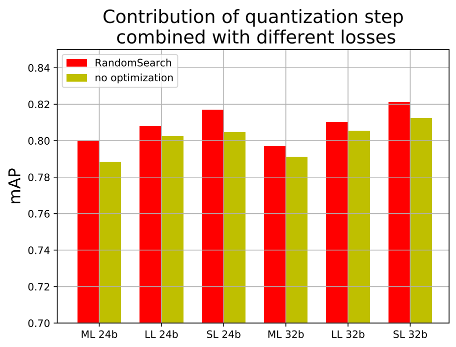


Figure 4.8: **Effect of quantization step**. Contribution of quantization step for different losses on NUS-WIDE Full @ 24 bits, 32bits.

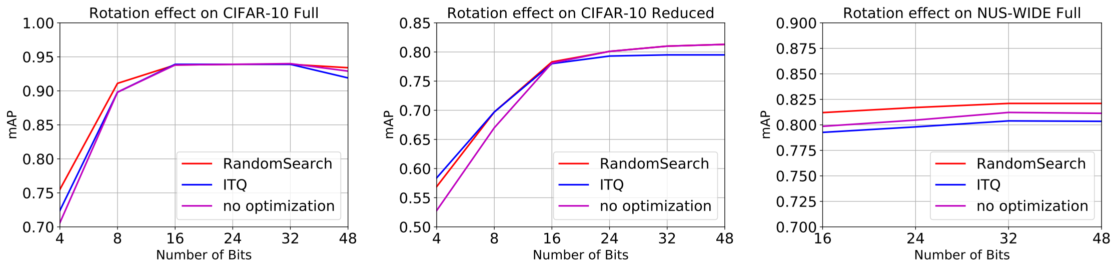


Figure 4.9: **Effect of learning rotation** Comparison of mAP values for a range of bit number for three scenarios: ITQ [74] and random search optimization and no rotation optimization.

4.8 Conclusions

We have introduced SDSH, a novel deep hashing method that moves beyond the sole goal of similarity preserving, and explicitly learns a hashing function that produces quality codes. This is achieved by leveraging a different relaxation method that eliminates the need for regularizing priors, and it enables the design of loss functions for learning balanced codes, and it allows to optimize the quantized hash function to maximize the mAP. Extensive experiments on three standard benchmark datasets demonstrated the strength of the approach. In particular, addressing the issue of quality in deep hashing approaches has revealed to be valuable, because the performance has increased particularly for more compact codes, which is very important for building efficient retrieval systems.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this dissertation we explored the reasons behind poor generative power of autoencoders compared to GANs, and designed a novel autoencoder architecture that overcomes these limitations. We explored application space of the introduced architecture and have shown that it significantly improves state of the art methods in novelty detection.

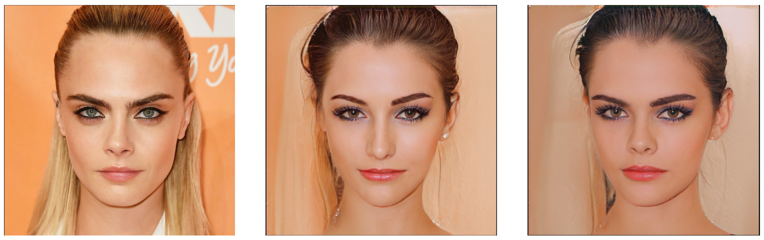
We introduced ALAE, a novel autoencoder architecture that is simple, flexible and general, as we have shown to be effective with two very different backbone generator-encoder networks. Differently from previous work, it allows learning the probability distribution of the latent space, when the data distribution is learned in adversarial settings. Our experiments confirm that this enables learning representations that are likely less entangled. This allows us to extend StyleGAN to StyleALAE, the first autoencoder capable of generating and manipulating images in ways not possible with StyleGAN alone, while maintaining the same level of visual detail.

We introduced GPND, an approach and a network architecture for novelty detection that is based on learning mappings f and g that define the parameterized manifold $\mathcal{M}$ which captures the underlying structure of the inlier distribution. Unlike prior deep learning based methods, GPND detects that a given sample is an outlier by evaluating its inlier probability distribution. We have shown how each architectural and model components are essential to the novelty detection. In addition, with a relatively simple architecture we have shown how GPND provides state-of-the-art performance using different measures, different datasets, and different protocols, demonstrating to also compare favorably with the out-of-distribution methods.

We have introduced SDSH, a novel deep hashing method that moves beyond the sole goal of similarity preservation, and explicitly learns a hashing function that produces quality codes. This is achieved by leveraging a different relaxation method that eliminates the need for regularizing priors, and it enables the design of loss functions for learning balanced codes, and it allows to optimize the quantized hash function to maximize the mAP. Extensive experiments on three standard benchmark datasets demonstrated the strength of the approach. In particular, addressing the issue of quality in deep hashing approaches has revealed to be valuable, because the performance has increased particularly for more compact codes, which is very important for building efficient retrieval systems.

5.2 Future Work

The proposed ALAE architecture introduces a new class of autoencoders that can be applied to large number of existing deep learning approaches in computer vision. We have deeply explored novelty detection application and have shown how the new architecture improves state of the art results. Potential future work would include an investigation of the application of ALAE to self-representation learning, generalization, similarity retrival and few-shot learning problems. In order for ALAE work well on human faces it is crucial to improve identity preservation. But even at current state it opens large amount of opportunities



(a) Input, real image (b) Reconstruction, no optimization (c) Reconstruction, with identity preservation

Figure 5.1: **Identity preservation**. Comparison of input image 5.1a, reconstruction 5.1b, and optimizaed reconstruction 5.1c

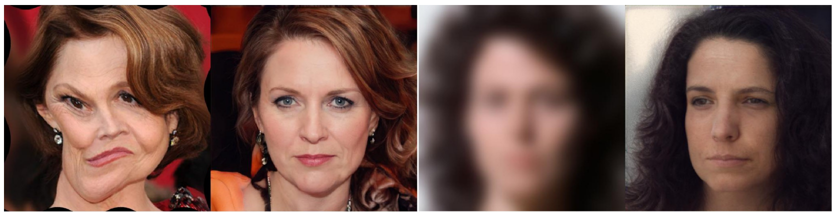


Figure 5.2: **Image restoration**.

for image manipulations. Here, we would like to present few attempts on using ALAE for such tasks.

Identity preservation. We add additional loss that forces the recontruction to maintain the same identy as the input. It tries to bring the reconstructed face perceptually closer to the input. We use FaceNet [179] to extract features from input and output. Then by using gradient decent, we adjust latent vector w to minimize MSE of the embeddings. The result of such optimization can be seen in Figure 5.1.

Image restoration. Since ALAE learns the manifold of the data, it is possible to take a corrupted sample and project it onto the manifold. ALAE can improve image restoration, superresolution, inpainting techniques. See Figure 5.2

Image manipulations. ALAE provides large space for image manipulations. Figure 5.3 demonstrates examples where two images were blended together in feature space.

Image translation. Using recent advances in data augmentation for training GANs [190],

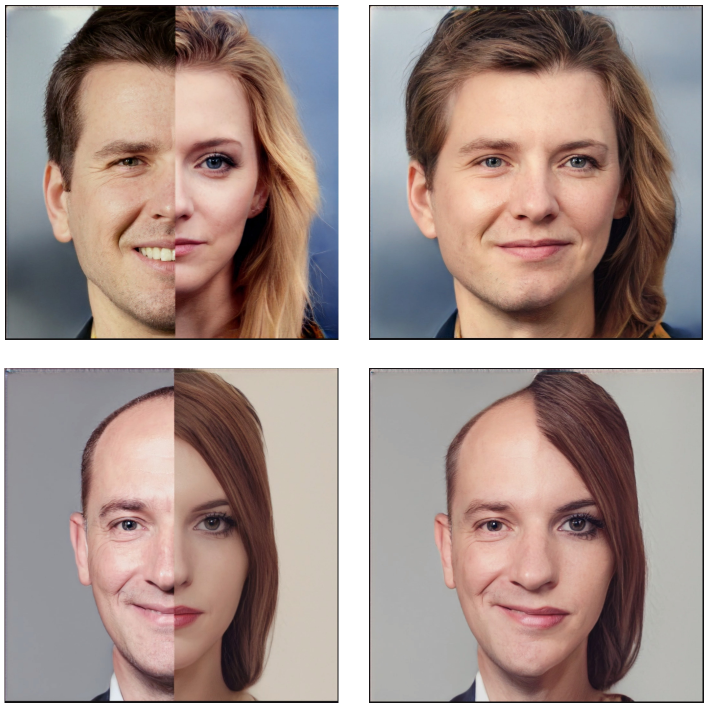


Figure 5.3: **Feature space manipulations** This figure shows concatenation of two images in feature space.

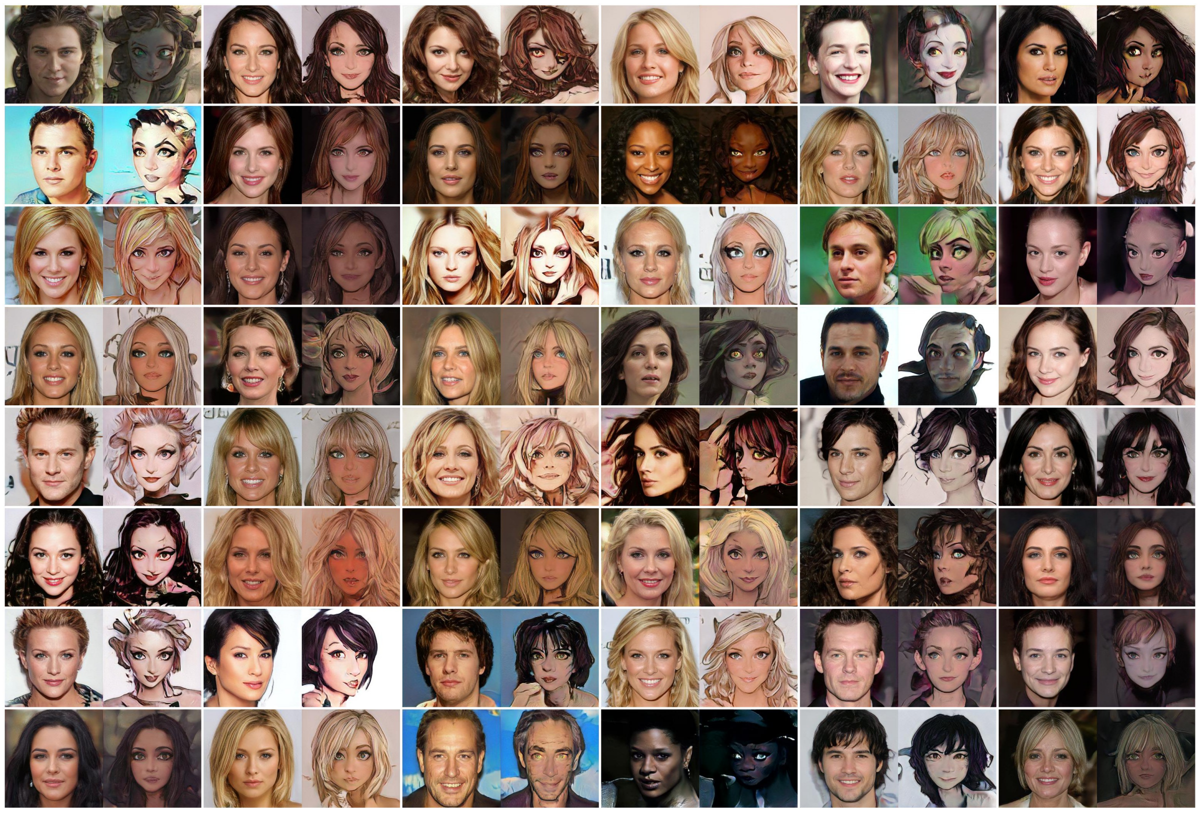


Figure 5.4: **Image to image translation**.

it became possible to fine-tune pretrained ALAE and other, smaller dataset. Combining encoder and generator trained on different domains, it is possible to use ALAE for image to image translation tasks as it is demonstrated in Figure 5.4, where generator was fine-tuned on a small toon dataset.

Bibliography

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2014.

[2] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533, 1986.

[4] Dana H Ballard. Modular learning in neural networks. In *AAAI*, pages 279–284, 1987.

[5] Ju-Seog Jang, Su-Won Jung, Soo-Young Lee, and Sang-Yung Shin. Optical implementation of the hopfield model for two-dimensional associative memory. *Optics Letters*, 13(3):248–250, 1988.

[6] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[7] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. In *ICLR*, 2016.

[8] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

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

[10] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. In *arXiv:1511.05644*, 2015.

[11] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223, 2018.

[12] D. P. Kingma and W. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[13] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation andapproximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

[14] 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. In *ICLR*, 2017.

[15] Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[16] Emily L Denton, Soumith Chintala, Rob Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1486–1494, 2015.

[17] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 5907–5915, 2017.

[18] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversar-

ial networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1947–1962, 2018.

[19] Zizhao Zhang, Yuanpu Xie, and Lin Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6199–6208, 2018.

[20] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[21] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[22] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[23] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

[24] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. In *arXiv:1701.07875*, 2017.

[25] J. Z. Nagarajan, V.and Kolter. Gradient descent GAN optimization is locally stable. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 5591–5600, 2017.

[26] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? *arXiv:1801.04406*, 2018.

[27] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *arXiv:1802.05957*, 2018.

[28] Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

[29] H. Kim and A. Mnih. Disentangling by factorising. In *ICML*, 2018.

[30] Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In *ICLR*, 2018.

[31] R. T. Q. Chen, X. Li, R. Grosse, and R. Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *NeurIPS*, 2018.

[32] Jakub M Tomczak and Max Welling. VAE with a VampPrior. In *AISTATS*, 2018.

[33] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning (ICML)*, pages 1558–1566, 2016.

[34] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. In *ICLR*, 2016.

[35] A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *NIPS*, 2017.

[36] D. Ulyanov, A. Vedaldi, and V. Lempitsky. It takes (only) two: Adversarial generator-encoder networks. In *AAAI*, 2018.

[37] Ari Heljakka, Arno Solin, and Juho Kannala. Pioneer networks: Progressively growing generative autoencoder. In *Asian Conference on Computer Vision (ACCV)*, pages 22–38. Springer, 2018.

[38] H. Huang, Z. Li, R. He, Z. Sun, and T. Tan. Introvae: Introspective variational autoencoders for photographic image synthesis. In *NIPS*, 2018.

[39] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016.

[40] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

[41] Arslan Basharat, Alexei Gritai, and Mubarak Shah. Learning object motion patterns for anomaly detection and improved object detection. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[42] Kenji Yamanishi, Jun-Ichi Takeuchi, Graham Williams, and Peter Milne. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.

[43] JooSeuk Kim and Clayton D Scott. Robust kernel density estimation. *Journal of Machine Learning Research*, 13(Sep):2529–2565, 2012.

[44] Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the International Conference on Machine Learning*. Citeseer, 2000.

[45] Paul Bodesheim, Alexander Freytag, Erik Rodner, Michael Kemmler, and Joachim Denzler. Kernel null space methods for novelty detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3374–3381. IEEE, 2013.

[46] Juncheng Liu, Zhouhui Lian, Yi Wang, and Jianguo Xiao. Incremental kernel null space discriminant analysis for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 792–800, 2017.

[47] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*, 2018.

[48] Ryota Hinami, Tao Mei, and Shin'ichi Satoh. Joint detection and recounting of abnormal events by learning deep generic knowledge. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3619–3627, 2017.

[49] Yang Cong, Junsong Yuan, and Ji Liu. Sparse reconstruction cost for abnormal event detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3449–3456. IEEE, 2011.

[50] Mahdi Soltanolkotabi, Emmanuel J Candes, et al. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, 40(4):2195–2238, 2012.

[51] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. Learning temporal regularity in video sequences. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, pages 733–742. IEEE, 2016.

[52] Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. Learning deep representations of appearance and motion for anomalous event detection. *arXiv preprint arXiv:1510.01553*, 2015.

[53] Huan-gang Wang, Xin Li, and Tao Zhang. Generative adversarial network based novelty detection usingminimized reconstruction error. *Frontiers of Information Technology & Electronic Engineering*, 19(1):116–125, 2018.

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

[55] Mahdyar Ravanbakhsh, Moin Nabi, Enver Sangineto, Lucio Marcenaro, Carlo Regazzoni, and Nicu Sebe. Abnormal event detection in videos using generative adversarial nets. *arXiv preprint arXiv:1708.09644*, 2017.

[56] Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1519, 2015.

[57] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3379–3388, 2018.

[58] Ki Hyun Kim, Sangwoo Shim, Yongsub Lim, Jongseob Jeon, Jeongwoo Choi, Byungchan Kim, and Andre S Yoon. Rapp: Novelty detection with reconstruction along projection pathway. In *International Conference on Learning Representations*, 2019.

[59] Seung Yeop Shin and Han-joon Kim. Extended autoencoder for novelty detection with reconstruction along projection pathway. *Applied Sciences*, 10(13):4497, 2020.

[60] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[61] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pages 4790–4798, 2016.

[62] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[63] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *NIPS*, pages 5574—-5584, 2017.

[64] D. Hendrycks and K. Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017.

[65] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*, 2018.

[66] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

[67] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. *arXiv preprint arXiv:2007.08176*, 2020.

[68] Xuming Ran, Mingkun Xu, Lingrui Mei, Qi Xu, and Quanying Liu. Detecting out-of-distribution samples via variational auto-encoder with reliable uncertainty estimation. *arXiv preprint arXiv:2007.08128*, 2020.

[69] Ankur Mallick, Chaitanya Dwivedi, Bhavya Kailkhura, Gauri Joshi, and T Yong-Jin Han. Can your ai differentiate cats from covid-19? sample efficient uncertainty estimation for deep learning safety. *choice*, 50:6.

[70] Davide Abati, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Latent space autoregression for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 481–490, 2019.

[71] Stanislav Pidhorskyi, Ranya Almohsen, Donald Adjeroh, and Gianfranco Doretto. Generative probabilistic novelty detection with adversarial autoencoders. In *Advances in Neural Information Processing Systems*, pages 6822–6833, 2018.

[72] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE TPAMI*, 40(4):769–790, 2018.

[73] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009.

[74] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE TPAMI*, 35(12):2916–2929, 2013.

[75] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[76] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137. IEEE, 2009.

[77] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009.

[78] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, pages 2938–2945, 2013.

[79] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial hashing network. In *CVPR*, pages 3664–3673, 2018.

[80] J. Heo, Y. Lee, J. He, S. Chang, and S. Yoon. Spherical hashing: Binary code embedding with hyperspheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2304–2316, 2015.

[81] Mohammad Norouzi and David M Blei. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011.

[82] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081. IEEE, 2012.

[83] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. Supervised hashing with latent factor models. In *ACM SIGIR*, pages 173–182, 2014.

[84] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[85] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *NIPS*, pages 2553–2561, 2013.

[86] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[87] Naiyan Wang and Dit-Yan Yeung. Learning a deep compact image representation for visual tracking. In *NIPS*, pages 809–817, 2013.

[88] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, volume 1, pages 2156–2162, 2014.

[89] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

[90] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016.

[91] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, pages 2064–2072, 2016.

[92] Yue Cao, Mingsheng Long, Bin Liu, Jianmin Wang, and MOE KLiss. Deep cauchy hashing for hamming space retrieval. In *CVPR*, pages 1229–1237, 2018.

[93] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[94] Xiaofang Wang, Yi Shi, and Kris M Kitani. Deep supervised hashing with triplet labels. *ACCV*, 2016.

[95] Zhixiang Chen, Xin Yuan, Jiwen Lu, Qi Tian, and Jie Zhou. Deep hashing via discrepancy minimization. In *CVPR*, pages 6838–6847, 2018.

[96] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016.

[97] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE TPAMI*, 36(4):744–755, 2014.

[98] Ting Zhang, Chao Du, and Jingdong Wang. Composite quantization for approximate nearest neighbor search. In *ICML*, number 2, pages 838–846, 2014.

[99] Xiaojuan Wang, Ting Zhang, Guo-Jun Qi, Jinhui Tang, and Jingdong Wang. Supervised quantization for similarity search. In *CVPR*, pages 2018–2026, 2016.

[100] Yue Cao, Mingsheng Long, Jianmin Wang, and Shichen Liu. Deep visual-semantic quantization for efficient image retrieval. In *CVPR*, 2017.

[101] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. [to appear].

[102] Stanislav Pidhorskyi, Quinn Jones, Saeid Motiian, Donald Adjeroh, and Gianfranco Doretto. Deep supervised hashing with spherical embedding. In *Asian Conference on Computer Vision*, pages 417–434. Springer, 2018.

[103] Zhong Qiu Lin Linda Wang and Alexander Wong. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest radiography images, 2020.

[104] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738, 2015.

[105] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop. In *arXiv:1506.03365*, 2015.

[106] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[107] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, abs/1409.1556, 2015.

[108] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[109] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[110] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[111] M. Mirza and S. Osindero. Conditional generative adversarial nets. In *arXiv:1411.1784*, 2014.

[112] Jun Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *IEEE International Conference on Computer Vision (ICCV)*, 2017-Octob:2242–2251, 2017.

[113] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The gan landscape: Losses, architectures, regularization, and normalization. In *arXiv:1807.04720*, 2018.

[114] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are gans created equal? a large-scale study. In *Advances in neural information processing systems (NeurIPS)*, pages 700–709, 2018.

[115] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[116] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[117] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[118] Harris Drucker and Yann Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992.

[119] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI Conference on Artificial Intelligence*, 2018.

[120] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[121] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[122] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[123] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

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

[125] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[126] Jürgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[127] Karl Ridgeway. A survey of inductive biases for factorial representation-learning. *arXiv preprint arXiv:1612.05299*, 2016.

[128] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[129] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[130] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[131] Ari Heljakka, Arno Solin, and Juho Kannala. Pioneer networks: Progressively growing generative autoencoder. In *Asian Conference on Computer Vision*, pages 22–38. Springer, 2018.

[132] Ari Heljakka, Arno Solin, and Juho Kannala. Towards photographic image manipulation with balanced growing of generative autoencoders. *arXiv preprint arXiv:1904.06145*, 2019.

[133] Z. Ge, Z. Song, and F. Gao. Review of recent research on data-based process monitoring. *Ind. Eng. Chem. Res.*, 52(10):3543–3562, 2013.

[134] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Marc Niethammer, Martin Styner, Stephen Aylward, Hongtu Zhu, Ipek Oguz, Pew-Thian Yap, and Dinggang Shen, editors, *Information Processing in Medical Imaging*, pages 146–157, Cham, 2017.

[135] Artur Kadurin, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. drugan: An advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular Pharmaceutics*, 14(9):3098–3104, 2017. PMID: 28703000.

[136] W. Li, V. Mahadevan, and N. Vasconcelos. Anomaly detection and localization in crowded scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):18–32, 2014.

[137] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette. Deep-cascade: Cascading 3d deep neural

networks for fast anomaly detection and localization in crowded scenes. *IEEE Transactions on Image Processing*, 26(4):1992–2004, 2017.

[138] Chong You, Daniel P Robinson, and René Vidal. Provable self-representation based outlier detection in a union of subspaces. *arXiv preprint arXiv:1704.03925*, 2017.

[139] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215 – 249, 2014.

[140] Shehroz S. Khan and Michael G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[141] M Sabokrou, M Fathy, and M Hoseini. Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder. *Electronics Letters*, 52(13):1122–1124, 2016.

[142] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[143] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[144] Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518–523, 1995.

[145] Larry Manevitz and Malik Yousef. One-class document classification via neural networks. *Neurocomputing*, 70(7-9):1466–1481, 2007.

[146] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[147] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

[148] Alireza Makhzani and Brendan J Frey. Pixelgan autoencoders. In *Advances in Neural Information Processing Systems*, pages 1972–1982, 2017.

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

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

[151] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. In *Advances in Neural Information Processing Systems*, pages 3308–3318, 2017.

[152] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. It takes (only) two: Adversarial generator-encoder networks. *arXiv preprint arXiv:1704.02304*, 2017.

[153] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[154] Pramuditha Perera, Ramesh Nallapati, and Bing Xiang. Ocgan: One-class novelty detection using gans with constrained latent representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2898–2906, 2019.

[155] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.

[156] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[157] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[158] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[159] FYYZS Song and Ari Seff Jianxiong Xiao. Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv: 1506.03365*, 2015.

[160] Pingmei Xu, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*, 2015.

[161] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[162] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[163] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[164] HDK Moonesinghe and Pang-Ning Tan. Outrank: a graph-based outlier detection framework using random walk. *International Journal on Artificial Intelligence Tools*, 17(01):19–36, 2008.

[165] Mostafa Rahmani and George K Atia. Coherence pursuit: Fast, simple, and robust principal component analysis. *IEEE Transactions on Signal Processing*, 65(23):6260–6275, 2016.

[166] Gilad Lerman, Michael B McCoy, Joel A Tropp, and Teng Zhang. Robust computation of linear models by convex relaxation. *Foundations of Computational Mathematics*, 15(2):363–410, 2015.

[167] Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust pca via outlier pursuit. In *Advances in Neural Information Processing Systems*, pages 2496–2504, 2010.

[168] Guangcan Liu, Zhouchen Lin, and Yong Yu. Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 663–670, 2010.

[169] Manolis C Tsakiris and René Vidal. Dual principal component pursuit. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 10–18, 2015.

[170] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.

[171] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *IEEE FOCS*, pages 459–468, 2006.

[172] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[173] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, 2015.

[174] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[175] J. Wang, S. Kumar, and S. F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431, 2010.

[176] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *IEEE TPAMI*, 34(1):66–78, 2012.

[177] M. Norouzi, D. M. Blei, and R. R. Salakhutdinov. Hamming distance metric learning. In *NIPS*, 2012.

[178] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014.

[179] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[180] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[181] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[182] Dmytro Mishkin and J. L. Matas. All you need is a good init. *CoRR*, abs/1511.06422, 2015.

[183] Maris Ozols. How to generate a random unitary matrix, 2009.

[184] R. E. Schwartz. The five-electron case of thomson's problem. *Experimental Mathematics*, 22(2):157–186, 2013.

[185] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: A real-world web image database from national university of singapore. In *ACM CIVR*, pages 48:1–48:9, 2009.

[186] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, pages 1556–1564, 2015.

[187] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[188] Alexandre Sablayrolles, Matthijs Douze, Nicolas Usunier, and Hervé Jégou. How should we evaluate supervised hashing? In *ICASSP*, pages 1732–1736, 2017.

[189] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE TIP*, 24(12):4766–4779, 2015.

[190] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020.

Appendix

Figure 5.5 shows what happens when we perform a bilinear interpolation in the $\mathcal{W}$ space of the representation of four images that we downloaded from the internet. The model used is StyleALAE trained on FFHQ. For a given interpolation an image is generated by drawing noise and using the generator network G. As it can be seen, the traversal that we obtain seems to vary very smoothly.

Figure 5.6 shows examples of generations from StyleALAE and StyleGAN. From a visual inspection it is very difficult to notice major differences. Figure 5.8 shows a random collection of generations obtained from StyleALAE.

Figure 5.9 and Figure 5.10 instead show a collection of reconstructions. We note that the original images are faces of celebrities that we have downloaded from the internet. Therefore they are not images from FFHQ. This is important because we argue that these images come from a distribution different from the FFHQ distribution. Indeed, FFHQ is a dataset of faces obtain from Flickr.com. They represent faces of non-celebrity people. More often than not, the faces do not wear any makeup, neither have they been altered (e.g., with Photoshop or other tools). In addition, the lighting conditions and cameras with which the FFHQ face images have been acquired, are normally very different from typical photoshoot stages where professional equipment is also used. Overall, this can change significantly the statistics of the images from FFHQ in comparison with those of celebrities. Nevertheless, we observe that StyleALAE works effectively on both reconstruction and mixing.

Differently from Figure 5.9 and Figure 5.10, Figure 5.11 shows reconstructions from images extracted from our testing split of the FFHQ dataset. Therefore, they should, in principle, belong to the same distribution used for training the model.

Figure 5.12 instead shows a random set of generations from the StyleALAE model trained on LSUN-Bedroom, whereas Figure 5.13 shows bedroom reconstructions from the same dataset.

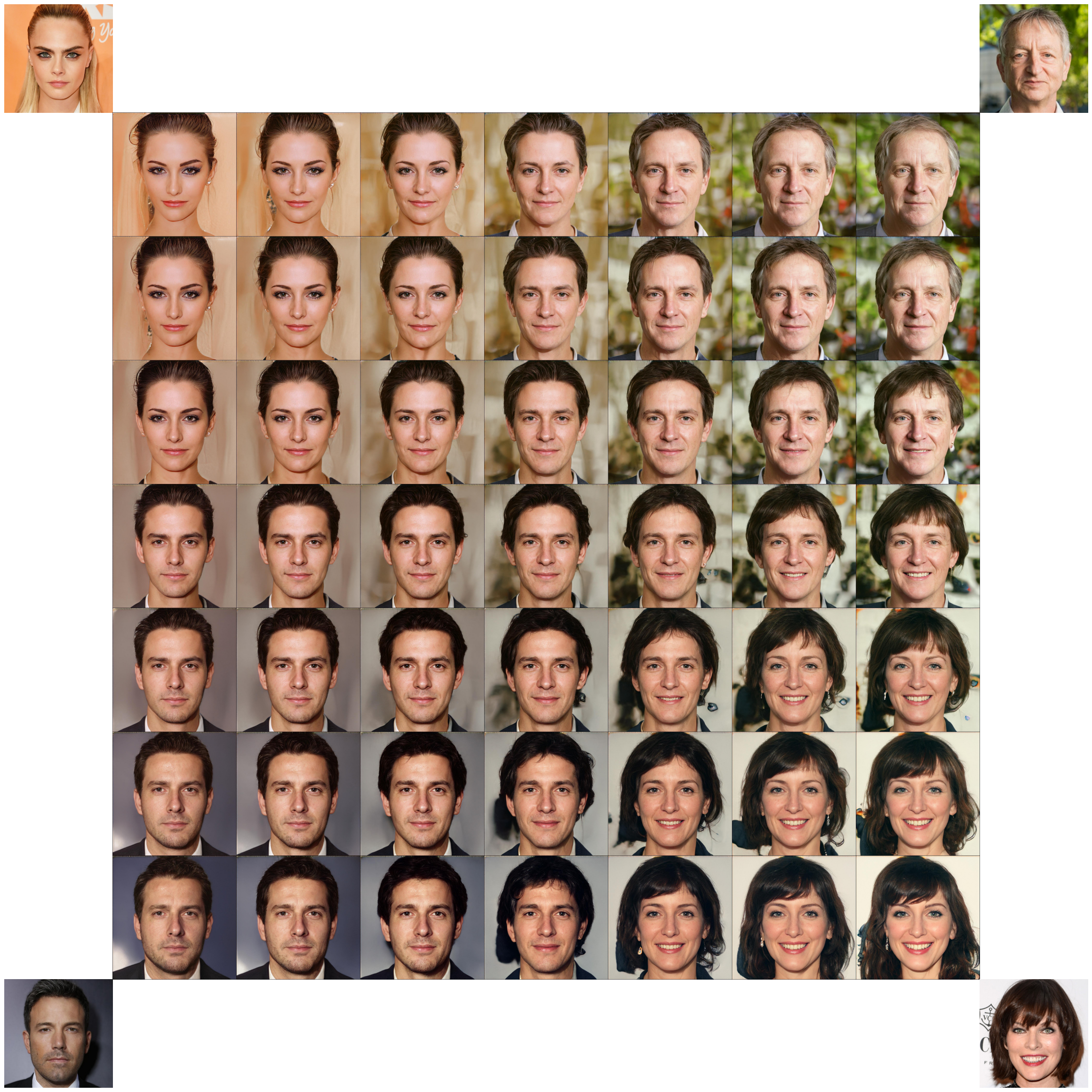


Figure 5.5: **Real image interpolations.** Qualitative results for interpolations of reconstructed images using StyleALAE trained on FFHQ with resolution 1024×1024. The images at the corners are real, and were not part of the training portion of FFHQ. All other images were obtained by bilinear interpolation in the latent space $\mathcal{W}$.

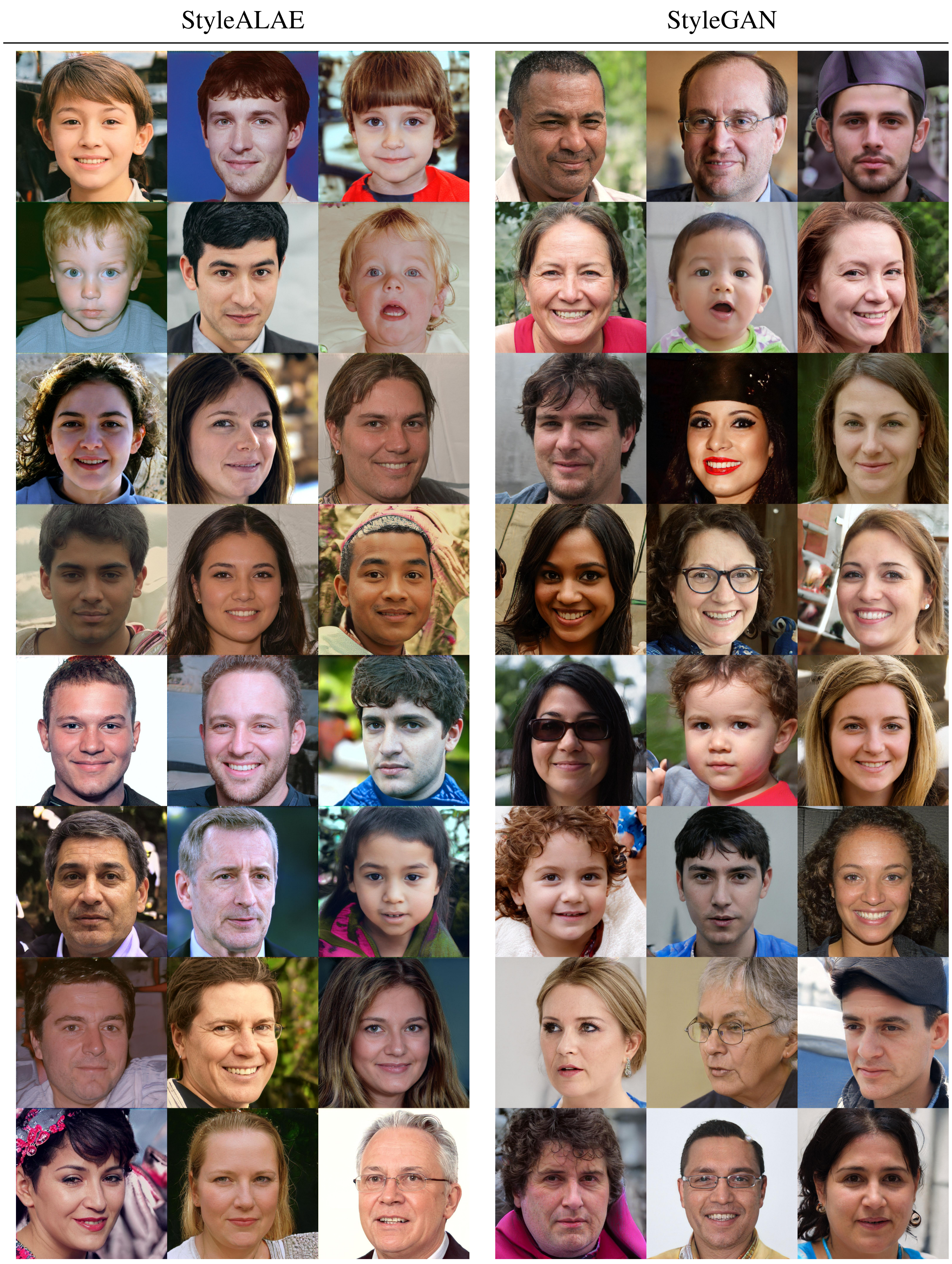


Figure 5.6: **StyleALAE vs StyleGAN generations on FFHQ dataset.** Original single face image resolution is 1024×1024.

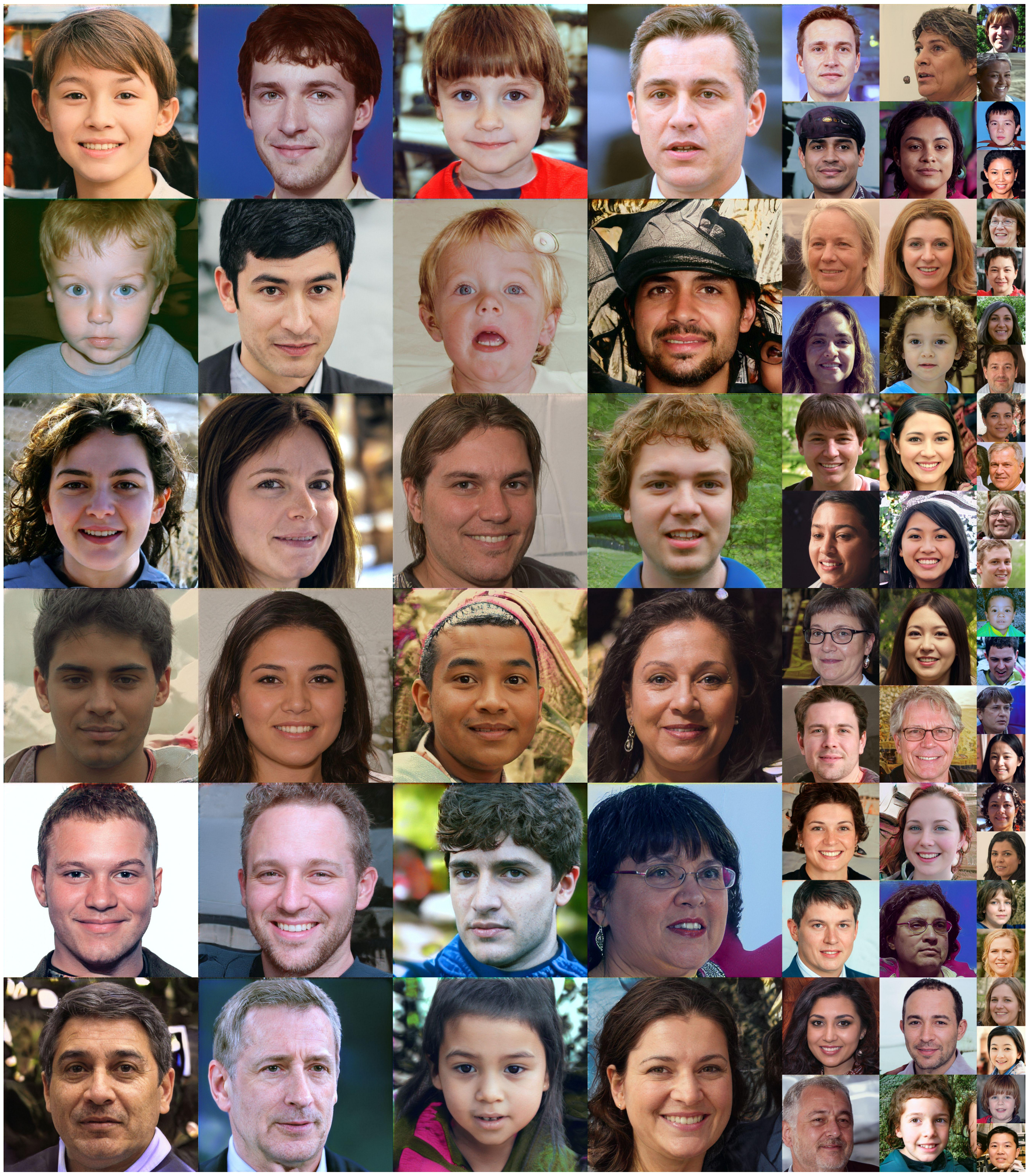


Figure 5.7: **StyleALAE generations on FFHQ dataset.** Original single face image resolution is 1024×1024.

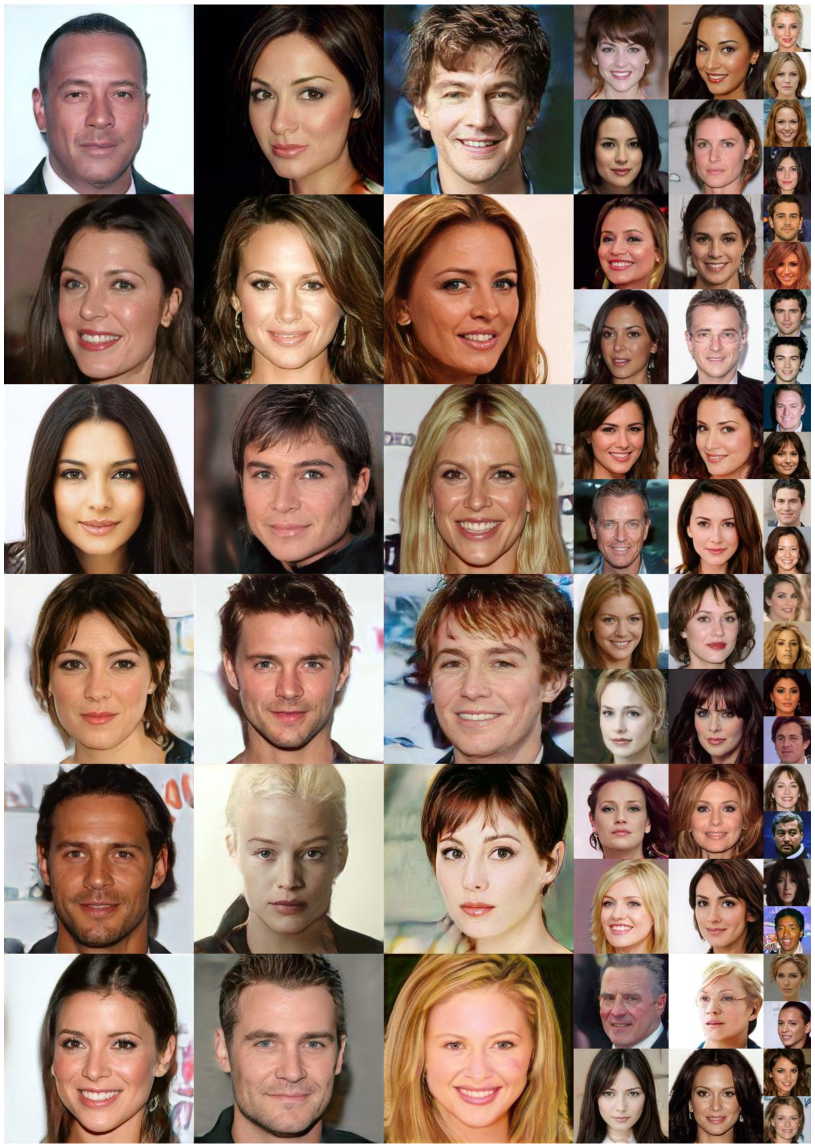


Figure 5.8: **StyleALAE generations on Celeba-HQ dataset.** Original single face image resolution is 256×256.

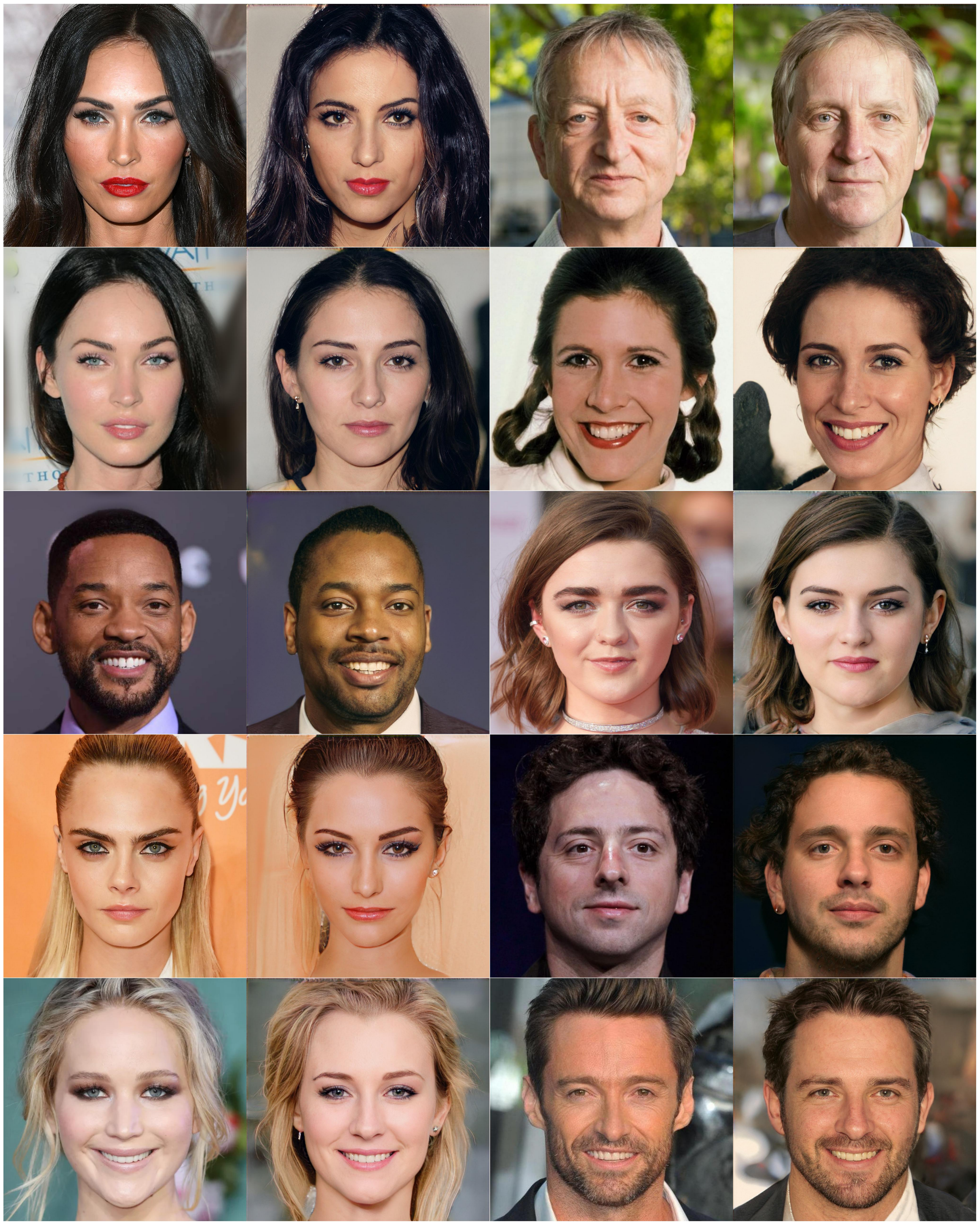


Figure 5.9: **StyleALAE reconstructions on FFHQ dataset.** Reconstructions from images that are not part of FFHQ. Original single face image resolution is 1024×1024.

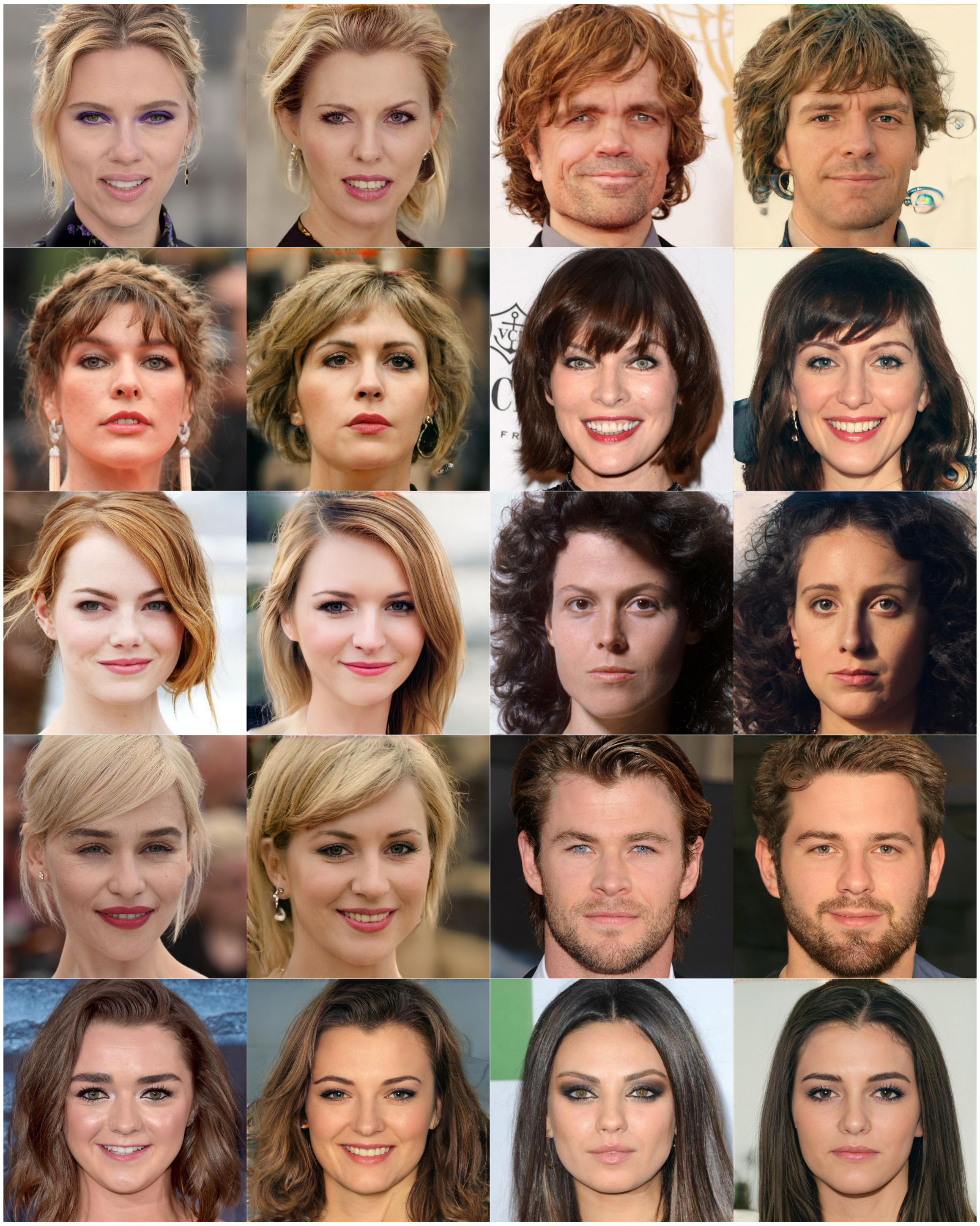


Figure 5.10: **StyleALAE reconstructions on FFHQ dataset.** Reconstructions from images that are not part of FFHQ. Original single face image resolution is 1024×1024.

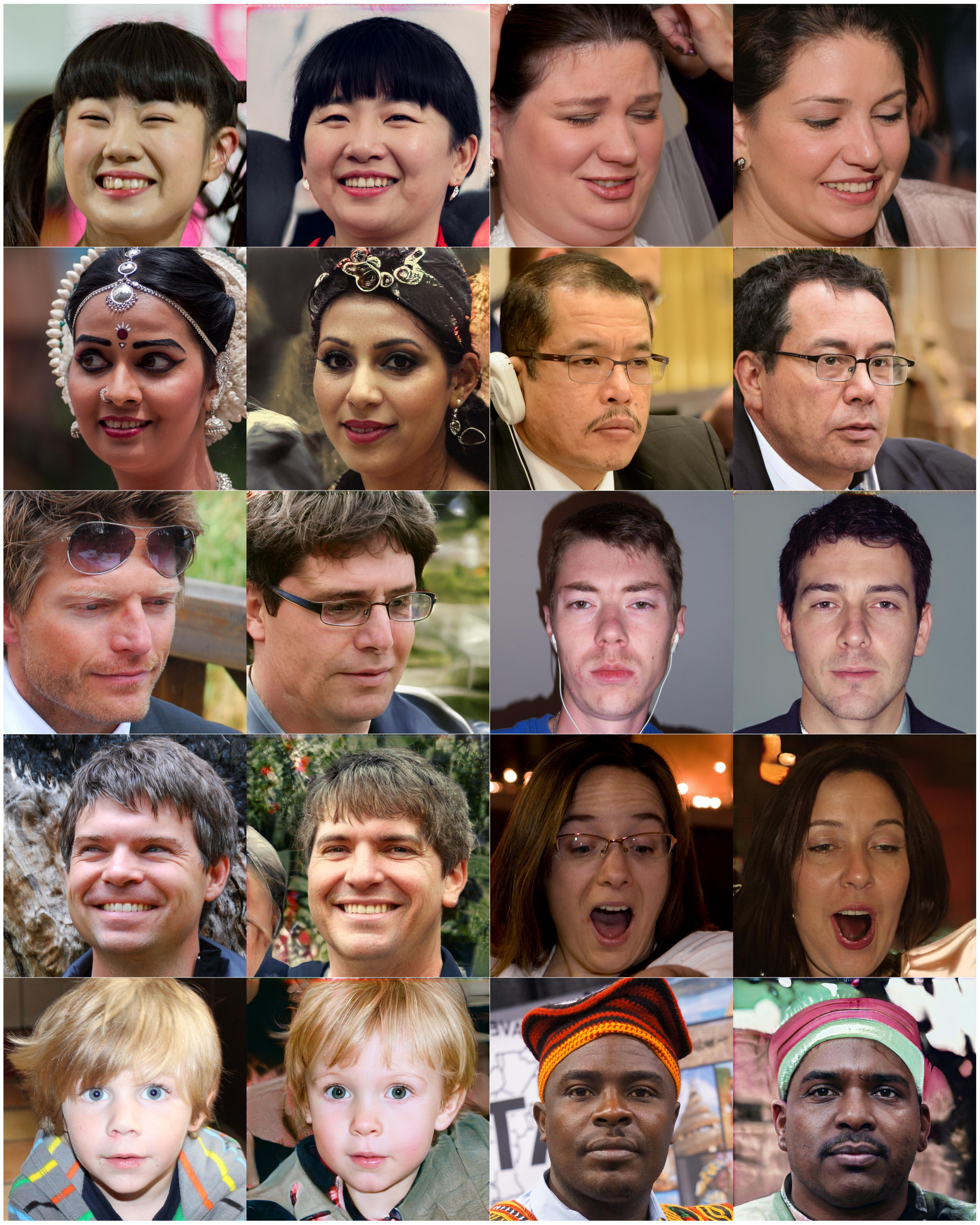


Figure 5.11: **StyleALAE reconstructions on FFHQ dataset.** Reconstructions from images that are from the test split of FFHQ. Original single face image resolution is 1024×1024.

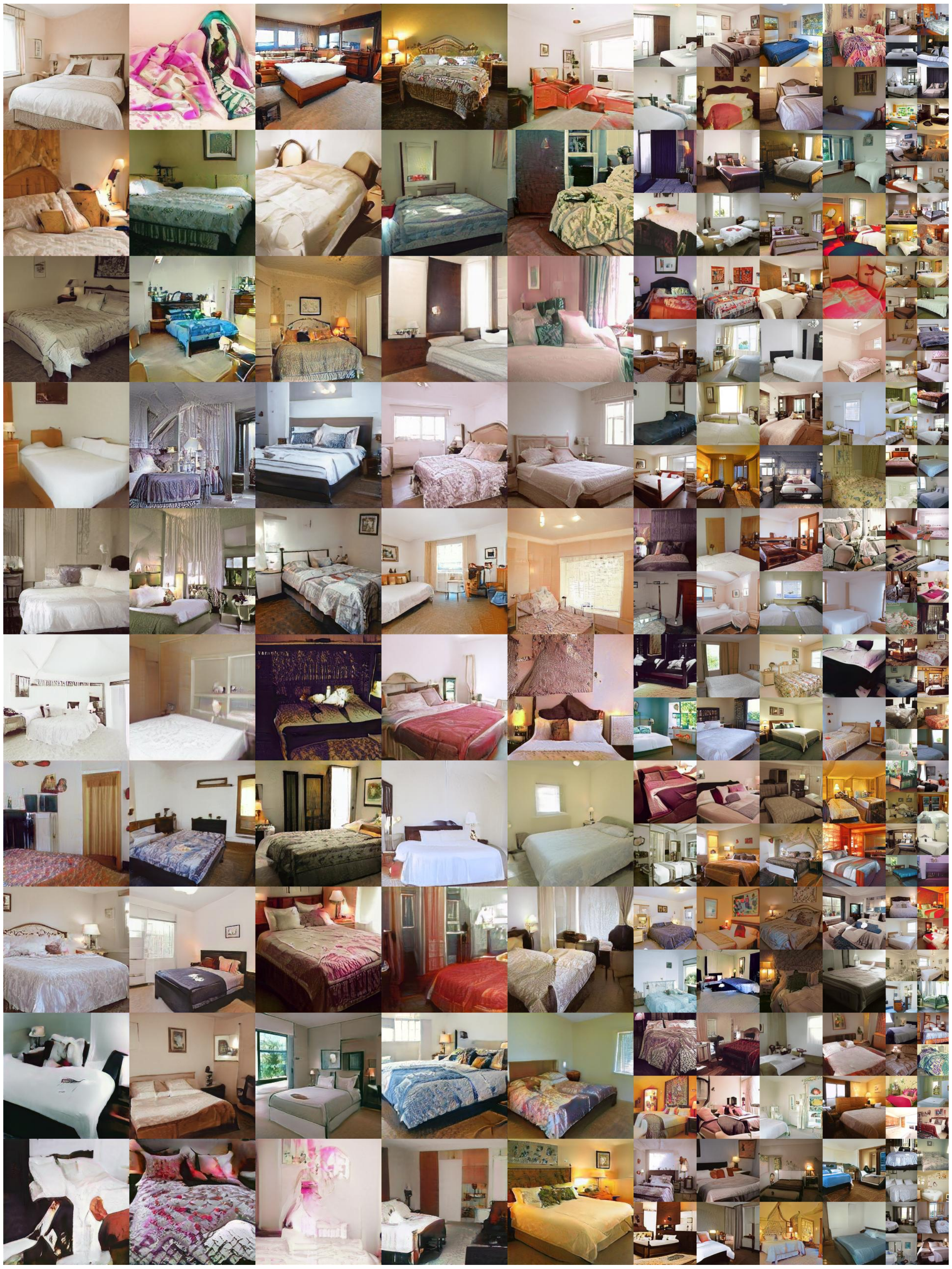


Figure 5.12: **StyleALAE generations on LSUN-Bedroom.** Original image resolution is 256 × 256.

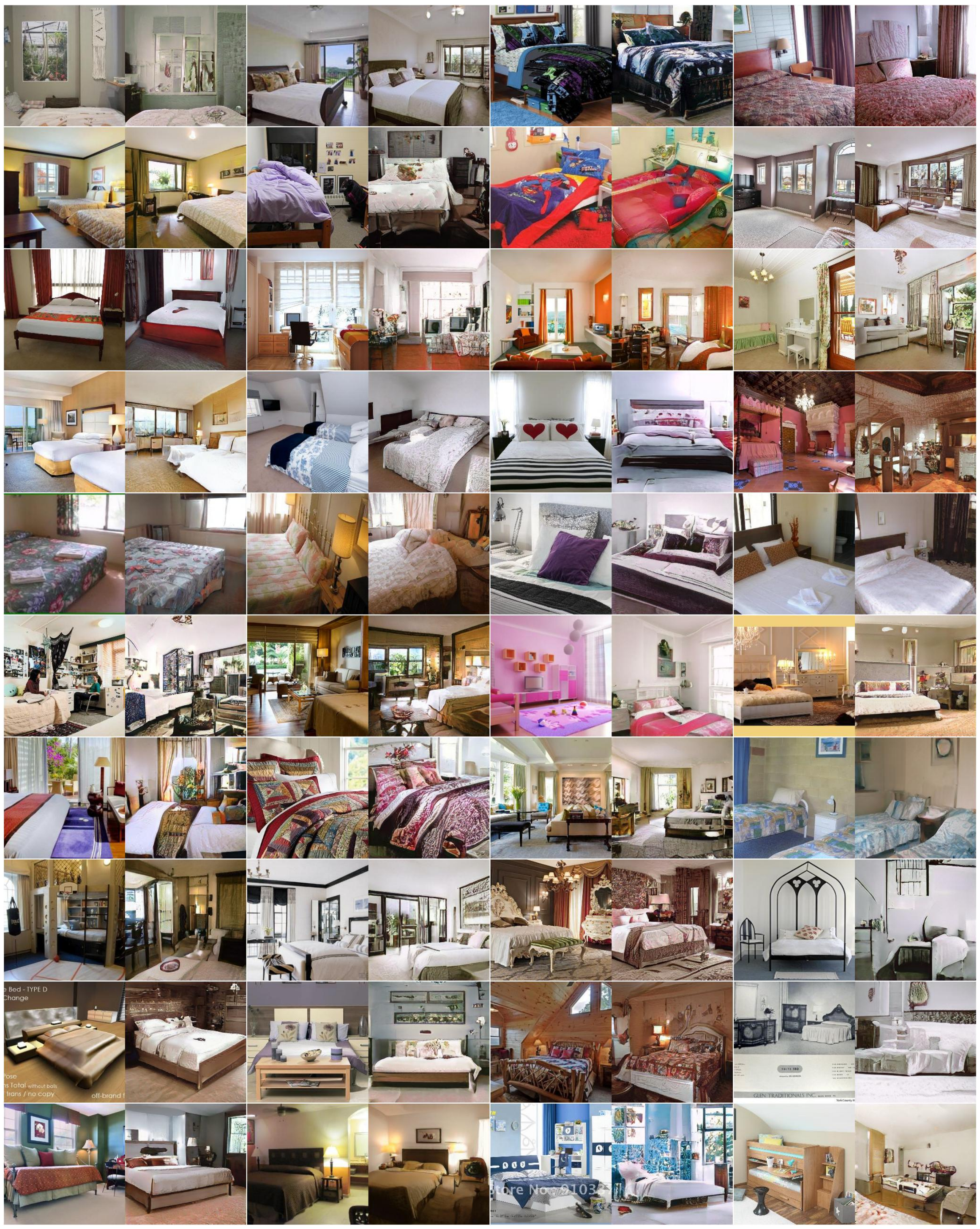


Figure 5.13: **StyleALAE reconstructions on LSUN-Bedroom.** Reconstructions from test split of LSUN-Bedroom. Original image resolution is 256 × 256.

Note that the reconstructions are from images taken from the test split, thus, they were not used for training.

Table 2.6 shows a comparison between StyleALAE and other approaches using the CelebA-HQ with resolution of 256×256. The FID score is computed between 50000 generated samples and the training samples. StyleALAE has the best FID score among the autoencoder type of models, and the best PPL.