UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING PHYSICAL AND SCIENCES

School of Electronics and Computer Science

Learning Factorised Representation Via Generative Models.

by

Zezhen Zeng

Thesis for the degree of Doctor of Philosophy

Supervisor: Professor Adam Prügel-Bennett
Co-Supervisor: Dr Jonathon Hare
Examiner: VLC

August 1, 2022

ABSTRACT

Deep learning has been widely used in real-life applications during the last few decades, such as face recognition, machine translation, object detection and classification. Representation learning is an important part of deep learning, which can simply be understood as a method for dimension reduction. However, the representation learned by the task-specific model is hard to be applied to other tasks without parameter tuning, since it discards irrelevant information from the input. While for generative models, the model can learn a joint distribution over all variables and the latent space can almost maintain the whole information of the dataset rather than task-specific information. But the vanilla generative models can only learn an entangled representation which cannot be used efficiently. Thus, a factorised representation is needed in most cases. Focus more on images, this thesis proposes new methods to learn a factorised representation.

This thesis starts by figuring out the quality of the representation learned by the backbone model Variational Autoencoder (VAE) visually. The proposed tool alleviates the blurriness of the vanilla VAE by introducing a discriminator. Then the potential of the VAE on transfer learning is explored. Collecting data is expensive, especially with labels. Transfer learning is one way to solve this issue. The results show a strong ability of the VAE on generalisation, which means the VAE can produce reasonable results even without parameter tuning. For factorised representation learning, this thesis follows a rule from a shallow level to a deep level. We propose a VAE-based model that can learn a latent space that factorises the foreground and the background of images, while the foreground in the experiments is defined as the objects inside the given bounding box labels. This factorised latent space allows the model to do conditional generation. The results can achieve a state-of-the-art Fréchet inception distance (FID) score. Then we investigate the unsupervised object-centric representation learning, which can be seen as a deeper level of the foreground representation. By observing that the object area tends to contain more information than the background in a multi-object scene, the model is designed to discover objects according to this difference. A better result can be obtained on the downstream task with the learned representation when compared to other related models.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Zezhen Zeng , declare that the thesis entitled *Learning Factorised Representation Via Generative Models.* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: conference paper in the 3rd International Conference on Pattern Recognition and Artificial Intelligence 2022

Signed:...

Date:..

Acknowledgements

I would like to thank both my supervisors Adam Prügel-Bennett and Jonathon Hare for their support and contribution throughout my PhD journey. Thanks for their mentoring and showing me how to do research and how to be a good PhD. I also want to thank all the academics and students of the VLC, it is a lovely and friendly group. Specially thanks to my parents, for their emotional and financial support. In the end, I also want to thank myself, for going through a PhD during the Covid-19 pandemic.

Chapter 1

Introduction

Deep learning is a fast-moving sub-field of machine learning that has been applied to many real problems such as classification(Krizhevsky et al., 2012), image segmentation(Girshick et al., 2014), image generation(Kingma and Welling, 2013), visual question answering(Antol et al., 2015) and representation learning(Bengio et al., 2013). Our work focus on image generation and representation learning. Image generation can be used to create new plausible samples on demand, it asks models to learn a joint probability distribution from the dataset and then generate entirely new images from random latent codes which are sampled from the distribution. The generated images can be helpful for the downstream tasks such as classification. There are several different types of methods for this task: the latent variable model such as the Variational Autoencoder (VAE) (Kingma and Welling, 2013; Rezende et al., 2014), the adversarial model such as the Generative Adversarial Network (GAN) (Goodfellow et al., 2014) and the autoregressive model such as PixelRNN(Van Oord et al., 2016). Our work is mainly based on VAEs and GANs, thus, we introduce the vanilla VAE and GAN in this chapter. For most of the deep learning methods, the performance highly depends on the data representation. Thus, we need to understand what kind of representation is good. So We introduce common priors for representation learning in this chapter. We mainly use VAEs as the backbone in the representation learning problem. At the end of the chapter, we summarize our motivations and contributions.

1.1 Variational Autoencoders

The intractability of posterior distributions is a notorious problem. It is hard to train a probabilistic model that can perform efficient inference with an intractable posterior distribution, especially in the deep learning field, the dataset is large in both dimension and amount. However, the VAE is a variational inference algorithm that can mitigate the intractability efficiently with approximate latent variables. Kingma and Welling (2013) proposed to use a neural network to model the posterior probability and maximize the evidence lower bound (ELBO).

Let $\boldsymbol{x}$ be a set of input variables, $\hat{\boldsymbol{x}}$ a set of the reconstruction of the $\boldsymbol{x}$ and $\boldsymbol{z}$ a set of latent variables of $\boldsymbol{x}$. In the generative model, we want to maximize the marginal likelihood.

$$p_\theta(\boldsymbol{x}) = \int p_\theta\left(\boldsymbol{z}\right) p_\theta\left(\boldsymbol{x}|\boldsymbol{z}\right) d\boldsymbol{z}.$$

where θ are the parameters of the network. However, the posterior probability $p_\theta\big(\boldsymbol{x}|\boldsymbol{z}\big) = p_\theta\big(\boldsymbol{z}|\boldsymbol{x}\big)\, p_\theta(\boldsymbol{z})/p_\theta(\boldsymbol{x})$ is intractable in this setting. It is hard to compute the posterior probability with finite computational power. Thus, the VAE introduces an inference model to infer the distribution $q_\phi\big(\boldsymbol{z}|\boldsymbol{x}\big)$ with parameters ϕ rather than compute the $p_\theta\big(\boldsymbol{z}|\boldsymbol{x}\big)$ directly. To be simplified, we use $\mathbb{E}_z$ to denote $\mathbb{E}_{z\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}$,

$$\begin{aligned}
\log p_\theta(\boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{z}}\left[\log \frac{p_\theta(\boldsymbol{x},\boldsymbol{z})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\boldsymbol{z}}\left[\log \frac{p(\boldsymbol{z})p_\theta(\boldsymbol{x}|\boldsymbol{z})q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\boldsymbol{z}}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z}}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right] + \mathbb{E}_{\boldsymbol{z}}\left[\log \frac{q_\phi(\boldsymbol{z}|\boldsymbol{x})}{p_\theta(\boldsymbol{z}|\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\boldsymbol{z}}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - D_{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big) + D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z}|\boldsymbol{x})).
\end{aligned} \tag{1.1}$$

The third term in the right hand side (RHS) is a Kullback-Leibler divergence which is non-negative. Thus, maximizing the first two terms in the RHS which is called the evidence lower bound (ELBO), is equivalent to maximizing the left hand side (LHS). The ELBO is a lower bound on the log-likelihood of the dataset, then we obtain the following inequality:

$$\log p_\theta(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z}}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - D_{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big) = L\left(\theta,\phi;\boldsymbol{x}\right) \tag{1.2}$$

The ELBO $L(\theta, \phi; \boldsymbol{x})$ is the objective function of the VAE. The first term in the RHS of inequality 1.2 is what we called reconstruction error and the second term is the KL term. However, we cannot backpropagate gradients through the variables $\boldsymbol{z}$ since $\boldsymbol{z}$ is directly sampled from $q_\phi\left(\boldsymbol{z}|\boldsymbol{x}\right)$. It can be solved by applying a change of variables $\boldsymbol{z}$, which is called the reparameterization trick. It proposes to add a noise variable from standard Gaussian distribution (or other distribution) to obtain a differentiable sample. To be detailed, we use a neural network to inference the mean and variance of the input data, which is μ and σ^2 in the following sections. The original $\boldsymbol{z} \sim q_\phi\left(\boldsymbol{z}|\boldsymbol{x}\right) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ is difficult to estimate the gradient through Monte Carlo. Thus, we reparameterize the $\boldsymbol{z}$ as $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a noise variable $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, $\odot$ is the element-wise product. This makes $\boldsymbol{z}$ differentiable with respect to $\boldsymbol{\phi}$. Once the model is trained by optimizing the $\mathcal{L}(\theta, \phi; \boldsymbol{x})$, a latent space representation $\boldsymbol{z}$ of $\boldsymbol{x}$ can be obtained from the inference model.

Currently, the VAE is an important backbone model for image generation and representation learning. And the VAE tends to be a basic module in sophisticated model. The reparameterization trick that has been proposed in VAE is also widely used even when the arichtecture of VAE is not been adapted.

Higgins et al. (2017a) illustrate the relation between the two terms in the ELBO and modified the objective function of the VAE as equation 1.3, which is called a β-VAE, :

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{z}}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \beta D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) || p(\boldsymbol{z})\right). \tag{1.3}$$

A higher β ($\beta > 1$) puts extra constraints to the capacity of latent variables $\boldsymbol{z}$ since the second term in equation 1.3 forces the posterior $q_\phi\left(\boldsymbol{z}|\boldsymbol{x}\right)$ to match the unit Gaussian prior $p(\boldsymbol{z})$. Thus, the choice of β brings a trade-off between the disentanglement and high-fidelity of reconstructions. This is due to the information loss since the capacity of latent bottleneck is restricted.

During our research, we found in the most implementations of VAE papers they use MSE error or chose a constant variance as the first term in the ELBO. In the original paper this was achieved by the decoder outputting a probability distribution akin to what happens in the latent space. More often it is assumed that the pixel errors are normally distributed with some variance σ^2. Thus the log-probability of generating all the images is

$$\sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{E}_{\boldsymbol{z}}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] = \sum_{i=1}^{N} \log\left(\mathcal{N}(\boldsymbol{x}|\hat{\boldsymbol{x}}, \sigma^2)\right) = -\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{2\sigma^2} - \frac{N}{2}\log(2\pi\sigma^2)$$

where $\mathcal{X}$ is the dataset and the sum is over all predicted pixels—i.e. the number of pixels in an image times the number of colour channels times the number of examples (or, more usually, the mini-batch size). However,

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$$

As a consequence

$$\sum_{\boldsymbol{x}\in\mathcal{X}} \mathbb{E}_{\boldsymbol{z}}[\log(p_\theta(\boldsymbol{x}|\boldsymbol{z}))] = -\frac{N}{2} - \frac{N}{2}\log(2\pi\sigma^2)$$

so that we should minimise $N\log(\sigma^2)/2$. In information theory terms this tells us that it is cheaper to communicate the residues if they are more tightly concentrated. We note that

$$\frac{\partial}{\partial \hat{x}_i}\frac{N}{2}\log(2\pi\sigma^2) = \frac{\hat{x}_i - x_i}{\sigma^2}$$

which is precisely the gradient of

$$\sum_{i=1}^{N}\frac{(x_i - \hat{x}_i)^2}{2\sigma^2}$$

if we ignored the dependence of σ^2 on $\hat{x}_i$. In many publically available implementations of VAEs the algorithm minimises $\sum_{i=1}^{N}(x_i - \hat{x}_i)^2$ which arbitrarily assumes $\sigma^2 = \frac{1}{2}$ rather than its true value. This setup is acceptable but it means that these implementations are effectively running a β-VAE with some unknown β. This makes comparing results from different VAE implementations difficult. In the following sessions we use the empirical variance as a default experiment set if we did not clarify that we use a constant variance.

1.2 Generative Adversarial Networks

Another popular generative model is the GAN, which was first proposed by Goodfellow et al. (2014). Different from VAEs, the GAN is an implicit generative model and it cannot infer representation from the input. However, the advantage of GAN is it can generate high-quality and sharp images. There are two networks of the GAN which is the same as the VAE. We build two networks to play a zero-sum game. One is called the discriminator and another one is called the generator. The discriminator is trained to classify an image whether it is real or unreal, while the generator is trained to generate images from random noise to fool the discriminator to make wrong predictions. The competition between the two models drive them to improve themselves. We define D as the discriminator and G as the generator, where they are represented by differentiable fucntions such as an MLP or a CNN. Then we define a prior on the random noise $p_{\boldsymbol{z}}$. $G(\boldsymbol{z})$ is a mapping from the prior to the data space $p_{data}(\boldsymbol{x})$. And $D(\boldsymbol{x})$ is the probability that $\boldsymbol{x}$ is real data. Then objective function of the GAN is the following equation:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{1.4}$$

The objective is to maximise V(D, G) with respect to D and minimise V(D, G) with respect to G. For a fixed generator, the optimal D is

$$D_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{(p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x}))} \tag{1.5}$$

when $p_{data} = p_g$, where $p_g(\boldsymbol{x})$ is the distribution of the generator over data $\boldsymbol{x}$. The advantage of GAN is that it can generate very sharp images and there is no inference during the training which makes the cost of computation cheaper. However, the GAN is also notoriously hard to train. First, the discriminator can easily distinguish the real and fake images at an early stage. Then the gradient decreases to 0 in this situation. However, the generator can also fool the discriminator with the same image or a subset of images for all inputs, which is called mode collapse.

1.3 Priors for Representation Learning

As mentioned before, the performance of deep learning methods depends on the representation. With different designs, we can obtain a different kind of representation from the same input. Rather than designed representation for specific tasks, we are more interested in the generic representation, which is more convenient to express the world. There are several common priors that are widely acknowledged (Bengio et al., 2013):

- The first is the smoothness, similar inputs should return similar representations. However, this is hard to achieve since a small difference between representations can grow exponentially when the representations reverse to raw data, especially when the dimension is large.

- The second is explanatory factors. Data from the real world consist of different factors. Rather than represented by a whole latent code, we prefer the data be represented by a set of codes while each code corresponds to an explanatory factor. For example, for an image with a bird in the sky, a representation with explanatory factors means we can find a certain part in the representation that represents the bird and sky separately. Moreover, the representation can be hierarchical. The representation of the bird in the image can be decomposed into other properties of the bird, such as the class and the colour of the bird. This prior is the main direction of our research and it still has a long way to go in the community.
- The third is natural clustering. The variations of representation on the manifold should correspond to a categorical variable. More precisely, for images of digital number 0 to 9, the representation of 1 tend to be well separated from 3 but close to 7.

There are some other priors that are not mentioned here which are out of our research. Our work in representation learning highly relies on VAE.

1.4 Motivations and Contributions

A Generative model can learn a joint distribution over all variables. The ultimate goal of generative models is to improve the performance on downstream tasks. There is still a long way to go. To achieve this, we need to get a deeper understanding of these models and find more possibilities based on different frameworks. Rather than being used on downstream tasks, our research can also be useful for the application like photo editing or background editing during an online meeting. For example, the factorised representation can be controlled by a tool bar in an application to edit the existed photo. There are already many works in this field which will be discussed in the next chapter.

It is unclear whether deep generative models can understand the image as a human does or not. When we use a random sample as input to the decoder (or generator), deep generative models tend to generate meaningless images without supervision especially for natural images datasets like CIFAR10 or ImageNet. It is easy for models to generate images that mix different creatures and non-creatures at random positions in an image frame, we do not want an image that shows a cat with wings or a car with legs. A highly factorised latent space enables us to control the generation with random samples while the original VAE and GAN cannot. The hope is generative models can understand images as a human does. One of our expectations for generative models like VAEs and GANs is to generate high quality and unseen meaningful images. Also, such a factorised representation from the latent space can be generalised to other tasks. We start to investigate this problem by visualizing the latent space of a VAE to figure out whether it captures information that corresponds to human knowledge.

A prominent issue of VAE is the blurriness of the generated images, which makes the output of the decoder become useless when the input image is complicated (such as CIFAR10 or ImageNet). We believe that the blurriness is not a weakness that needs to be fixed, it is what the VAE is designed to get when we add the KL-term. In contract, what we need to do for VAE is to utilise the advantage of VAE which is the representation learnt by the model. So we proposed a tool to visualize the latent space learnt by VAE, it helps us understand what VAE learns from the dataset qualitatively and proves that we can gain useful representation from this latent space. We call this tool *latent space renderer*-GAN (or LSR-GAN) and will discuss more in Chapter 3.

When we realize the ability of VAE to learn a global representation, it is natural to think about whether it can be applied to other datasets. This kind of problem is called transfer learning. It is useful when the training data is not enough for the target problem and can also reduce the computational cost. Also, the pre-training model can help us get a better ELBO. In Chapter 4, we explore the possibility of VAE to do transfer learning.

Talking about generating images, it is always attractive to manipulate the generation. Obtaining factorised representation is a direction to achieve manipulation and also enables the generation of unseen meaningful images. A factorised representation means we can change one attribute of the image without influencing other attributes when we modify the latent space, each channel is sensitive to only one attribute. We propose a model to learn a factorised representation between the foreground and background of an image. More specifically, it learns a representation that factorises the objects and background in an image. Moreover, we demonstrate that our model is capable to factorise other kinds of attributes such as font style and category. This model is discussed in Chapter 5.

This work is published:

Compositing Foreground and Background Via Variational Autoencoders.

Accepted in the 3rd International Conference on Pattern Recognition and Artificial Intelligence.

The aforementioned model is not perfect. Although it can factorise the representation of the foreground and background, it treats all the foreground objects as a whole rather than a combination of separate objects, which makes the model restricted. However, the background and foreground information are still general conceptions while it can still be subdivided into small parts. For example, an image shows many people playing beach volleyball on the beach. Then the background can be subdivided into the sea and the sand, even the net, and the foreground can be subdivided into multiple people. Moreover, each person has his location, own height and own style. So we developed an algorithm that is able to learn object-oriented representation from multi-object images. Understanding realistic scenes as objects is a core ability of human perception and form the foundation of human's imagination. Objects form the basis of humans' high-level cognition (Spelke, 1990). Thus, a model that can learn object-oriented representation is consistent with how humans understand the world. Our model is based on a simple observation that the high error area of an image after transmitting usually belongs to objects. In the experiment, we can use the VAE or other compression methods to simulate the transmitting procedure, this gives us an error map with high error in the foreground area. By restricting the capacity of the bottleneck, Our model can learn the representation that factorised location, scale and appearance of an object among multiple objects. This model is discussed in Chapter 6

This work is published:

Unsupervised Representation Learning Via Information Compression.

Accepted in the 3rd the International Conference on Pattern Recognition and Artificial Intelligence.

Chapter 2

Literature Review

In this chapter, we review the development based on VAE and GAN models in three parts: VAE extensions, GAN extensions and VAE&GAN hybrids. Some of the extensions are not only focused on image generation tasks. For VAE extensions, we review the improvements on the latent space of VAE models, the VAE models on disentanglement, the works on object-centric representation learning and other related applications of VAE models. For GAN extensions, we first review the work on stabilizing the training of GAN and then review conditional GANs which can generate meaningful images. We also review GAN extensions on disentanglement. In the end, we review the hybrid work of VAE&GAN.

2.1 VAE Extensions

As a deep generative model, the VAE is widely used to represent high-dimensional data by a low dimensional representation in an unsupervised manner. The VAE can be viewed as two parts with independent parameters, the encoder and the decoder. The encoder is used to learn meaningful representation from data while the decoder is approximately the inverse process. Unlike the ordinary Variational Inference (VI) that each data point has a separate variational distribution, the encoder applies only one set of parameters to infer the distribution of the whole dataset. So the encoder of the VAE can be used for large datasets easily. The most important contribution of the VAE is the reparameterization trick. By introducing a noise term sampled from the Normal distribution, it makes the whole system differentiable. Kingma et al. (2014) also explored the VAE framework in a semi-supervised manner which performs well in classification tasks. Since the advantages of the VAE model, there is a series of papers showing up that extended the original VAE framework.

Here we introduce those extensions in three different directions, the first is the extensions that aim at a better latent space of the VAE, the second is the extensions that work on disentanglement and the third is the extensions that work on object-centric representation learning. In the last part, we review other applications of VAE models.

2.1.1 Better Latent Space

One type of VAE extension is in improving the latent space, this type of work aims at a better performance on density estimation and more informative latent space. One advantage of the latent space learnt by VAEs is it is easy to be transferred to other datasets, as we explore in Chapter 4. Thus, a more informative latent space would be beneficial to transfer learning and other downstream tasks.

Learning useful representations in an unsupervised manner is always a challenge in the deep learning field. However, when we adopt a powerful recurrent neural network (RNN) decoder in the VAE or the latent space with a high capacity, the latent code $\boldsymbol{z}$ is likely to be ignored and the model will learn a distribution that does not depend on $\boldsymbol{z}$ (Bowman et al., 2015). This is categorised as an optimization challenges of the VAE and also called "posterior collapse". At the early stage, it is easy for the model to set the posterior to be the same as the prior when the latent code carries little information. This avoids the KL cost.

The solution Bowman et al. (2015) proposed is to force the information into the latent space by weakening the decoder via dropout. While Gulrajani et al. (2016) combines VAE with PixelCNN (Van Oord et al., 2016) that is able to generate high-quality images and still capture non-trivial representations. Yang et al. (2017b) make an assumption that it is crucial to impose the prior on all the hidden states of the RNN-based VAE model. Thus, they propose holistic regularisation which averages the KL term for each state to solve the KL vanish issue. Dieng et al. (2018) proposed a new generative skip model, by adding a skip connection between the latent space and the decoder to promote higher mutual information between the latent code and the data. Alemi et al. (2018) start from analysing the ELBO in an information-theoretic view and derive a rate-distortion curve that shows the trade-off between compression and reconstruction accuracy. By choosing reasonable parameters according to the curve, they claim it can prevent the powerful decoder from ignoring the latent code.

Another type of extension is using the flow-based method on the latent code (Sønderby et al., 2016; Nielsen et al., 2020; Vahdat and Kautz, 2020). One popular flow-based method is Normalizing Flow (NF) (Rezende and Mohamed, 2015). The idea behind NF is to start with an initial random sample $\boldsymbol{z}_0$ from a distribution with a known function, and then update $\boldsymbol{z}_0$ with a transformation function $\boldsymbol{f}_t$, where t is the iteration index, then $\boldsymbol{z}_t$ is

$$\boldsymbol{z}_t = \boldsymbol{f}_t(\boldsymbol{z}_{t-1}, \boldsymbol{x}) \tag{2.1}$$

where $\boldsymbol{x}$ is the input data. The last iteration $\boldsymbol{z}_t$ can have a more flexible distribution. This flow cannot scale well to high-dimensional space since it needs a long chain of invertible transformation to capture dependencies in high-dimensional latent space, and it is also time-consuming. Nevertheless, Chen et al. (2016b) observed that a simple autoregressive flow can obtain a more expressive model. The latent code $\boldsymbol{z}$ is from a continuous noise $\boldsymbol{\epsilon}$ with transformation $\boldsymbol{f}$:

$$\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{\epsilon}) \tag{2.2}$$

Also, this model is trained in a lossy fashion to solve the information loss in the latent space. They construct a decoder with a small local receptive field to only store the unimportant information like texture, this results in a case that the decoder is not able to represent $p(\boldsymbol{x}|\boldsymbol{z})$ over $\boldsymbol{x}$ without the dependency on $\boldsymbol{z}$. The idea behind this method is still to restrict the information that can be stored in the decoder.

Autoregrssive flow is another type of flow-based method, Kingma et al. (2016) proposed inverse autoregressive flow (IAF). Unlike the NF that takes the $\boldsymbol{z}_{t-1}$ and $\boldsymbol{x}$ as the input for computing the following $\boldsymbol{z}_t$, the $\boldsymbol{z}_t$ only depends on previous $\boldsymbol{z}_{1:t-1}$ in IAF. In IAF, the autoregressive transformation takes $\boldsymbol{\mu}_0$ and $\boldsymbol{\sigma}_0$ as the initial state, with a random sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$, it has $\boldsymbol{z}_0 = \boldsymbol{\mu}_0 + \boldsymbol{\sigma}_0 \odot \boldsymbol{\epsilon}$, and the following transformation is:

$$\boldsymbol{z}_t = \boldsymbol{\mu}_t + \boldsymbol{\sigma}_t \odot \boldsymbol{z}_{t-1} \tag{2.3}$$

where at each step t, there is an autoregressive neural network that takes $\boldsymbol{z}_{t-1}$, $\boldsymbol{\sigma}_{t-1}$, $\boldsymbol{\mu}_{t-1}$ and a hidden state $\boldsymbol{h}$ from the network as the input to output $\boldsymbol{\sigma}_t$ and

$\boldsymbol{\mu}_t$. The autoregressive neural network is structured to output triangular matrix which also ease the computation difficulty and can scale well to high-dimensional latent space. The vanilla VAE can be regarded as a special version of IAF when we take just one step iteration with a linear transformation.

Compared to latent-variable VAE models, the disadvantage of flow-based models is the sequential operation takes expensive computation power.

One method for improving the latent space of VAE models is introducing auxiliary latent variables. Generally, this method adds a continuous auxiliary variable $\boldsymbol{u}$ to both the inference model and the generative model of a VAE. Then the encoder can infer a distribution that can be factorized as:

$$q_\phi\left(\boldsymbol{z}, \boldsymbol{u} | \boldsymbol{x}\right) = q_\phi\left(\boldsymbol{u} | \boldsymbol{x}\right) q_\phi\left(\boldsymbol{z} | \boldsymbol{x}, \boldsymbol{u}\right). \tag{2.4}$$

Where $\boldsymbol{x}$ is the input and $\boldsymbol{z}$ is the latent code. The decoder is over the joint distribution that can be factorized as :

$$p_\theta\left(\boldsymbol{z}, \boldsymbol{u}, \boldsymbol{x}\right) = p_\theta\left(\boldsymbol{u} | \boldsymbol{x}, \boldsymbol{z}\right) p_\theta\left(\boldsymbol{z}, \boldsymbol{x}\right). \tag{2.5}$$

The auxiliary variable has been shown that it can improve the flexibility of the latent distributions and lead to a better performance (Salimans et al., 2015; Maaløe et al., 2016; Ranganath et al., 2016).

In addition, some works aim at modifying the prior assumption (Dilokthanakul et al., 2016; Jiang et al., 2017; Tomczak and Welling, 2018; Takahashi et al., 2019) to improve density estimation results, since Hoffman and Johnson (2016) show that priors are an important factor in density estimation. Although Normal distribution is a common choice, it can also result in over-regularization, which returns in poor performance on density estimation. These works extend the VAE framework with other types of priors such as mixture distributions. One intuitive advantage of mixture distributions is the learned representation is natural clustering well since the multi-modal distribution can fit complex datasets better than one-modal distribution. However, the number of mixture distributions is a hyperparameter that needs to be tuned. Nevertheless, Dai and Wipf (2019) argued the standard Gaussian prior is not the reason that the vanilla VAE gets poor performance. Based on their proof, they propose a Two-Stage VAE which can achieve a Fréchet inception distance (FID) score (Heusel et al., 2017) that is comparable to GAN type models.

Unlike the aforementioned models that are all trying to learn a continuous representation, Van den Oord et al. (2017) Vector Quantised-Variational AutoEncoder(VQ-VAE) applied vector quantisation to the latent space to obtain a discrete latent representation. The prior distribution over the latent space is a catrgorical distribution rather than a Gaussian distribution. Each latent code in the latent space of VQ-VAE is mapped to the nearest embedding in the embedding space, then the input of the decoder is structured from the embedding space. Since there is no gradient for the mapping operation, the author just copies gradients from the decoder input to the encoder input. One alternative solution is to use the subgradient through the vector quantisation operation. It also achieves a decent performance on video and audio data. Although VQ-VAE is inspired by VAE, it does not train the model on the ELBO. And the following work VQ-VAE2 (Razavi et al., 2019) applies a two-stage training strategy to obtain hierarchical representations. Rather than using a single embedding space, the author adopts two embedding spaces with different sizes. The top embedding with a smaller size models the global information and the bottom embedding with larger size models the local details. It also works on high-fidelity images such as 1024×1024 human face, while most of the models tend to get degenerated results.

2.1.2 Disentanglement with VAE

Multiple explanatory factors and a hierarchical organization of explanatory factors are two of the general-purpose priors in the representation learning field Bengio et al. (2013). Such a factorised representation can also be called disentangled representation. Moreover, Higgins et al. (2018) define a disentangled representation as if it can be decomposed into a number of subspaces, each subspace is compatible with and can be transformed independently by a unique symmetry transformation, where symmetry transformation means to change one factor while keeping other factors invariant. And those disentangled representations have been shown to be beneficial to downstream tasks Higgins et al. (2017b).

With the recent development in deep generative models like the VAE, there has been a lot of extensions of VAE-based models for learning disentangled representation. Since the essential objective of generative models is to capture the underlying data distribution.

The VAE extensions that work on disentanglement can be divided into supervised methods and unsupervised methods. Supervision such as class labels allows us to obtain a factorised representation easily. While there are also semi-supervised models that can factorize the class information and other information (Cheung et al., 2014; Louizos et al., 2015). Although Cheung et al. (2014) propose a model based on a vanilla autoencoder rather than VAE. And the VFAE (Louizos et al., 2015) can also be trained unsupervised, it introduces a sensitive variable $\boldsymbol{s}$ and the encoder and the decoder of the VFAE accept $\boldsymbol{s}$ and $\boldsymbol{x}$, $\boldsymbol{s}$ and $\boldsymbol{z}$ as input respectively, which is slightly different from introducing auxiliary latent variables. By forcing the mutual information between $\boldsymbol{s}$ and $\boldsymbol{z}$ to be minimal to encourage independence between factors.

One line of supervised models achieve a factorised latent space by utilising a discriminator (Mathieu et al., 2016; Makhzani et al., 2016; Szabó et al., 2017; Xiao et al., 2017), where the discriminator is used to distinguish if different observations own the same property. The general idea of this type of work is to swap a certain part of latent codes between two inputs from different or the same category and force the category of the generated output to depend on the swapped latent code via a discriminator. This kind of method is capable of generating unseen combinations. ML-VAE (Bouchacourt et al., 2018) is another VAE-based model that requires weak supervision. In their work, they propose group-level supervision that each group of observations share a common but unknown value for a factor. The group supervision can help the model to anchor the semantics into the latent representation. Esser et al. (2018) combine the U-Net (Ronneberger et al., 2015) with a VAE for mapping the input shape such as edges to the target images and using one extra encoder to learn a latent representation for appearance. This approach enables both conditional generation and style transformation. While Harsh Jha et al. (2018) use a pair of inputs, for each pair, they use a different image with the same label and they also swap the factors during training. Apart from swapping the latent factors, they also feed the reconstructions back to the encoder to obtain a cycle-consistent architecture. By forcing the encoder to output the same factor with both the original images and the reconstructions, it enhances the performance of the model.

After Higgins et al. (2017a) showed an objective function of the VAE models with a β higher than 1 can achieve disentanglement in the latent space, there have been a series of works on unsupervised disentanglement. With the derivation from

Hoffman and Johnson (2016):

$$\begin{aligned}\beta\mathbb{E}_{p_{data}(\boldsymbol{x})}[D_{KL}\big(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)] &= \beta\mathbb{E}_{p_{data}(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{z})}\right] \\ &= \beta\mathbb{E}_{p_{data}(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{z}|\boldsymbol{x})q(\boldsymbol{z})}{q(\boldsymbol{z})p(\boldsymbol{z})}\right] \\ &= \beta\mathbb{E}_{p_{data}(\boldsymbol{x})}\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log\frac{q(\boldsymbol{z}|\boldsymbol{x})}{q(\boldsymbol{z})}+\log\frac{q(\boldsymbol{z})}{p(\boldsymbol{z})}\right] \\ &= \beta I_q(\boldsymbol{x},\boldsymbol{z})+\beta D_{KL}\big(q(\boldsymbol{z})||p(\boldsymbol{z})\big)\end{aligned} \tag{2.6}$$

It shows a better interpretation of the trade-off between disentanglement and better density estimation by tuning the value of β. Penalising the $I_q(\boldsymbol{x},\boldsymbol{z})$ leads to a less informative latent space that results in a poor reconstruction. While penalising $D_{KL}\big(q(\boldsymbol{z})||p(\boldsymbol{z})\big)$ forces $q(\boldsymbol{z})$ to $p(\boldsymbol{z})$, since the prior $p(\boldsymbol{z})$ is commonly defined as Normal distribution, it encourages independence among variables in $\boldsymbol{z}$. A higher β actually penalises the two terms results in the trade-off.

Then a group of works aims at achieving the disentanglement while remains a good density estimation (Zhao et al., 2017; Kim and Mnih, 2018; Chen et al., 2018; Esmaeili et al., 2019). Although the two terms in Equation 2.6 cannot be calculated separately due to intractability. They augment the ELBO with a third term:

$$D_{KL}\left(q(\boldsymbol{z})||\prod_j q(\boldsymbol{z}_j)\right) \tag{2.7}$$

Which is also known as Total Correlation (TC) (Watanabe, 1960), measures the information shared among the latent code. The difference between these works is the method they sample the $q(\boldsymbol{z}_j)$. The disadvantage of these unsupervised models is that the semantic meaning of each disentangled latent code is unknown without testing manually. Also, it is unsure if each unit of latent codes control one factor or more than one unit controls one factor when there are more latent codes than properties of the data.

There are also methods that aims at solving specific task rather than general purpose. For example, Yang and Yao (2019) propose dVAE that disentangled the pose, viewpoint and image content of hand images. And they demonstrate the dVAE is capable of generating high-quality synthesised hand images with specific sampling.

2.1.3 Object-centric Representation Learning

Recently, there is another line of work focus on unsupervised object-centric representation learning and scene decomposition. Human brains can process the realistic scene as a whole and decompose it into different parts using visual clues and prior knowledge. This cognitive ability also enables humans to imagine different scenes. Objects form the basis of humans' high-level cognition (Spelke, 1990). Thus, learning good object representations could be an important step towards making artificial intelligence closer to human intelligence. Learning structured representation from a multi-object scene without supervision is a challenging task in the deep learning community. While supervised methods can detect objects easily, it is usually too costly to collect labels for all the real scenes. A VAE is an unsupervised model that can learn a compact representation from data. However, VAEs are designed to learn the data as a whole but not a combination of different parts. Those aforementioned works are all for single-object scenes, we take one more step from these works, focusing on unsupervised multi-object representation learning. The ability of decomposing multi-object scenes into object representations is important to high-level artificial intelligence. We explore this task in Chapter 6, however, our motivation is to transmit an image efficiently while our model can also learn the object-centric representation at the same time.

Recently, there are a bunch of works in unsupervised multi-object representation learning. And there are two lines of the work: scene-mixture models and spatial-attention models (Lin et al., 2020). For scene-mixture models, the multi-object scene is represented as a mixture of multiple single-object or non-object scenes (background) (Greff et al., 2017; Burgess et al., 2019; Engelcke et al., 2020, 2021; Locatello et al., 2020). This type of model can produce a segmentation mask of objects and generate a full-scale image of a single object while the disentangled representation of positions and scales have been neglected in this manner. Also, these scene-mixture models ask recurrent learning for each scene, which means the following segmentation masks are always depending on the previous mask. Thus, most scene-mixture models need RNN networks. And the negative log-likelihood in the objective function of the vanilla VAE is turned into a form:

$$\log\left(\sum_{k=1}^{K}\boldsymbol{m}_k p(\boldsymbol{x}|\boldsymbol{z}_k)\right) \tag{2.8}$$

Where $\boldsymbol{m}_k$ is the mask learned by the model and $\sum_{k=1}^{K} \boldsymbol{m}_k = 1$, the sum of $\boldsymbol{m}_k$ equals to 1 makes sure that the output reconstruct the whole image. $\boldsymbol{z}_k$ is the latent code of the corresponding kth object. Moreover, adopting spatial broadcast decoder (Watters et al., 2019) is a common choice in these models. Compared to the vanilla decoder, spatial broadcast decoder consists of multiple layers of CNN with 1x1 stride, and it takes 2 more channels that represent Cartesian coordinates, which is aimed at providing an architectural prior for disentangling the positional feature from the non-positional feature in the latent space. The one-dimension latent code produced by the encoder will be tiled into the same size as the input. The spatial broadcast decoder has been demonstrated is beneficial to disentangling, reconstruction accuracy, and to out-of-distribution generalization.

The MONet (Burgess et al., 2019) applies a recurrent attention mechanism that produces deterministic soft masks for each component in the scene with a attention mechanism that follows a VAE to learn the desired representation. For all images, the model iterates K times where K is the maximum number of objects in the dataset. However, MONet adopts U-Net as the backbone of the attention mechanism which increases the computation cost. Also, the iterative training scheme for all images is time-consuming. Different from MONet, IODINE (Greff et al., 2019) employs an iterative refinement mechanism for the latent code, which is computationally expensive. Rather than produce a representation for one object each iteration, IODINE produces the representation code for all objects and refines those representations for N iterations, where N is a hyperparameter. And the performance of IODINE is highly related to the value of N. Genesis (Engelcke et al., 2020) is similar to MONet, replacing the attention mechanism with an encoder and employing an RNN network after the encoder to infer masks of objects, then following another VAE to infer the object representations. Unlike MONet and IODINE, all the modules in Genesis can be processed in parallel, which can save time compared to iterative training. Genesis-V2 (Engelcke et al., 2021) is the upgraded version of Genesis which replaces the RNN network with a semi-convolution embedding method. Similar to MONet, it uses a U-Net encoder as the backbone which enhance the model's ability in extracting object representation.

Other models that combine with the self-attention mechanism (Vaswani et al., 2017) can also perform image decomposition and object representation learning (van Steenkiste et al., 2020; Locatello et al., 2020).

Unlike scene-mixture models that only learn an entangled representation for each object, spatial-attention models can learn an explicit disentangled representation about positions or scales through spatial transform network (Eslami et al., 2016; Crawford and Pineau, 2019; Deng et al., 2021; Lin et al., 2020; Zhu et al., 2021). However, this type of model adopts the spatial transform network (STN) (Jaderberg et al., 2015) which can only produce rectangular bounding boxes rather than segmentation masks. Also, the model is less accurate when the dataset contains multiple objects with different scales of size since the original STN is not designed to solve the multi-object image tasks. Originally, Eslami et al. (2016) and Crawford and Pineau (2019) are both trained sequentially, but according to Lin et al. (2020), these models can all be trained in a parallel manner.

Attend-Infer-Repeat (AIR) (Eslami et al., 2016) is the first work that tries to infer the object representation as "what", "where" and "pres" variables. The "what" variable represents the shape and appearance of objects, the "where" variable represents the position and scale of objects and the "pres" variable is a unary code which formed by 1 or 0, where 0 means the termination of inference. The following work of AIR is Spatially Invariant Attend, Infer, Repeat (SPAIR) (Crawford and Pineau, 2019), it introduces another variable "depth" which specify the relative depth of objects. Also, the authors of SPAIR replace the RNN network in the AIR with the CNN network and switch the "pres" from a unary code to a binary variable that can be sampled from a Bernoulli distribution, while the Gumbel-Softmax (Jang et al., 2016) allows us to differentiable sample from such a discrete distribution. Then this "pres" variable starts to represent the presence of objects in cells. The image is encoded as a feature map, and each cell in the feature map is processed with the nearby cells that have already been processed to produce objects. Thus, the whole process is sequential. The two above models both ignore the background part in the image. SPACE (Lin et al., 2020) applies a similar approach to the SPAIR to learn the object-centric representation and add another background network that is close to Genesis to infer the background components. When the number of background components is one, the background network can be a vanilla VAE. Also, SPACE processes all the cells fully parallel which saves time with GPU.

The aforementioned works all succeed in learning object-centric representation. GSGN (Deng et al., 2021) takes one step further in that it can decompose an object into more parts to learn a hierarchical representation, but it asks a more complicated architecture of the model while it applies SPACE as a backbone to learn a decomposition of images.

2.1.4 Other applications

As a deep generative model, the VAE is not only used for common images, there are many applications that apply the VAE into other types of data. One example is that the VAE trained on chemical molecule structures (Gómez-Bombarelli et al., 2018), they build a VAE to learn a continuous representation of molecules that enables search of new molecules with desired properties. Apart from chemistry, the VAE has also been applied to biology in different ways. Way et al. (2020) test VAE on compress gene expression data and notice that it is better to combine more than one compression method to learn the representation of gene data. And Seninge et al. (2021) propose a new architecture of the VAE model which is inspired by biology insights and their new model is aimed at an explanatory biological model for drug treatment experiments.

Ravanbakhsh et al. (2017) applies the VAE to galaxy images. Since the cosmological surveys asks for an accurate shape measurement of distant galaxies, and the measurement usually relies on a calibration that requires large sets of galaxy images, which is limited in real world. They propose to use deep generative models such as VAEs to generate synthesized galaxy images.

Videos can be regarded as a series of sequential images, however, videos can express spatio-temporal pattern which images cannot. In videos, the motions of each element needs to be coherent and plausible over time, which requires model to learn a long dependency over each frame in videos. There are a bunch of VAE-based models that can generate consistent video frame and can even be used in prediction task (Finn et al., 2016; Babaeizadeh et al., 2017; He et al., 2018).

2.2 GAN Extensions

The GAN model is a generative model that is trained via an adversarial process, where a generator and a discriminator are trained separately. The generator is trained to capture the data distribution and the discriminator is trained to classify whether a sample is from the real data rather than the generator. The task of the generator is to fool the discriminator to make a wrong prediction. The original GAN model is an implicit generative model which can only generate samples from a prior distribution but cannot represent the distribution of data. But it is capable of generating sharp images compared to VAEs. The GAN model is notoriously on the training stability since the GAN model is similar to a minimax two-player game, the generator and the discriminator can easily find a solution at the beginning where the generator produces random noises for all the input and the discriminator predicts all the output from the generator as fake, rather than a discriminator that predicts $\frac{1}{2}$ everywhere. Here we first review the literature on stabilising and enhancing the vanilla GAN model.

2.2.1 Stabilising the Training of GANs

The DCGAN (Radford et al., 2015) introduces a general CNN architecture of GAN types network that is more stable and stronger than the original GAN and becomes the common default setting of most following research. The original GAN model and even the original VAE model are all tested on MLP architecture. CNNs are more often used in supervised learning. The DCGAN bridges the gap between the CNNs for supervised learning and unsupervised learning, the guideline of the architecture can be summarised as follows:

- Replace all pooling layers with stride CNNs or transposed CNNs.
- Use batch normalization (Ioffe and Szegedy, 2015).
- Remove MLP layers for deeper architectures.
- Use ReLU activation (Nair and Hinton, 2010) in the generator for all layers except for the last layer, which uses Tanh.
- Use LeakyReLU activation (Xu et al., 2015) in the discriminator for all layers.

The DCGAN also shows a meaningful interpretation with arithmetic operations on semantic attributes.

The GANs are not only unstable with training, but also easily suffer from the mode collapse, where the generator is stuck at a local minimum and fools the discriminator with one type or a subset of real data distribution. There is a series of literature about stabilizing the training of GAN and fixing the mode collapse issue. To avoid the situation that the generator is not synchronized with the discriminator, one intuitive solution is to make the minimax two-player game imbalanced. Unrolled GAN (Metz et al., 2016) fixes the stabilisation issue by training the model in an unrolled manner, the GAN plays the two-player game for K times and only update the discriminator at the first step. However, it applies the gradient descent at each step to optimise a new discriminator. Then the generator is updated through the new discriminator. Thus, the generator depends on what the new discriminator will return. This imbalanced game reduces the probability that the generator collapse to a point. Then Heusel et al. (2017) introduced a two-time scale update rule which uses different learning rates for the generator and discriminator which improves the convergence. The Wasserstein GAN (Arjovsky et al., 2017) shows that setting a Lipschitz constraint on the weights of networks can improve the training procedure while it assures the discriminator is less sensitive to the small change of inputs. Unlike Kullback–Leibler divergence and Jensen-Shannon divergence that the value cannot reflect the real distance between two distributions when there is no overlap between two distributions, the Wasserstein distance provides a continuous function that makes the gradient more stable. Wasserstein GAN with gradient penalty (WGAN-GP) (Gulrajani et al., 2017) improves the stability of Wasserstein GAN by introducing a new weight clipping method. Rather than the Wasserstein distance, Mao et al. (2017) propose to use the least square loss function to replace the sigmoid cross-entropy loss function for the discriminator, which can also stabilise the training. Miyato et al. (2018) propose spectral normalization to enforce Lipschitz constraints in the network which can both stabilize the training and solve the mode collapse problem. And SA-GAN (Zhang et al., 2019) applies a self-attention mechanism (Cheng et al., 2016) to achieve a long-range dependency modelling. They succeed in generating details for high-resolution images. Recently, Jiang et al. (2021) propose a new GAN model that is based on pure transformer modules (Vaswani et al., 2017) rather than CNNs, which helps the model capture semantic contexts and low-level textures.

In addition, introducing a VAE to the GAN model can also stabilise the training of the GAN, which has been demonstrated in the literature of VAE&GAN Hybrids (See details in section 2.3 and Chapter 3).

2.2.2 Conditional GANs

GANs are prone to generate sharp but meaningless images when the data is complicated and diverse. An alternative method to solve this issue is to add extra information during the training, such as text or class labels. Conditional GANs are a type of GANs that use conditional information for the discriminator and generator, and this type of GANs is capable of class conditional image generation. Reed et al. (2016) concatenate a representation code from detailed text descriptions with both the input of the generator and a lower dimension feature vector from the discriminator, the resulting model can generate plausible bird and flower images based on the detailed text descriptions. Odena et al. (2017) add an extra one-hot vector to noise vectors before feeding the vector into the generator and add an auxiliary classifier to a GAN model to make sure the class of the generated image corresponds to the extra one-hot vector. This architecture guarantees the class of generated images which can avoid generating meaningless images. Similar to Reed et al. (2016), Miyato and Koyama (2018) add label information to feature vectors instead of noise vectors, where the feature vectors are the inputs of the last layer. However, rather than concatenating the vector of label information with the feature vectors, the authors adopt inner product of the two vectors as the additional term of the objective function, which shows an improvement of the quality of the class conditional image generation experimentally. BigGAN (Brock et al., 2018) is another model trained with label information, it is the first model that adopts a large number of parameters and large batch size, which is expensive to be trained. However, it demonstrates incredible qualitative and quantitative results on high-resolution images. And the following work BiBigGAN (Donahue and Simonyan, 2019) shows improved representation learning performance by adding an encoder to the model and also modifying the discriminator to take both the images and the corresponding latent code as the input.

Another type of conditional GAN is aimed at image-to-image translation: StarGAN (Choi et al., 2018) is a unified model that can perform image-to-image translations for multiple domains. The discriminator of StarGAN classifies the image as fake or real and also classifies the corresponding domain, while the generator takes the original image and the target domain as the input to generate the fake image or takes the original image and the original domain as the input to reconstruct the original image. They demonstrate effective results on human face translation. The following work StarGAN v2 tackles the issue that the previous work cannot scale well to more domains and improve the performance by adding more domain vectors and a cycle consistency loss. The StackGAN (Zhang et al., 2017) is another work that can generate high-quality images based-on text description by stack the representation from text and image. Rather than modify both the generator and the discriminator of GANs, StyleGAN (Karras et al., 2019) only modifies the generator, by adding adaptive instance normalization (Huang and Belongie, 2017) after each CNN layer of the generator. Before feeding the noise vector into the generator, they use a MLP to map the noise vector into an intermediate latent space that controls the style. After this, the same authors revisit the generator normalization method and redesign the network to improve the performance (Karras et al., 2020).

2.2.3 Disentanglment with GANs

Although the GAN is an implicit generative model, it can still learn a disentangled latent space even in an unsupervised manner. InfoGAN (Chen et al., 2016a) augments the latent code $\boldsymbol{z}$ with an additional code $\boldsymbol{c}$ (which can be categorical code), in the original GAN framework, the generator tends to ignore the additional code $\boldsymbol{c}$. InfoGAN forces the mutual information between $\boldsymbol{c}$ and $\boldsymbol{x}$ to make sure the latent code $\boldsymbol{c}$ is representative and it results in a disentangled latent space. As demonstrated in the paper, InfoGAN learns interpretable representations that are competitive with supervised methods. OOGAN (Liu et al., 2020) follows the strategy of InfoGAN and the authors notice that in contrast to the VAEs that the latent vector is inferred by the encoder, the latent vector in GAN is sampled manually. Thus, they design the additional code $\boldsymbol{c}$, to possess the property of inter-dimensional independence, and train the network with those designed latent vectors. Rather than utilising mutual information to achieve a disentangled latent space, (Pan et al., 2021) propose contrastive disentanglement in GANs (CD-GAN). Contrastive learning aims at maximizing the similarity of positive pairs and minimizing it between negative pairs. Thus, this CD-GAN requires a small number of supervisions and can disentangle the factors of inter-class variation of data.

Also, there is a line of work that focuses on the factorisation of the background and the foreground. This is a task we explore in Chapter 5 although our architecture is based on VAEs rather than GANs, which brings an advantage of our model is to infer the representation from existing data. These models generate foreground and background separately and recursively, the generated images are stitched at the final stages to produce a complete natural image (Yang et al., 2017a; Singh et al., 2019; Li et al., 2020). In LR-GAN (Yang et al., 2017a), there is a LSTM layer that connects the background generator and the foreground generator. Intuitively, this LSTM provides information to the foreground generator about which part has been generated in the past. Similar to LR-GAN, FineGAN (Singh et al., 2019) generate the background first, rather than using a LSTM layer, the authors share the same latent code between both the background generator and the foreground generator. At each stage, the shared latent code will be concatenated with the background code and the foreground code. Unlike the LR-GAN that is fully unsupervised, the FineGAN asks for a bounding boxes and the number of categories for training. Both LR-GAN and FineGAN are unable to factorise the background and the foreground of an existed image since the two models do not have an encoder. MixNMatch (Li et al., 2020) is the following work of FineGAN and the author improved the network by adding additional encoders, thus, the MixNMatch can encode real data into discrete codes or feature maps by different settings.

2.3 VAE&GAN Hybrids

VAEs can infer an approximation probability distribution of data but generate blurry images while GANs can provide sharp generation but cannot infer from existing data. The intuitive thought to merge the advantages of the two models is to design a VAE&GAN hybrid. We also introduce a new hybrid model in Chapter 3. Rather than merge the advantages of the two models, our motivation is to visualise the latent space of the vanilla VAE. Also, our hybrid model is not end-to-end and asks for a pre-trained and frozen VAE. Here we review some main hybrids that are based on the vanilla VAE and GAN.

The adversarial types autoencoder is the most intuitive and simplest way to combine a VAE or an autoencoder and GAN models. Most of these models introduce a discriminator into the autoencoder training. VAE/GAN applies a discriminator trained with reconstructed images, generated images and real images (Larsen et al., 2015). And they replace element-wise errors with feature-wise errors for a better-learned distribution with translation invariance, where the feature comes from the lth layer of the discriminator. And they show that the model can learn a latent space that captures high-level visual features which can be modified using simple arithmetic. Makhzani et al. (2016) apply a discriminator to distinguish the output of the encoder and the random sample from the prior distribution. It uses the discriminator to replace the KL term in the loss function of the VAE. MDGAN (Che et al., 2017) is another hybrid that is close to VAE/GAN. The authors argue that the unstable training of GANs and the mode collapse issue are due to the very particular functional shape of the discriminator in high dimensional spaces, which can easily allow training to become stuck. Thus, they try to match the manifold of the GAN to real data by treating the objective function of the vanilla VAE as a regularization term to penalise the mode collapse issue.

IAN (Brock et al., 2017) is a unified model which means the discriminator is not separate and the discriminator is updated with the generator. The authors use the discriminator of the GAN as a feature extractor of the encoder, thus, the encoder is built on the top of the last layer of the discriminator. Also, the model adopts feature-wise loss rather than element-wise loss. The generator accepts both noise vectors and the output of the encoder as inputs to generate different images. And they extend this model into an application called Neural Photo Editor. Srivastava et al. (2017) introduces a network F_θ to the GAN. The task of F_θ is to map both the real images and synthetic images to a Gaussian distribution which is the same as an encoder network. When the input of F_θ is the output of the generator, the loss function is the MSE error between the input of the generator and the output of F_θ. If the input of F_θ is real data, the loss function is a cross-entropy between Gaussian prior and the output of F_θ. This design can resist mode collapse than other GAN variants and can produce more realistic samples.

2.4 Summary

Currently, high-quality image generation can be achieved by different models, which indicates that those models can learn a representation for the whole image of the dataset efficiently. However, for VAE-based models, we argue that the advantage is to learn representations rather than generate high-quality images, also, an entangled representation is not efficient for the downstream task and is also not controllable for the generation task. Thus, a model that can learn a factorised representation of the dataset is still needed. Many previous works focus on the factorisation for different attributes of the object. However, we claim that the factorisation between the foreground and the background is needed before the factorisation for attributes. Thus, we start from this point and manage to factorise the foreground and the background in the latent space, then we focus on a deep-level factorisation for the foreground which can also include the previous works. This flow is more natural as it is from shallow to deep.

Chapter 3

Imagining the Latent Space of a Variational Autoencoder

The VAE is a powerful method for representation learning and the decoder of a VAE can visualize the latent space in a human-understand manner (as images). But the output of the decoder is too blurry to be recognized for complex data. Thus, we introduce a tool that can reconstruct sharp images from VAE's latent space.

3.1 Introduction

VAEs have made a significant impact since their introduction by Kingma and Welling (2013). However, one of their perceived problems is their blurriness. This has spawned a wave of research into trying to improve the reconstruction performance (Larsen et al., 2015; Dai and Wipf, 2019; Vahdat and Kautz, 2020). We argue that such attempts are misguided. The advantage of VAEs is to capture a useful representation of the dataset but not sharp reconstructions or generations. This is a consequence of the well-known *evidence lower bound* or ELBO objective function consisting of a negative log-probability of generating the original image from the latent representation (this is often implemented as a mean squared error between the image and the reconstruction, although as we argue in Chapter 1 this term originally should be proportional to the logarithm of the mean squared error) and a KL-divergence between the probability distribution representing a latent code and a 'prior distribution' (usually taken as a multivariate normal with mean zero and unit variance). These two terms have a nice interpretation in terms of the *minimum description length* (Rissanen, 1978)—this has been described elsewhere, for example, Chen et al. (2016b). The KL-term can be viewed as a measure of the amount of information in the latent code while the log-probability of the image measures the amount of information required to change the image produced by the decoder into the input image (see Section 3.3 for details).

The great strength of a VAE is that it builds a model of the dataset that does not over-fit (i.e. code for detailed features found in specific images). However, because of this it typically will not do a good job of reconstructing images as the latent code does not contain enough information to do the reconstruction (for very restrictive datasets such as MNIST and Celeb-A a lot of information can be captured in the latent space, but for more complex datasets like ImageNet or CIFAR the reconstructions are poor). Of course, if you want good reconstructions on the dataset then the simplest solution is to remove the KL-divergence term and just use an autoencoder. While the KL-term forces $q(\boldsymbol{z}|\boldsymbol{x})$ to the prior $p(\boldsymbol{z})$, thus the learned representation will not retain all the information of the data. If we wish to generate realistic-looking images we need to imagine the information discarded by the encoder. As a rather simplified analogy, consider a verbal description of an image "a five-year-old girl in a blue dress standing on a beach". If we asked different artists to depict such a scene there is clearly not enough information to provide pixel-wise or feature-wise similarity between their interpretation although each artist could render a convincing image that satisfies the description. Similarly, if we want a VAE to generate realistic images we need to build a renderer that will imagine an image consistent with the latent variable representation.

A simple way to achieve this is using a modified Generative Adversarial Network (GAN). We call such a model a *latent space renderer*-GAN (or LSR-GAN). To generate an image we choose a latent vector $\boldsymbol{z}$ from the prior distribution of the VAE. This is passed to a generator network that generates an image, $\hat{\boldsymbol{x}}$, with the same dimensions as the latent space of a pre-trained VAE. The generated image has both to convince a discriminator network that it is a real image—as is usual for a GAN (Goodfellow et al., 2014)—at the same time the representation of $\hat{\boldsymbol{x}}$ obtained from the VAE encoder should match $\boldsymbol{z}$. To accomplish this we simply add an augment term to the normal GAN loss function for the generator (L_G)

$$L_G - \lambda \log(q_\phi(\boldsymbol{z}|\hat{\boldsymbol{x}})) \tag{3.1}$$

where $q_{\boldsymbol{\phi}}(\cdot|\hat{\boldsymbol{x}})$ is the probability distribution generated by the VAE encoder given an image $\hat{\boldsymbol{x}}$ and $\boldsymbol{z}$ is the latent vector that was fed into the GAN generator. Note that when training the LSR-GAN we freeze the weights of the pre-trained VAE encoder. The constant λ is an adjustable hyperparameter providing a trade-off between how realistic the image should look and how closely it captures the information in the latent space. This modification of the objective function can clearly be applied to any GAN or used with any VAE. Although the idea is simple, it provides a powerful method for visualising (imagining) the information stored in a latent space. By matching the latent space of a GAN to a pre-trained VAE, also avoids the GAN from mode collapse.

Combinations of VAEs and GANs are, of course, not new (Larsen et al., 2015; Makhzani et al., 2016; Brock et al., 2017; Huang et al., 2018; Srivastava et al., 2017). In all cases we are aware of GANs have been combined with VAEs to "correct" for the poor reconstruction performance of the VAE or help GANs gain approximate distributions of data. As we have argued (and expound on in more details in Section 3.3), we believe that the decoder of a VAE does the job it is designed to do. They cannot reconstruct sharp images, because the latent space of a VAE loses information about the image, by design. All we can do is imagine the type of image that a point in the latent space represents.

In the next section, we show examples of images generated by the LSR-GAN for both normal VAEs and β-VAEs (we also spend time describing VAEs, β-VAEs and the LSR-GAN in more detail). In addition, we also present systematic experiments showing the performance of a VAE and LSR-GAN. In Section 3.3, we revisit the minimum description length formalism to explain why we believe a VAE is doomed to fail as a generative model. In Section 3.4 we draw out the similarities and differences between our approach to hybridising VAEs with GANs and other work in this area. We present some additional experimental results in Section 3.5. We conclude in Section 3.6.

3.2 Imagining Latent Spaces

A natural question to ask about the VAE is what information about an image gets represented in the latent space. However, this is hard to be measured quantitatively since there are different criteria for different kinds of information. Thus, we propose LSR-GAN which can help us answer this question qualitatively, in our experiment, the VAE tends to reserve information like shape, texture and colour for big object. The procedure is similar to the reconstruction of the original VAE, however, we replace the decoder with a generator. By feeding the generator with a latent code obtained from the pre-trained VAE encoder, we can obtain sharp reconstructions (although the reconstructions losts some details). We show examples of this for both CIFAR-10 (Krizhevsky et al., 2009) and ImageNet (down-sampled to 64×64) (Krizhevsky et al., 2012) dataset. We also show the effect of different choices on the β value (in Equation 1.3) when we pre-train the VAE. In all cases showed in this chapter, the input images are taken from a test set that is independent of the training set. Note that both CIFAR-10 and ImageNet are "complex" for VAEs in the sense that they represent extremely diverse sets of images while we ask the VAEs to learn a generic latent space. As a consequence, the VAE latent space will struggle to store detailed information about the images and the VAE reconstructions will be poor.

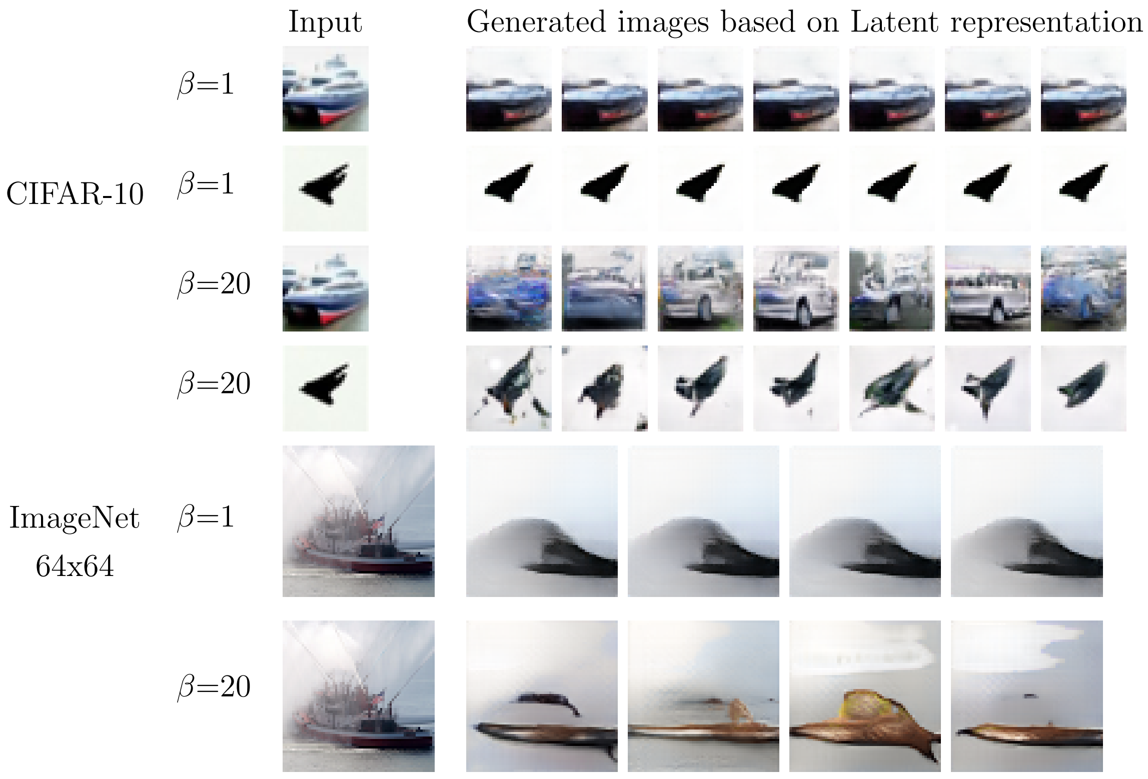


Figure 3.1: Images generated by our LSR-GAN from latent representations of an input image (test images) based on different β value on CIFAR-10 and ImageNet, the first column is the original image.

To get a sense of the variation in the information stored in latent spaces we show in Figure 3.1, where the left image is the input and right images are the outputs generated by the LSR-GAN generator seeded with a latent vector encoding of the input image. For the CIFAR-10 dataset, we choose a boat and a bird image, the VAE can learn an informative latent space to a decent degree, and the performance gets degenerated with a higher $\boldsymbol{\beta}$. While for the ImageNet dataset, we still choose a boat image. The number of categories is much higher than CIFAR-10, thus, the VAE cannot really preserve many details for a single image. The reconstructions capture the shape and background, but clearly loses a lot of details. In some cases it appears that the type of object is being captured, although in the case of the boat with the β-VAE (with $\beta = 20$) the wrong object is being rendered. We note that there is very little variation between the different samples drawn from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, particularly for the standard VAE ($\beta = 1$), showing that the latent space of the VAE is relatively smooth (there is more variation when $\beta = 20$).

3.2.1 LSR-GAN

Before introducing the LSR-GAN, we represent the structure of a VAE schematically below. We sample an input $\boldsymbol{x}$ from some dataset, $\mathcal{D}$. To be concrete we will consider the case where the inputs are images. Although clearly a VAE can be used to represent many different types of data, it is mainly used on images.

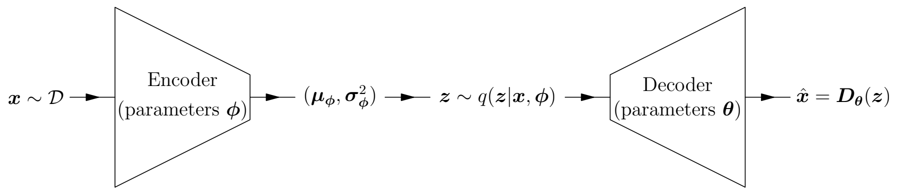

$\boldsymbol{x} \sim \mathcal{D}$
Encoder
(parameters $\boldsymbol{\phi}$)
$(\boldsymbol{\mu}_\phi, \boldsymbol{\sigma}^2_\phi)$
$\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\phi})$
Decoder
(parameters $\boldsymbol{\theta}$)
$\hat{\boldsymbol{x}} = \boldsymbol{D}_{\boldsymbol{\theta}}(\boldsymbol{z})$

Figure 3.2: The diagram of a vanilla VAE.

For each input $\boldsymbol{x}$ the encoder outputs a mean vector, $\boldsymbol{\mu}$, and standard deviation vector, $\boldsymbol{\sigma}$, that describes an axis aligned normal distribution, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}\big|\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \operatorname{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})))$. A latent variable $\boldsymbol{z}$ is sampled from this distribution and then fed to a decoder. For simple black and white datasets such as MNIST the decoder outputs a scalar at each pixel location that can be interpreted as the probability that the pixel is black. For more complex datasets the decoder usually generates a "reconstruction" $\hat{\boldsymbol{x}}$. The probability of generating a pixel value x_i is then usually taken as a normal distribution with mean $\hat{x}_i$ (i.e. $p_{\boldsymbol{\theta}}(x_i|\boldsymbol{z}) = \mathcal{N}(x_i|\hat{x}_i, \sigma^2)$) and variance σ^2 that measures the expected size of the errors between the input images, $\boldsymbol{x}$, and the reconstructions, $\hat{\boldsymbol{x}}$.

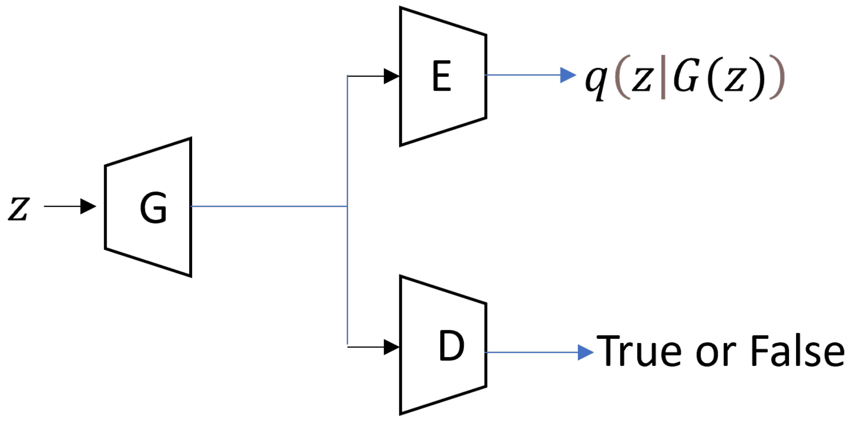


Figure 3.3: The diagram of LSR-GAN.

LSR-GAN is a novel hybridization of the VAE and GAN model. The most distinct difference of LSR-GAN from previous work is that it is a two-stage model. In the first stage we train the VAE model. Having done this we freeze the weights of the VAE and train the GAN. We train the discriminator, D, of LSR-GAN in the same way as a normal GAN. That is, we minimise a loss function

$$\mathcal{L}_{\text{D}} = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \tag{3.2}$$

where G is the generator of LSR-GAN. The job of the discriminator, D is, to decide whether its input is a real image or not. Thus, to optimise the loss function we need to maximize the log-probability of passing the real data, $\boldsymbol{x}$, while minimising the log-probability of accepting a random sampling $G(\boldsymbol{z})$ generated by a generator G seeded with a random latent vector $\boldsymbol{z}$. The architecture of the generator is the same as that of a normal GAN but the loss function is slightly different. We add an additional term giving

$$\mathcal{L}_{\text{G}} = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log D(G(\boldsymbol{z}))] + \lambda \log(q_{\phi}(\boldsymbol{z}|G(\boldsymbol{z}))). \tag{3.3}$$

The parameters of the discriminator and generator are trained in the usual tick-tock fashion using gradient descent. We built the VAE and the generator of GAN using a ResNet (He et al., 2016) as it gave slightly better performance than using a standard CNN. The architecture of the discriminator is the same as DCGAN (Radford et al., 2015), and the diagram is shown in Figure 3.3. Figure 3.4 shows the reconstruction results of two datasets. For each pair in the figure, the left one is the original image and the right one is the reconstruction image from LSR-GAN. Although the reconstructions are sharp enough, those images do not retain enough details. The details vary in different images, some outputs can reconstruct very detailed information like the bird in ImageNet, and some reconstructions are extremely poor like the grass and flower in ImageNet.

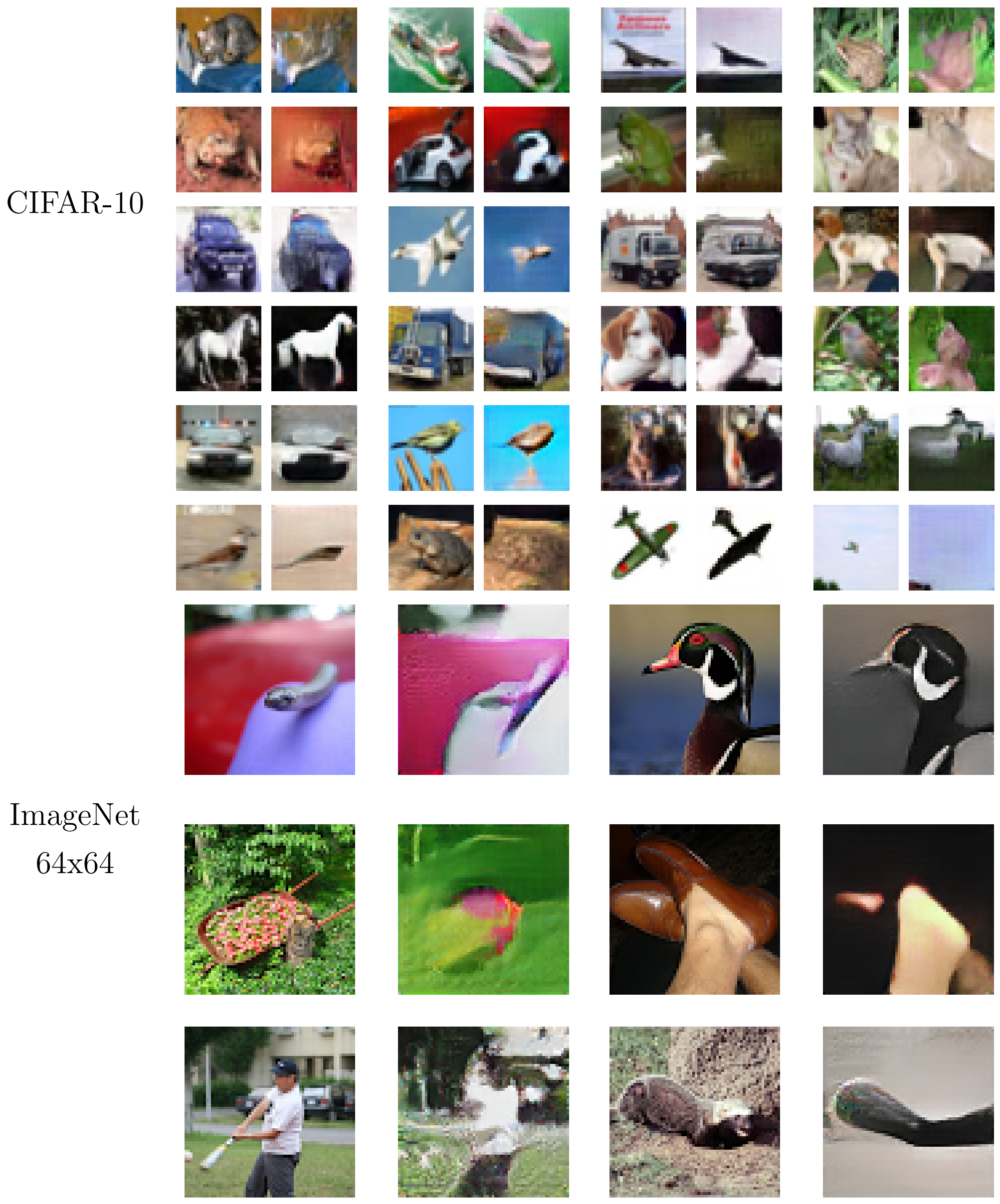


Figure 3.4: Examples of input-output pairs of images. The left image is an image from the test dataset. The right image is the image generated by the LSR-GAN seeded with a latent vector encoding of the input image ($\beta = 1$)

To test the LSR-GAN we use the VAE to generate a latent representation $\boldsymbol{z}$ for an image drawn from an independent test set. The latent vector is then used as a seed value for the generator in the LSR-GAN. The LSR-GAN can get sharper reconstruction images than the VAE (see Figure 3.5). Although not visually so

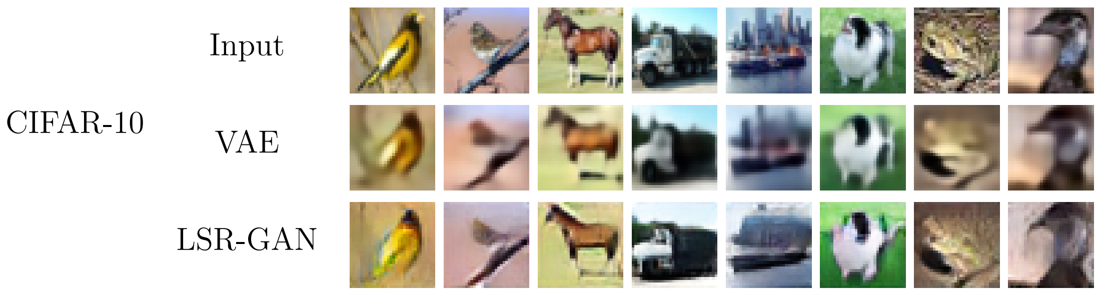


Figure 3.5: The comparison between reconstruction images of VAE and LSR-GAN on test dataset.

obvious, we have used a quantitative measure of sharpness computed as luminance-normalised Laplacian (San Pedro and Siersdorfer, 2009, Section 3.1.2). Sharpness measures the level of detail of an image, San Pedro and Siersdorfer (2009) claim the sharpness of an image can be determined as a function of its Laplacian, normalized by the local average luminance around each pixel, as shown in Equation 3.4:

$$Sh = \sum_{x,y} \frac{L\left(x,y\right)}{\mu_{xy}}, with\, L\left(x,y\right) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{3.4}$$

where $\mu_{x,y}$ denotes the average luminance around pixel (x,y). While the luminance value is the images in the YUV colour space. Higher value means better sharpness. For the reconstructed images from the VAE we obtained a measure of 0.17 ± 0.03 while for the LSR-GAN we obtain 0.28 ± 0.08 (i.e. an improvement of a factor of two). We have also computed the Fréchet inception distance (FID) scores (Heusel et al., 2017) for CIFAR-10. Where FID is a metric that calculates the Fréchet distance between the fake images and the real images. as shown in Equation 3.5:

$$FID = \|\mu_1 - \mu_2\|_2^2 + tr\left(\Sigma_1 + \Sigma_2 - 2\left(\Sigma_1^{\frac{1}{2}}\Sigma_2\Sigma_1^{\frac{1}{2}}\right)^{\frac{1}{2}}\right) \tag{3.5}$$

Where (μ_1, Σ_1) and (μ_2, Σ_2) are the mean and covariance of fake images and real images respectively, which are obtained from the inception model (Szegedy et al., 2015). The lower distance score means better similarity. For images seeded from a testing example, the VAE achieved a score of 89.8 while LSR-GAN achieved a score of 35.9, while for images seeded with random latent variable (i.e. $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$) the FID score for the VAE is 138.6 while for the LSR-GAN it is 36.4. This should not be surprising. The decoder of the VAE is training only where there are training images. Although the VAE does not do too badly generating testing examples,

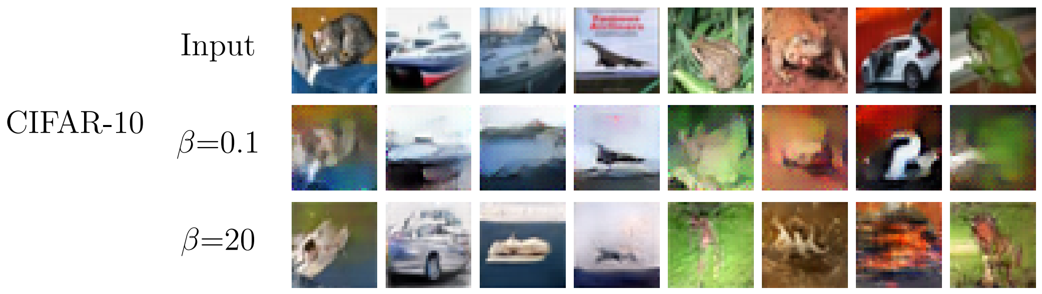


Figure 3.6: Examples of input-output pairs of images for different β values for a β-VAE. The first row is from the test dataset. The second and third rows are reconstructions generated by LSR-GAN for VAEs trained with different β values.

these tend to be substantially closer to the training examples than random samples in the latent space. In contrast, the LSR-GAN is trained on random samples so that the generator will have to produce "realistic" images over the whole latent space. Of course, whether these generated images represent anything recognisable is open to question. For diverse training sets such as CIFAR-10 and ImageNet this may be very difficult. What image should we expect from a latent vector halfway between a truck and a bird?

3.2.2 LSR-GAN with Beta-VAE

As the reconstruction highly depends on the latent space we obtained from the pre-trained VAE. A small change in the VAE can cause a different results. According to the β-VAE introduced by Higgins et al. (2017a), where the KL-divergence term in a normal VAE is weighted by a parameter β

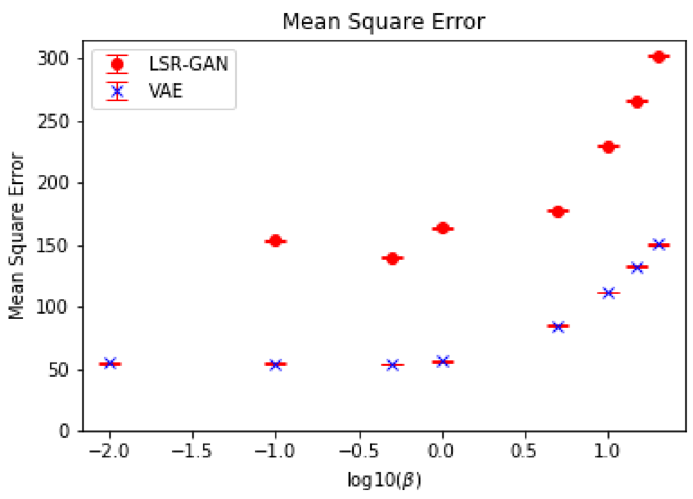
$$\mathcal{L} = -\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] + \beta D_{KL}\big(q(\boldsymbol{z}|\boldsymbol{x})||\mathcal{N}(\mathbf{0},\mathbf{I})\big). \tag{3.6}$$

The argument is that by making $\beta \gg 1$ we encourage disentanglement. Contrariwise, by making $\beta \ll 1$ we make a VAE closer to an auto-encoder.

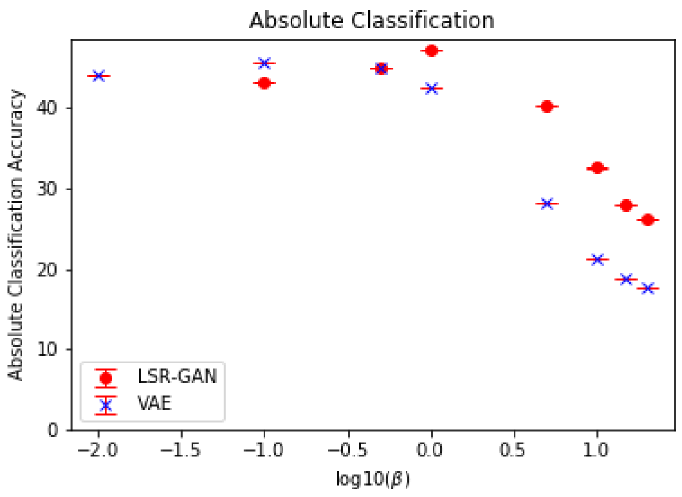
Thus, in Figure 3.6 we show more examples of input-output pairs from the LSR-GAN that the pre-trained VAE with different values of β. We observe that for large β the output images are quite different from the input images. In contrast, the output images of small β are more close to the original images. Also, those images in the third row lost more details compared to the second row, which is consistent with the theory that the VAE with a higher β value retains less information in the latent space.

Although the LSR-GAN model generates slightly clearer, less blurry, images, it has a higher reconstruction error than the VAE decoder. We show the mean squared error measured on the testing set from CIFAR-10 as a function of β in Figure 3.7(a). When the β is small enough the LSR-GAN just collapses. Because when the VAE is trained with a small β, it close to an autoencoder, which can only compress the input data rather than learn a proper distribution. Thus the LSR-GAN cannot match the sample to a reasonable distribution. The poor performance of the LSR-GAN on mean squared error (MSE) is unsurprising, it uses the same information as the VAE (i.e. the information stored in the latent space). By producing sharper images it will pay the price of getting the boundary wrong. The blurry edges from the VAE is a way to hedge its bet and reduced the mean squared error. Interestingly, the mean squared error remains fairly constant until we reach $\beta = 1$ after which it rapidly increases. One interpretation of this fact is that the VAE with $\beta = 1$ is successfully encoding all the useful information (i.e. compressible information) so for reconstructing unseen images it will perform as well as an auto-encoder. As we increase β above 1, the reconstruction error increases rapidly.

Rather than only look at the MSE between the reconstructions and the original images, we test the reconstructions by passing them to a simple classifier which consists of 2 layers of CNN and a layer of MLP. In Figure 3.7(b) we show the absolute classification performance of the LSR-GAN and the VAE, which means we only count the result as true when the reconstruction image is classified as the correct label. The classifier is trained on the original CIFAR-10 training set. The classifier performance achieved an 84% correct classification on the raw images. We find little variation as we decrease β smaller than 1. As we increase β above 1 the classification accuracy falls off. Again we can attribute this to the latent space of the VAE (with $\beta = 1$) capturing less information. Due to the small value of the error, the error bar is not obvious in the figure. Interestingly the high-β VAE fails to capture "objectness" well. This suggests that, at least for CIFAR-10, the type of object does not contain very much information about its appearance and is rapidly discarded.



(a)



(b)

Figure 3.7: Performance of the VAE (blue points) and LSR-GAN (red points) versus β. In (a) we show the mean squared error, while in (b) we show the classification performance using a classifier taking the reconstructed images. The images are taken from CIFAR-10. Error bars show standard error.

3.3 Minimum Description Length

To understand what VAEs do it is useful to interpret them in the framework of the minimum description length (MDL) formalism. In MDL we consider communicating a dataset $\mathcal{D}$ through a communication channel using as few bits as possible. We can do this using lossy compression, where we encode each input $\boldsymbol{x}$ by a code $\boldsymbol{z}$, which we communicate down our channel. The receiver decodes the message and produces an approximation of the input $\hat{\boldsymbol{x}}$. To communicate the original information we send the code $\boldsymbol{z}$ together with the error $\boldsymbol{\epsilon} = \boldsymbol{x} - \hat{\boldsymbol{x}}$ between the input $\boldsymbol{x}$ and the reconstruction $\hat{\boldsymbol{x}}$. The expected cost of transmitting an input is

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[M(\boldsymbol{z}) + E(\boldsymbol{\epsilon})]$$

where $M(\boldsymbol{z})$ is the number of bits needed to communicate the code, $\boldsymbol{z}$, and $E(\boldsymbol{\epsilon})$ is the number of bits required to communicate the error, $\boldsymbol{\epsilon}$. In the MDL formalism we attempt to find a code that minimises the description length $\mathcal{L}$. To communicate the model and errors we need to use an optimal coding strategy. Rather than specifier and actual code we can use the Shannon bound (i.e. the negative log-probability of the tokens we transmit). For this to be meaningful, we need to specify both the errors and code to a finite precision. The precision of the errors will determine the accuracy of the data we communicate. If the i^{th} component of the error is distributed according to $p(\epsilon_i)$ then the cost of communicating the error to a precision of Δ is approximately $-\log(p(\epsilon_i)\,\Delta) = -\log p(\epsilon_i) - \log(\Delta)$. The factor $-\log(\Delta)$ is common to all coding schemes so is irrelevant to choosing optimal codes $\boldsymbol{z}$. In contrast the precision to which we transmit the model will directly determine the cost $M(\boldsymbol{z})$. There is a balance to be struck: a more precise model can potential lead to a better reconstruction $\hat{\boldsymbol{x}}$, reducing the reconstruction cost, $E(\boldsymbol{\epsilon})$, but at the same time increasing the cost, $M(\boldsymbol{z})$, of communicating the code $\boldsymbol{z}$.

The KL-divergence term, $\mathrm{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}))$ (also known as the relative entropy) can be interrupted as the communication cost (in nats) of transmitting a random variable $\boldsymbol{z}$ with uncertainty given by $q(\boldsymbol{z})$ assuming an underlying probability distribution of all random variables of $p(\boldsymbol{z})$. Using this interpretation we see that the loss function of a VAE is equivalent to the expected message length (in nats) of communicating a sample from the dataset $\mathcal{D}$ by using a random variable $\boldsymbol{z}$ with uncertainty $q(\boldsymbol{z})$. By minimising the loss function we find a coding scheme with the minimum description length (or, at least, an approximate local minimum). By encoding a message as a random variable $\boldsymbol{z}$ drawn from a distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ the VAE is able to find an optimal balance between accuracy to which it transmits the model (determined by the standard deviation vector, $\boldsymbol{\sigma}$, generated by the VAE encoder) and the need to reduce the reconstruction error. *From an MDL perspective the ELBO is the correct objective function, and should not be regarded as a approximate lower bound to what we really want to achieve.* If there are too many dimensions in the latent space then some of the components of $\boldsymbol{z}$ (channel in information theory terms) are such that z_i is approximated distributed by $\mathcal{N}(z_i|0,1)$ for all inputs $\boldsymbol{x}$. The channel is effectively "switched off" (and it will be ignored by the decoder as it is just a source of random noise). This is referred to as posterior collapse and is sometimes viewed as problematic, however, from the MDL viewpoint it acts as an elegant automatic dimensionality selection technique.

The job of the decoder in a variational autoencoder is to reconstruct the image only using information that can be compressed. With an extremely powerful encoder and decoder and a limited dataset it would be possible for the encoder to communicate an identifier of the input image and for the decoder to reproduce the image just from the identifier, thus avoiding communicating any information about the visual content of the image—this requires that the decoder memorises all the images. This would be an extreme case of posterior collapse. There is some evidence that with very strong encoders and decoders that the amount of information stored in the latent space (as measured by the KL-divergence) decreases (Bowman et al., 2015). This might point to a weakness of the VAE set-up—the MDL set-up really only makes sense when the dataset is arbitrarily large—, but this problem could be ameliorated by data augmentation. However, using standard CNN encoders and decoders we found no evidence for memorisation of the images (for example, the VAE would produce a similar level of reconstruction for images from a separate test set). For language modelling there seems to be more evidence that VAEs often fail to extract information in the latent space, but for images it seems likely that a properly trained VAE will extract a good fraction of the information. As a consequence we should not think of the decoder of a VAE as a generative model: It will, by design, produce blurry and poor quality reconstructions. We see the mapping from images to latent space as a many-to-one mapping. Thus, the mapping from the latent space to images will be ambiguous and the best we can do is imagine an image compatible with the latent variable: exactly what we have designed the LSR-GAN to do.

3.4 Differences between LSR-GAN and Related Work

The hybridisation of VAE and GAN models have been developed for several years. But most of those works train the model in an end-to-end manner, like AAE (Makhzani et al., 2016), VAE/GAN (Larsen et al., 2015) and MDGAN (Che et al., 2017). None of these methods feeds the output of the generator back into the encoder or trains their network in two-stages, which is the biggest difference between these methods and ours. Also, some of these hybrid models adopt an autoencoder instead of VAE while the VAE in our model cannot be replaced by an autoencoder.

There are a few models that use the output of the decoder to feed the encoder. This type of method is so called "introspective". The Introspective Adversarial Network (IAN) (Brock et al., 2017) is a unified model which the discriminator is not separate. IAN encodes features from both raw images and synthetic images during the training. While our encoder only accept synthetic images. Another model that adopts the introspective method is IntroVAE (Huang et al., 2018), it constructs the inference model E and generator model G in a circulation loop. IntroVAE has the ability to generate high-resolution images. But it does not contain any discriminator network.

The most close work to our LSR-GAN is VEEGAN (Srivastava et al., 2017). It introduces a network F_θ to the GAN. The task of F_θ is the same as an encoder. The loss function is different when the input of F_θ changes. It minimises the MSE error between the input of the generator and the output of F_θ when the input is the output of the generator. If the input of F_θ is real data, The loss function is a cross-entropy between Gaussian prior and the output of F_θ. Another related model is the Generative moment matching networks (GMMN) (Li et al., 2015). In this model the autoencoder is frozen and they then minimize the maximum mean discrepancy (MMD) between the generated representation and data representation, and they use a uniform prior to generate the representations. In LSR-GAN, we match two Gaussian distributions by maximizing the probability. None of these related works are two-stages models except GMMN.

3.5 Experiments

In this section, we present some quantitative and qualitative results. Figure 3.8 describes the structure of the VAE's encoder and decoder, the GAN's generator. The discriminator is the same as the one used in DCGAN (Radford et al., 2015) The encoder and decoder/generator are based on ResNet. Both networks are optimized using Adam (Kingma and Ba, 2015) with a learning rate of 2×10^{-4} and $\beta_1 = 0.5$. The additional fully connected layer at the beginning of the generator is necessary as the latent variable of the encoder is from fully connected layers. It helps the initial random variable $\boldsymbol{z}$ be transformed into feature maps, which

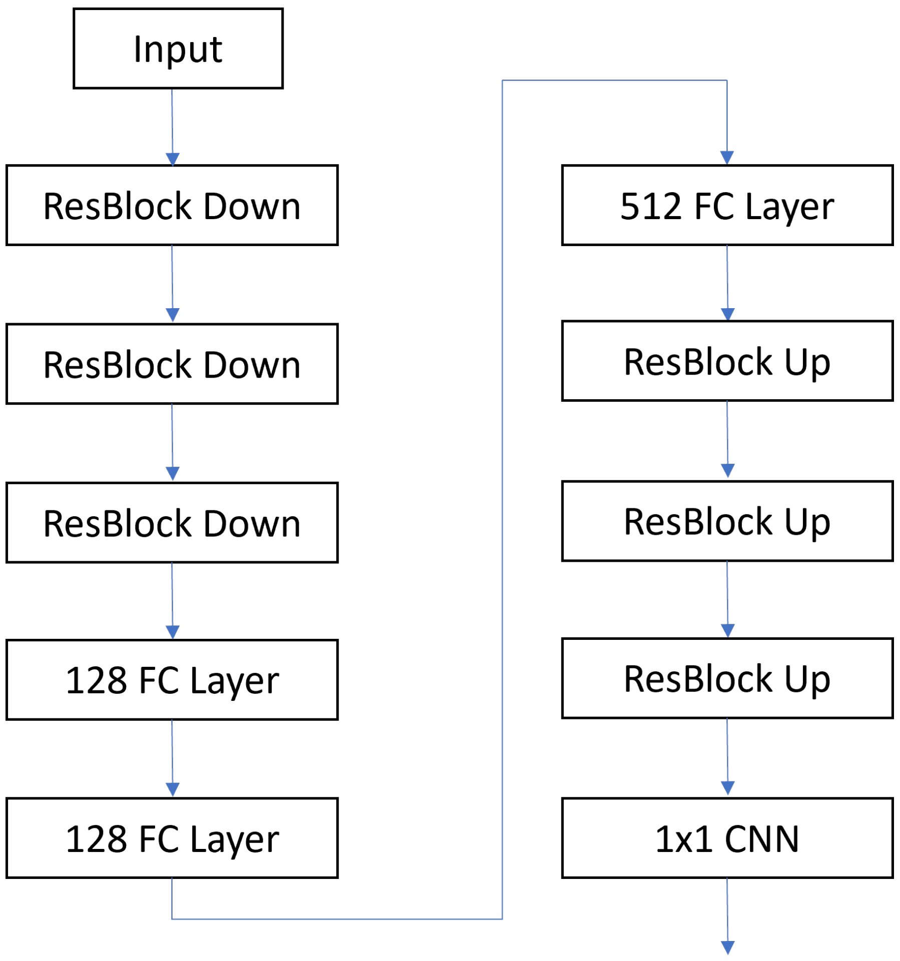


Figure 3.8: The architecture details.

can stabilize the training of GAN. We show the results on four different datasets: CIFAR-10, ImageNet, Celeb-A and MNIST. Most of the quantitative results are based on the CIFAR-10 dataset, which is a dataset that contains 10 different categories and the image scale is 32×32. While the ImageNet dataset contains more images and categories, the original size of the image varies, and we rescale all the images into 64×64. The Celeb-A is a dataset consist of different human faces, we also rescale the images to 64×64 and crop the centre part. The MNIST is a simple dataset contains binary images of handwritten digits from 0 to 9, the size is the smallest which is only 28×28.

3.5.1 Dependence of LSR-GAN on β

As mentioned before, the images generated from the LSR-GAN highly relies on the quality of the latent space obtained from the VAE. Thus, the choice of β value can bring different effects. In Table 3.1 we present measurements of the outputs from both VAEs and LSR-GAN that are trained on CIFAR-10 with different values of β. Some of this data is also presented graphically in Figure 3.7, but we have included additional measurements.

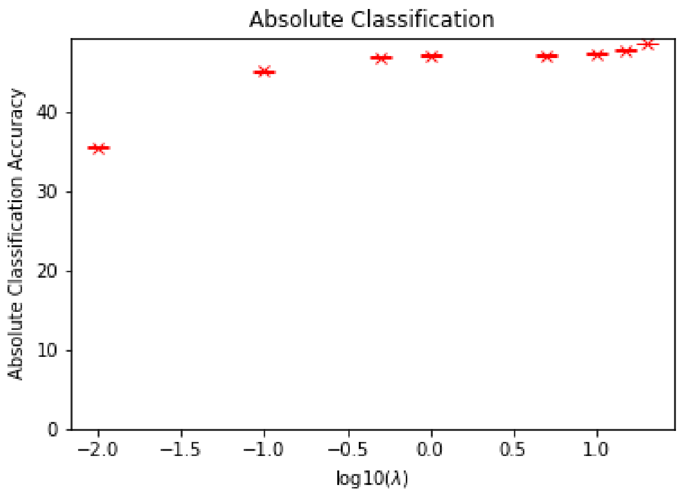
Table 3.1: The measurement for different β values. Absolute acc is the rate that classifier classifies reconstruction images right. Relative acc is the rate that classifier classifies reconstruction images the same as raw images.

	β=0.01	β=0.1	β=0.5	β=1	β=5	β=10	β=15	β=20
MSE (VAE)	54.56 ± 0.30	54.07 ± 0.29	53.80 ± 0.29	55.9 ± 0.3	84.64 ± 0.42	111.49 ± 0.53	132.56 ± 0.61	150.40 ± 0.66
MSE (LSR-GAN)	Model Collapse	153.09 ± 0.66	139.75 ± 0.63	163.06 ± 0.73	177.22 ± 0.86	229.53 ± 1.07	265.74 ± 1.19	302.17 ± 1.47
Absolute acc (VAE)	$44.08 \pm 0.22\%$	$45.66 \pm 0.21\%$	$45.01 \pm 0.17\%$	$42.56 \pm 0.13\%$	$28.17 \pm 0.13\%$	$21.26 \pm 0.20\%$	$18.89 \pm 0.18\%$	$17.70 \pm 0.23\%$
Absolute acc (LSR-GAN)	Model Collapse	$43.29 \pm 0.18\%$	$45.06 \pm 0.23\%$	$47.19 \pm 0.13\%$	$40.23 \pm 0.18\%$	$32.58 \pm 0.21\%$	$27.93 \pm 0.21\%$	$26.18 \pm 0.17\%$
Relative acc (VAE)	$45.69 \pm 0.21\%$	$44.11 \pm 0.20\%$	$43.38 \pm 0.21\%$	$43.75 \pm 0.25\%\%$	$28.34 \pm 0.15\%$	$21.59 \pm 0.21\%$	$18.90 \pm 0.18\%$	$16.90 \pm 0.21\%$
Relative acc (LSR-GAN)	Model Collapse	$45.02 \pm 0.18\%$	$44.71 \pm 0.16\%$	$48.41 \pm 0.25\%\%$	$43.54 \pm 0.18\%$	$37.90 \pm 0.17\%$	$33.68 \pm 0.20\%$	$31.91 \pm 0.19\%$

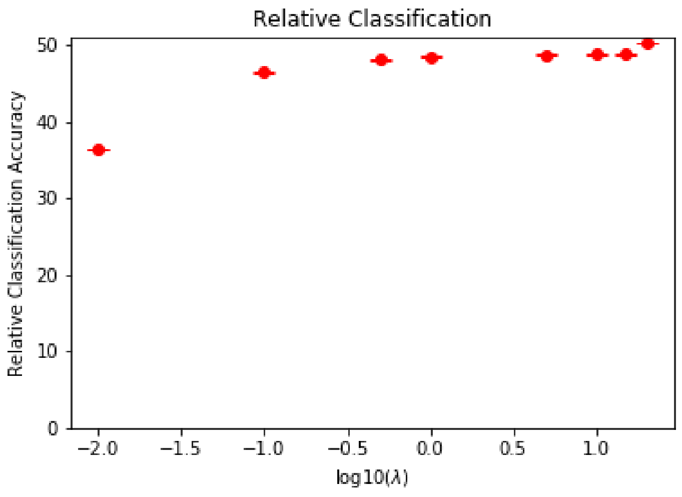
In addition to MSE, there are two different classification results in the table. Both results are from the same classifier which is trained on the original training set. The absolute classification accuracy shows the rate of how many reconstructions are classified as the true category among all images. While the relative classification shows the rate of how many outputs of reconstruction images are the same as outputs of original images among all images. The LSR-GAN collapses with $\beta = 0.01$, which has been explained before. As shown in the table, the VAE is better than the LSR-GAN in MSE comparison for all the β values. However, LSR-GAN is better than VAE in both absolute and relative classification results. This also proves that MSE is not a proper measure either of the image-quality or the information preservation. And the classification accuracy roughly decreases as the β value increases, this is acceptable because the latent space lost more information. Moreover, the classification results from the LSR-GAN is more stable than the VAE when the value of β increases. This shows with a higher β value, the decoder of the VAE gets degenerated when the latent space loses more information, while the generator of LSR-GAN can visualising the learned information much better than the decoder of the VAE. One thing in Table 3.1 that needs to be noticed is that the MSE of LSR-GAN when $\beta = 0.1$ is close to the MSE of VAE at $\beta = 20$. But the image we get from LSR-GAN is still much sharper than VAE and the classification accuracy based on LSR-GAN generation is also better. This also shows that the MSE can not help us get realistic images or informative images.

3.5.2 Dependence of LSR-GAN on λ

Unlike the β value of the pre-traiend VAE that affects the latent space. The hyper-parameter λ balances the need to produce convincing images (from the discriminator's point of view) with the requirement that the latent space of the GAN should be close to that for the VAE. These two objectives are not necessarily contradictory, although we will see that changing λ has benefits and drawbacks. The β value is the same for these experiments.

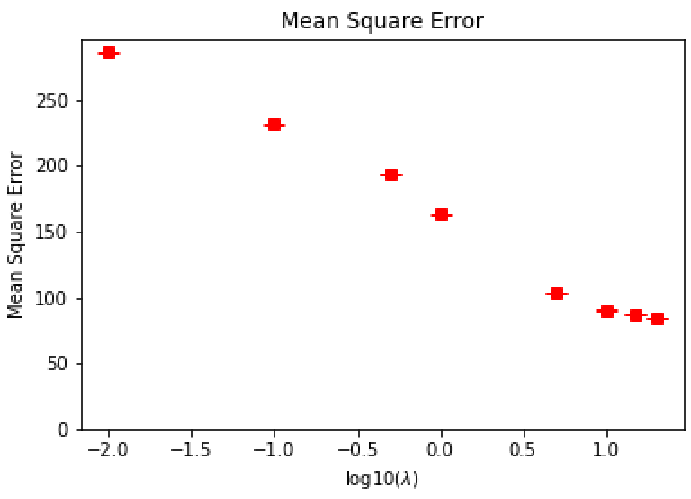


(a)

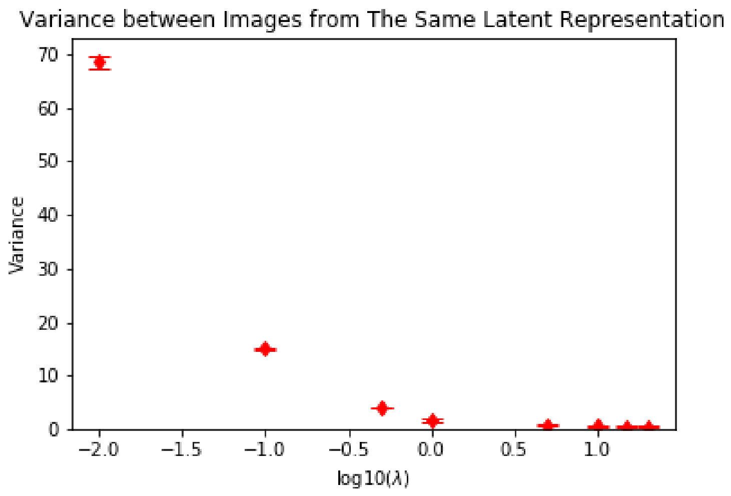


(b)

Figure 3.9: Graphs showing the classification performance of images generated by our LSR-GAN with different λ values (on test dataset).



(a)



(b)

Figure 3.10: Graphs showing the classification performance of images generated by our LSR-GAN with different λ values (on test dataset).

In Figure 3.9 we show the effect of changing λ over approximately three orders of magnitude. Figure 3.9a shows the absolute classification accuracy, Figure 3.9b shows the classification accuracy compared to the class labels predicted by the classifier on the raw images, Figure 3.10a shows the MSE value and Figure 3.10b shows the variance in the predictions when choosing different samples from $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. We take the logarithm based on 10 of those λ values since the gaps between those λ values are large. We see that increasing λ improves the classification performance (both relative and absolute). Also, increasing λ produces a significant reduction in the reconstruction error. These improvements all depends on the similarity of the latent space between the LSR-GAN and the pre-trained encoder. More intuitively, it also causes a reduction in the variance between images sampled independently from $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. That is, using the encoder in the LSR-GAN acts as a regulariser ensuring close by points in the latent space map to similar images.

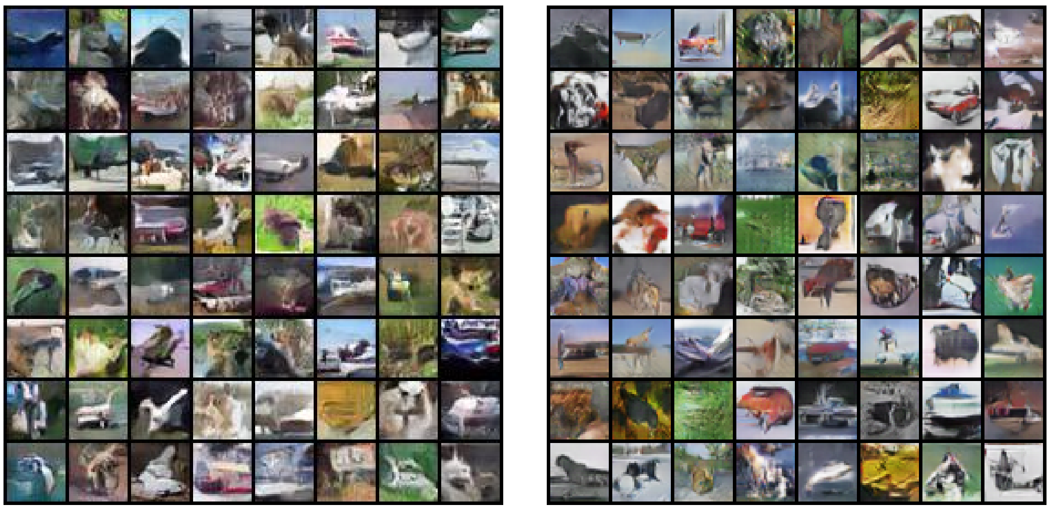
Although the graph can deliver an intuitive view of the improvement, it still lacks exact values. More details are given in Table 3.2. It can be observed that when λ is higher than 1, by forcing the matching between two distributions with a higher λ value, the results of classification accuracy get better than the original VAE. Even for the MSE, it shows a trend that the LSR-GAN can do better. However, the generator collapses when the λ is too high. Just like an very large β value of the VAE, such as 1000, will return a messed up latent space without any information for the dataset, a higher λ like 100 would result in a degenerated generator.

Table 3.2: The measurement for different λ values. Variance is the variance among the images generated by same latent representations. Absolute acc is the rate that classifier classifies reconstruction images right. Relative acc is the rate that classifier classifies reconstruction images the same as raw images.

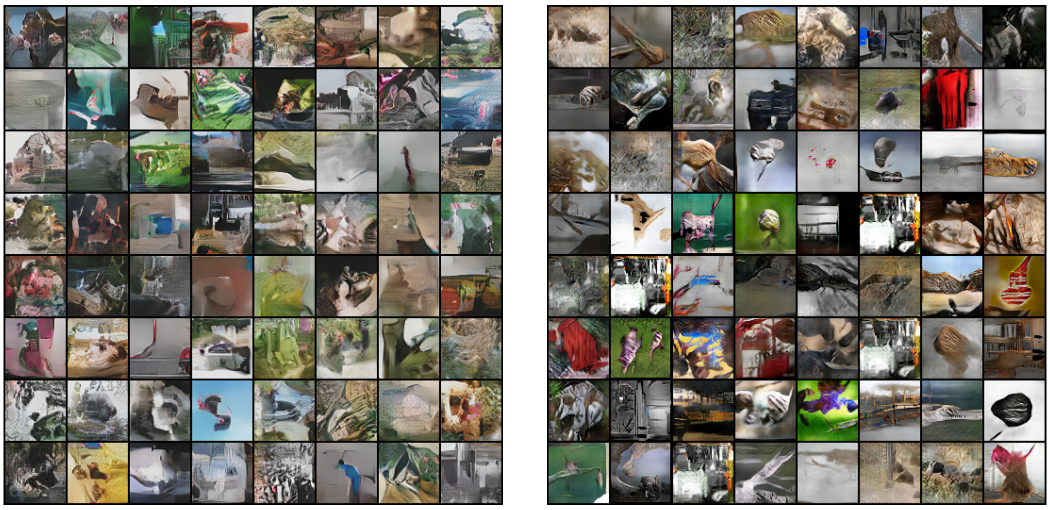
	λ=0.01	λ=0.1	λ=0.5	λ=1	λ=5	λ=10	λ=15	λ=20
MSE	285.99 ± 1.39	231.34 ± 1.14	193.72 ± 0.90	163.06 ± 0.73	104.15 ± 0.46	90.66 ± 0.43	86.81 ± 0.42	84.69 ± 0.40
Variance	68.54 ± 1.17	15.06 ± 0.25	3.96 ± 0.08	1.67 ± 0.02	0.70 ± 0.01	0.54 ± 0.00	0.50 ± 0.00	0.48 ± 0.00
Absolute acc	$35.52 \pm 0.23\%$	$45.2 \pm 0.21\%$	$46.96 \pm 0.28\%$	$47.19 \pm 0.13\%$	$47.21 \pm 0.20\%$	$47.45 \pm 0.18\%$	$47.89 \pm 0.21\%$	$48.72 \pm 0.23\%$
Relative acc	$36.37 \pm 0.18\%$	$46.40 \pm 0.20\%$	$48.08 \pm 0.21\%$	$48.41 \pm 0.25\%$	$48.65 \pm 0.15\%$	$48.72 \pm 0.30\%$	$48.79 \pm 0.28\%$	$50.22 \pm 0.21\%$

3.5.3 Examples of Generated Images

In this part we show sample images generated by LSR-GAN and DCGAN with a random seed $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Those results can give use a sense about how good the random generations are for different datasets, where the results are from both LSR-GAN and DCGAN. The LSR-GAN is trained with β =1 and λ=1. The samples are shown in Figure 3.11 for an LSR-GAN trained on CIFAR-10 and ImageNet. We rescale the images for better alignment. Where the original size of CIFAR-10 is 32×32 and of ImageNet is 64×64. Although the images superficially look reasonable on close inspection it is clear that most samples for the LSR-GAN trained on CIFAR-10 and ImageNet are not real world objects. This reflects the fact that the images for these two dataset are very variable leaving most of the latent space representing rather surreal objects.



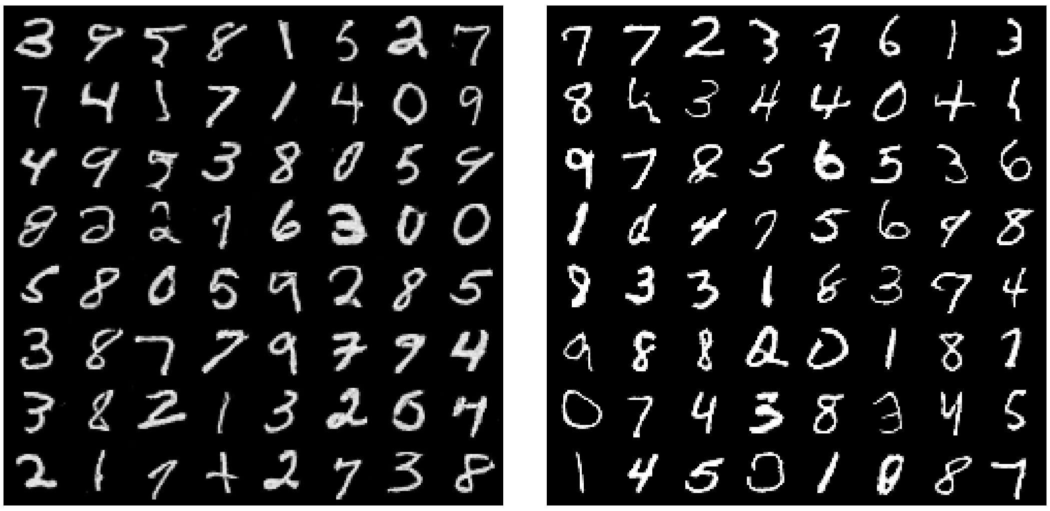
(a)



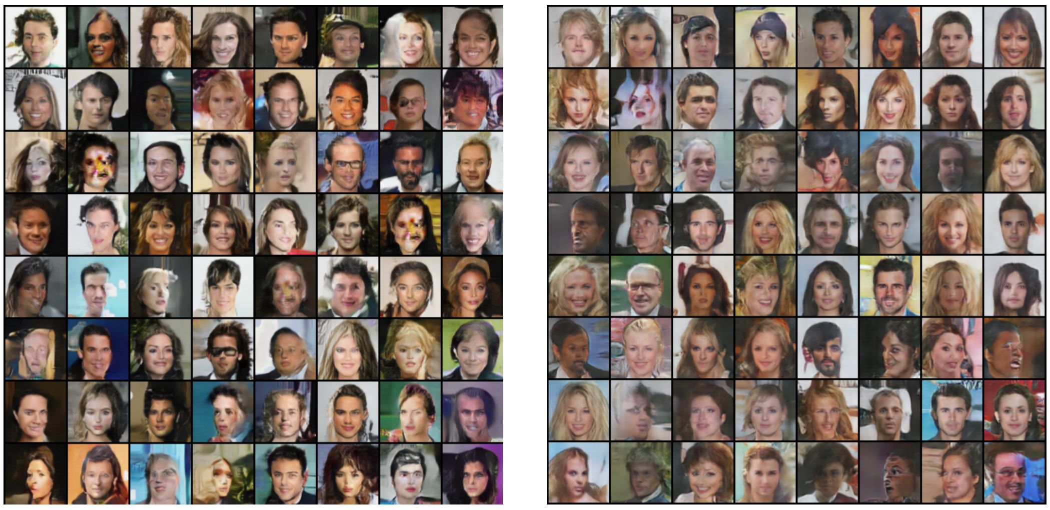
(b)

Figure 3.11: Random samples generated by LSR-GAN and DCGAN. Left column is from LSR-GAN and righ column is from DCGAN. (a) is CIFAR-10 and (b) is ImageNet.

We have also trained LSR-GAN on MNIST and Celeb-A with samples shown in Figure 3.12. These two datasets are less complex than CIFAR-10 and ImageNet. Perhaps unsurprisingly, most samples are identifiable.



(a)



(b)

Figure 3.12: Random samples generated by LSR-GAN and DCGAN. Left column is from LSR-GAN and righ column is from DCGAN. (a) is MNIST and (b) is Celeb-A.

3.6 Conclusion

VAEs are often taken to be a method for generating samples that is easier to train than a GAN, but gives slightly worse results. If this is the only objective then it is clearly legitimate to modify the VAE in anyway that will improve its performance. However, we believe that this risks losing one of their most desirable properties, namely their ability to learn features of the whole dataset. We have argued that because of this property, a VAE is not an ideal generative model. It will not be able to reconstruct data accurately and consequently will struggle even more when generating new samples.

As we have argued, a consistent way of using the latent space of a VAE is to use a GAN as a data renderer, using the VAE encoder to ensure that the GAN is generating images that represent the information encoded in the VAE's latent space. This involves "imagining" the information that the VAE disregards. LSR-GAN can be particularly useful in generating random samples, although, as shown in the section 3.5, for very diverse datasets the samples are often not recognisable as real-world objects. Although there are already many VAE-GAN hybrids, to the best of our knowledge, they are all designed to "fix" the VAE or 'enhance" the GAN.

Chapter 4

Transfer Learning in Variational Autoencoder

Transfer learning is the process of using knowledge learned on one dataset to solve a problem involving a second dataset. In this chapter, we investigated the ability of VAE to transfer the knowledge learnt from one dataset to another dataset. VAEs are powerful at learning representations, however, it is time-consuming and money-consuming to train different VAEs from the beginning for different datasets. Also, VAEs ask for a huge amount of data which is difficult and expensive to be collected in real world. Transfer learning can help us solve this issue by using a VAE that is pre-trained on a related dataset, which allows us to get a well-trained VAE with less time and data. Also, transfer learning enables the VAE to hold a semantic latent space with few data. Although transfer learning is rarely used in VAEs we will see in this chapter that it is surprisingly effective.

4.1 Introduction

There is a common assumption for most of the deep learning methods that the training set and test set are from the same distribution. It is hard for a model to perform well on other out of distribution datasets. But this does not imply that the knowledge learnt by the neural network cannot be transferred to the out of distribution dataset. In many tasks, people believe that a ResNet (He et al., 2016) pre-trained with ImageNet can enhance the results especially when the training data are not enough. This idea can be helpful in deep generative models. In the real world, it is expensive to collect data and re-train a huge model. It would be efficient if we can apply the knowledge learnt from a source domain to the target domain. Since the distribution learnt by a generative model can be applied to other downstream tasks such as classification. It is expected that this property can be transferred to the new dataset when we apply transfer learning on generative models.

There are attempts for transfer learning that are based on deep generative models developed already. Wang et al. (2018); Noguchi and Harada (2019) and Zhao et al. (2020) both achieve generating images with limited training data when the pre-trained GAN model is powerful enough, these models can generate samples with diversity even when the style in training data is limited. Wang et al. (2018) show that these techniques can help the model converge faster and performs better than re-train a new model. While Li et al. (2019) solve the mode collapse problem by modifying the class embeddings of a pre-trained class-conditional GAN and can generate samples of new classes. All of these previous works are based on GAN models but not VAE models. And these GAN-based pre-trained models can only work for image generation but no other tasks. Belhaj et al. (2018) propose a semi-supervised classification network based on VAEs that can share the knowledge across the domain among supervised data and unsupervised data when the labelled data are not enough. Unlike this work, we explore the ability of VAEs to act as a pre-trained model and we find it can help VAEs to converge faster and get a better ELBO, which leads to a better representation.

4.2 Overview of Visualisation Technique

In the deep learning field, the data is usually non-linear and high-dimensional, it is common that the model projects the data on a manifold with the geometric shape like curve, sphere, etc. Using Principal Components Analysis (PCA) can lead to poor performance especially when dealing with non-linear data. Thus, we apply t-Distributed Stochastic Neighbor Embedding (t-SNE) (Van der Maaten and Hinton, 2008) which is a widely used method for visualizing high-dimensional data.

PCA is a deterministic method, which aims at finding the principal components to remain the most important information. Unlike PCA, t-SNE preserves the pairwise similarities between data points. The cost function is

$$\boldsymbol{C} = KL(\boldsymbol{P}||\boldsymbol{Q}) = \sum_i \sum_j p_{ij} \log(\frac{p_{ij}}{q_{ij}})$$

Where $\boldsymbol{P}$ is the joint distribution in the high-dimensional space and $\boldsymbol{Q}$ is the joint distribution in the low-dimensional space. We define x and y as the data point in the high-dimensional and low-dimensional space separately. Then q_{ij} is the pairwise similarities in the low-dimensional space given as:

$$q_{ij} = \frac{exp(-\left\|y_i - y_j\right\|^2)}{\sum_{k \neq l} exp(-\|y_k - y_l\|^2)}$$

However, this form cannot be used directly to p_{ij}, the pairwise similarities in the high-dimensional space. Since if there is an outlier in the dataset, it will make the other data points have little effects on the cost function. Thus, p_{ij} is defined as:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n}$$

Where n is the number of data points and $p_{i|j}$ is given as:

$$p_{i|j} = \frac{exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{i \neq k} exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$$

σ_i is the variance of the Gaussian that is centred on x_i. By optimising the cost function, we can obtain distribution in the low dimension that preserves the same similarity in the high dimension. Usually, we need to set the value of perplexity which is the number of nearest neighbours that is used in the algorithm. We set perplexity as 30 which is a common setting in all experiments.

4.3 Transfer Learning

Transfer learning is used to improve the model from one domain by transferring information from another related domain. It is useful when there is a limited amount of target data. With big data becoming a common setting for deep learning models, using existing datasets that are related to the target dataset is an attractive approach. For a real-world example, a person who has learned Chinese before can learn Japanese more efficient than other people that have not learned Chinese since there are a lot of Chinese characters in Japanese words.

There are many research works that have successfully investigated transfer learning such as image classification (Zhu et al., 2011; Kulis et al., 2011) and multi-language text classification (Zhou et al., 2014). In many transfer learning solutions, the researchers focus on correcting the difference between the source domain and the target domain. However, this is not necessary for the training of VAEs since we assume the prior distribution in VAEs is Normal distribution for all different datasets. In other words, for VAEs trained on different datasets we map the posterior distributions to the same prior distribution. Presumably, a well-trained VAE should perform reasonable reconstruction and representation extraction on a related dataset and the results can be improved through fine-tuning. Thus, we investigate this assumption in the next section.

4.4 Experiments

In this section, we investigate the performance of a vanilla VAE pre-trained model with the COCO dataset (Lin et al., 2014) on other datasets. The architecture of the encoder is 3 layers CNN network with Batch Normalization and 2 layers fully connected network, and the decoder consists of 4 layers CNN network. The optimizer is Adam with a learning rate of 2×10^{-4} and $\beta_1 = 0.5$ (Kingma and Ba, 2015). We resize all the images to 64×64 to fit the size of the encoder. The diagram of the model architecture is shown in Figure 4.1.

First, we test the pre-trained VAE without fine-tuning which means all the parameters of the VAE have been frozen and there is no training procedure when we test the pre-trained VAE on other target datasets. Then we demonstrate the performance of a pre-trained VAE with fine-tuning which means we train another 50 epochs on target datasets. The pre-trained VAE can be regarded as an initialisation of the VAE training, however, we restrict the size of the target dataset and the training time, we also compare the results with a vanilla VAE trained on the target dataset without transfer learning. In the last, we investigate the performance of VAEs on the semantic latent space.

4.4.1 Transferring VAE without Fine-tuning

When the training dataset is large and complex enough such as ImageNet or COCO, the VAE can reconstruct other simple datasets like CIFAR-10, Celeb-A

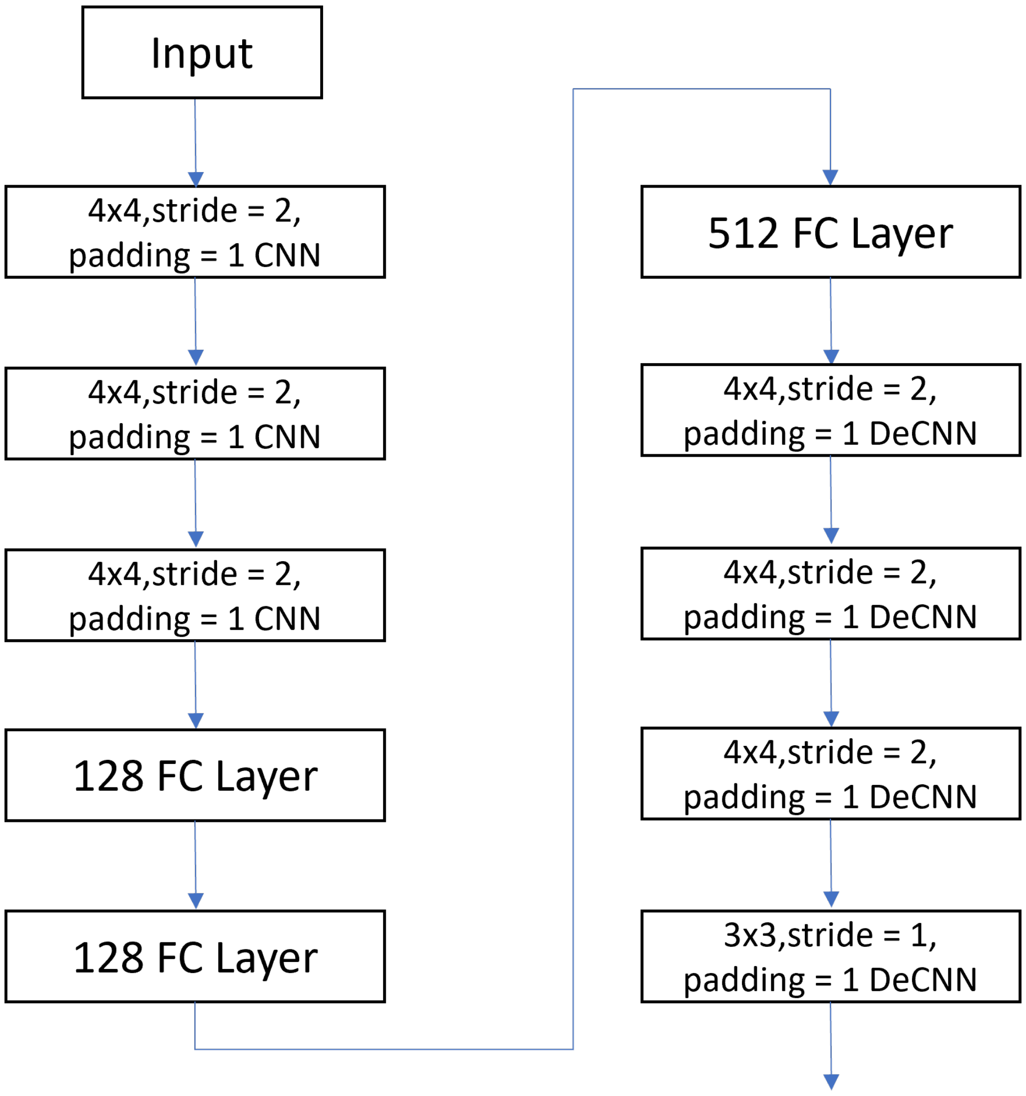

Input
4x4,stride = 2, padding = 1 CNN
4x4,stride = 2, padding = 1 CNN
4x4,stride = 2, padding = 1 CNN
128 FC Layer
128 FC Layer
512 FC Layer
4x4,stride = 2, padding = 1 DeCNN
4x4,stride = 2, padding = 1 DeCNN
4x4,stride = 2, padding = 1 DeCNN
3x3,stride = 1, padding = 1 DeCNN

Figure 4.1: The diagram of the model architecture

and MNIST, although it cannot generate random samples like MNIST or Celeba-A dataset without fine-tuning.

As shown in Figure 4.2, there are reconstruction images of 4 different datasets. For each dataset, the first row is the original image and the second row is the reconstruction image. We upsample CIFAR-10 and MNIST to 64×64 and convert MNIST to RGB images while it remains black and white. All the reconstruction images are reasonable although they are still blurry and lost details. It is acceptable since the VAE pre-trained on COCO has never seen the details of other datasets. The results show that once a VAE is trained with enough data, it is able to reconstruct samples with the same type of data. But a VAE trained with COCO can never generate images like MNIST. Then we investigate the difference

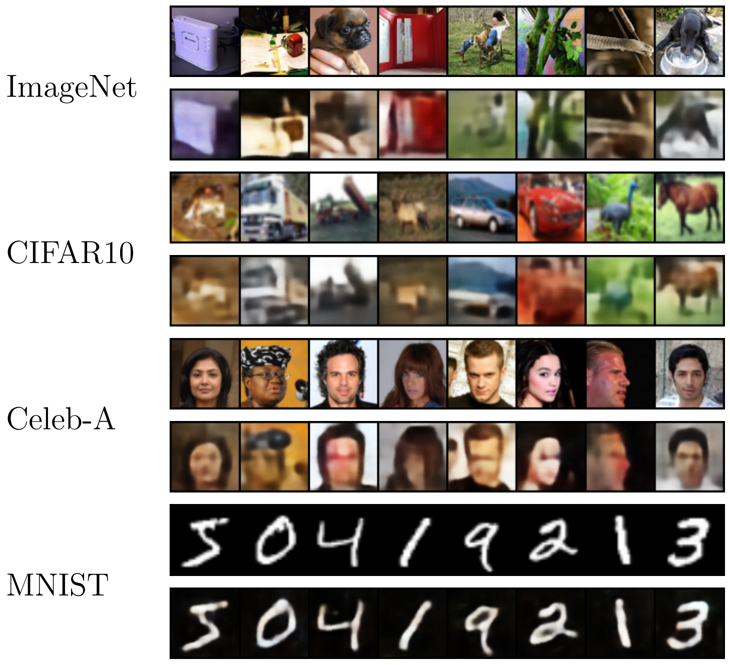


Figure 4.2: Reconstructions of 4 different datasets from a VAE trained with COCO dataset. For each dataset, the first row is the original image and the second row is the reconstruction image. All the datasets have been rescaled to 64x64.

between the distribution of these datasets in the latent space. We randomly select 500 samples from the 5 different datasets and collect the μ from the encoder.

Figure 4.3 is the result after we apply t-SNE. Most data points are in close proximity. It is not surprising that MNIST is out of the distributions of other datasets in the latent space since other images are all real images but MNIST is a dataset containing binary images of handwritten digits. But the decoder is still able to reconstruct it. We cannot see too many differences between COCO, ImageNet and CIFAR-10 in this two dimensions visualisation since these three distributions highly overlap. And Celeb-A is not like the other 3 datasets that cover all the area and it only lies in the corner area, which corresponds to the speciality of Celeb-A that it only contains human-face, can be regarded as a subset of the other 3 datasets at a representation view. Then we investigate another quantitative measurement.

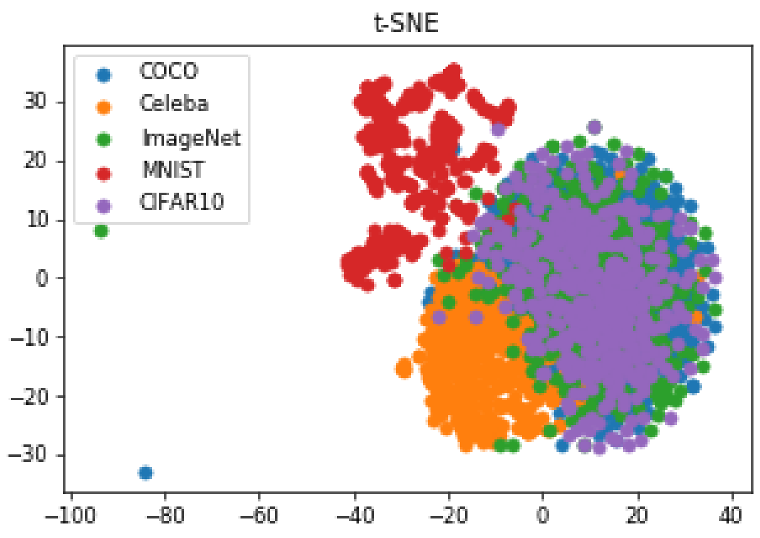


Figure 4.3: t-SNE result of 5 different datasets in the latent space, each dataset contains 500 samples.

First, we use a simple multiclass perceptron to test if these samples from different datasets are linearly separable in the latent space. As shown in Table 4.1, MNIST and Celeb-A are able to be classified in most cases by the linear classifier, which is not surprising according to the t-SNE result. COCO, ImageNet and CIFAR10 are still hard to be classified linearly even in the original high-dimensional space, this tells us that VAEs can encode these natural images into a close distribution, and this makes the transfer learning based on VAE more interpretable. We also compute the KL value between the posterior distributions of these different datasets and the prior (normal distribution in a vanilla VAE set). Lower is more close to the prior. Again, COCO, ImageNet and CIFAR10 are at the same level while Celeb-A and MNIST are quite far away from the former three datasets. The VAE pre-trained with COCO get the lowest KL value on CIFAR10 dataset. The reason of this result is because we upscaled CIFAR10 from 32×32 while other datasets are downscaled. The upscaling operation makes CIFAR10 images more blurry which is more close to the output of decoder. VAE is able to distinguish the datasets when they are visually far from each other but can still reconstruct them. And we can improve the result on the specific dataset by fine-tuning.

	COCO	ImageNet	CIFAR10	Celeb-A	MNIST
μ train acc.	44.6%	46.8%	42.8%	97.4%	99.8%
μ test acc.	35.4%	40.2%	33.0%	89.8%	100 %
KL	163.47	169.54	154.29	181.44	216.92

Table 4.1: Quantitative results in the latent space, classification accuracy based on a linear classifier, there is no good or bad accuracy for this result. The last row is KL values between the posterior distribution and the prior (normal distribution in a vanilla VAE set).

4.4.2 Transferring VAE with Fine-tuning

As other pre-trained models, a pre-trained VAE can help training converge faster and perform better especially when the amount of data is limited. We use Celeb-A as a training dataset to fine-tune the pre-trained VAE. And we choose only 100 images randomly from the original dataset. We fine-tune the model for 50 epochs. For comparison, we train a new VAE with the same 100 images for 50 epochs. And Figure 4.4 shows the reconstructions for test images. VAE-New is a retrained model and VAE-Tune starts from a pre-trained VAE. When the training data is limited, a VAE without pre-training cannot reconstruct the images well. A pre-trained VAE converge faster and better.

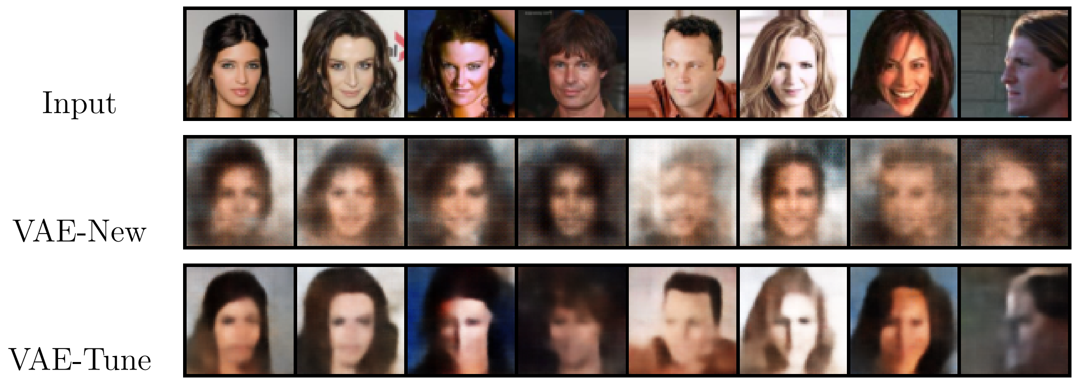


Figure 4.4: The reconstructions of Celeb-A. VAE-New is a retrained model and VAE-Tune starts from a pre-trained VAE.

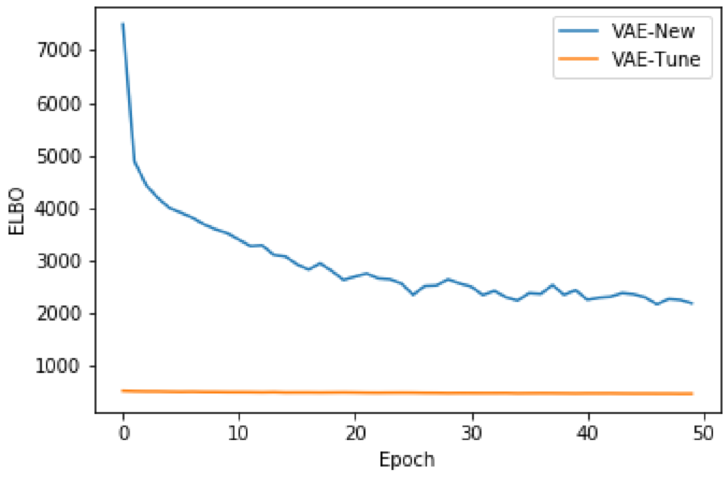


Figure 4.5: The ELBO of two models. VAE-New is a retrained model and VAE-Tune starts from a pre-trained VAE. The X-axis represents epoch and Y-axis represnets ELBO.

As shown in Figure 4.5, the ELBO of VAE-New is much higher than VAE-Tune which makes the line of VAE-Tune looks like a straight line. We can get the same results even if VAE-New has been trained for the same number of iterations as the pre-trained VAE. It will not change this result if we use the whole Celeb-A dataset as a training set.

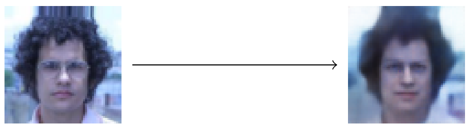


Figure 4.6: The reconstruction of face with light color glasses

4.4.3 The Semantic Latent Space

By applying transfer learning, the VAE is also able to learn a semantic latent space as a vanilla VAE does. If we increase the size of training data, the VAE-Tune can capture a meaningful representation with arithmetic operations on semantic attributes from limited data. It is crucial to increase the size of data because the VAE-Tune is prone to discard redundant information during fine-tuning. For example, if we use a smaller Celeb-A dataset that does not include face images with glasses, then the VAE-tune cannot reconstruct face images with glasses (especially for light colour glasses), as Figure 4.6 shows. For arithmetic operation, we first do a simple test

$$Female\_NoGlasses + Male\_Glasses – Male\_NoGlasses = Female\_Glasses \tag{4.1}$$

We do the experiment based on the above equation in the latent space and get Figure 4.7. Although the quality of images is not perfect, it corresponds to the semantic equation. To do this, we simply choose 3 images that correspond to the desired label in Equation 4.1 and use the μ obtained from the encoder to do the calculation. The result shows a human face that owns the attributes we are looking for according to Equation 4.1. Even the face orientation is plausible. In Figure 4.7, the negative image is a human face located at the left of the image and looks to the right, which can be approximately eliminated by the second image, the human face with glasses one. The second one is close to the left and the human face slightly looks to the right. Thus, the result holds the same orientation as the first female image. This result shows that the VAE-Tune can learn a reasonable semantic latent space even with limited data.

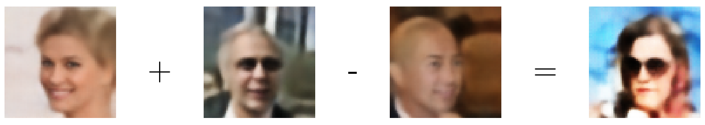


Figure 4.7: The arithmetic operaion about generating female with glasses.

Also, we do interpolations between an image with one attribute and the image without the attribute. We randomly pick 500 smiling face images and compute the mean of μ of these images and do the same thing on 500 non-smiling face images. Then we do interpretations following the semantic Equation 4.2. In this equation, $\lambda \in [0, 0.5]$.

$$NoSmile\_Sample + \lambda(Smile – NoSmile) = Smile\_Sample \tag{4.2}$$

The transition from a non-smiling face to a smiling face shown in Figure 4.8 proves there is a reasonable interpolation between semantic attributes in the latent space. We also have done an interpolation between two different attributes. Following Equation 4.3 we get Figure 4.9, it still performs well and it does not affect other attributes even when we have modified two attributes.

$$Sample + \lambda(Smile–NoSmile) + \lambda(Glass–NoGlass) = Smile\_Glass\_Sample \tag{4.3}$$

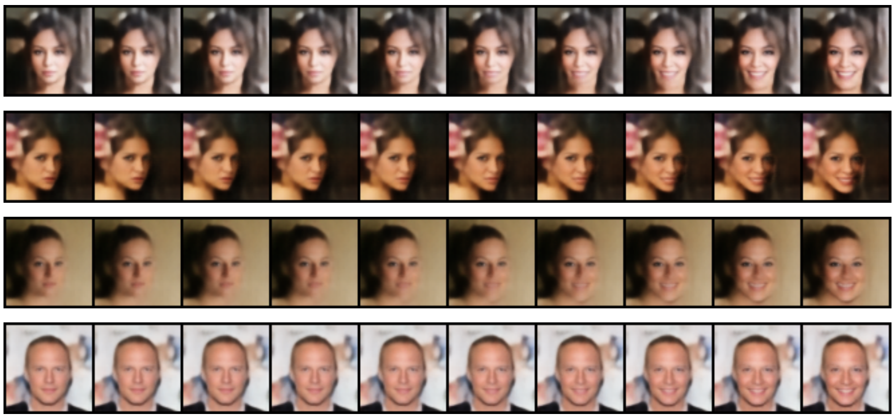


Figure 4.8: The face images from NoSmile to Smile

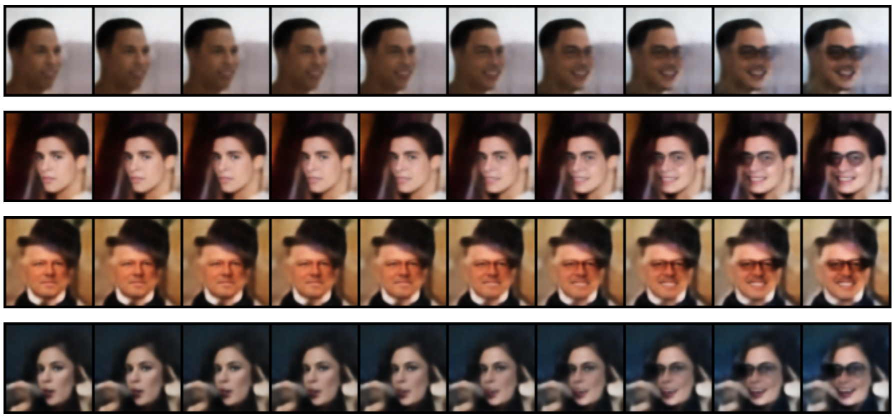


Figure 4.9: The face images from NoSmile_NoGlass to Smile_Glass

4.5 Conclusion

Transfer learning is a useful field especially when we only have a small amount of training data. Previous works that apply deep generative models in transfer learning usually focus on GANs due to its strong ability to generate high-quality images. We investigate whether VAE is capable to do transfer learning. Although what we have done a vanilla VAE without pre-training can also do, the advantage of applying a pre-trained VAE is that it can converge faster and perform better.

As we discussed in the previous chapter, VAEs are designed to encode information. The latent space a pre-trained VAE have learnt is also meaningful even when the training sets are totally different. This property can be useful when we have limited data for training set, which also indicates that the learned factorised representation is able to transferred from one dataset to another dataset, it is also a part of generalisation, as we will show in Chapter 6.

Chapter 5

Compositing Foreground and Background Via Variational Autoencoders

In this chapter we introduce a new model that can learn a representation that factorises the foreground and background information.

5.1 Introduction

Learning factorized representations of visual scenes is a challenging problem in computer vision. Human brains can process the realistic scene as a whole and decompose it into different parts using visual clues and prior knowledge. This cognitive ability also enables humans to imagine different scenes. Objects form the basis of humans' high-level cognition (Spelke, 1990). Thus, learning good object representations could be an important step towards making artificial intelligence closer to human intelligence. In visually inspecting a scene, one object is often attended to as the foreground and the rest of the scene is the background. There exists a considerable body of work learning representation for each object in a scene and achieve objects segmentation (Nash et al., 2017; Greff et al., 2019; Burgess et al., 2019; Lin et al., 2020). We argue that a good object representation should not only benefit the downstream tasks such as classification, or segmentation, but also enable generative models to create images conditioned on the object representations.

Our aim is to build a generative model for classes of images that allows us to alter the foreground objects independently of the background. This requires building a model that factorizes and composites these two parts representations of the image. However, the same part of an image can be either the background or the foreground according to different task. Thus, we simply define the object inside the given bounding box as the foreground in our experiment setting for different datasets. Existing works that can factorize the foreground and background of images are all based on hierarchical Generative Adversarial Networks (Yang et al., 2017a; Singh et al., 2019; Li et al., 2020). Here we introduce a new VAE-based model that can be used to factorize the background and foreground objects in a continuous latent space and composite factors of those training images to generate new images in one shot. Compared to GAN-based models, our VAE-based models can infer the latent representation of existing images in addition to performing generation.

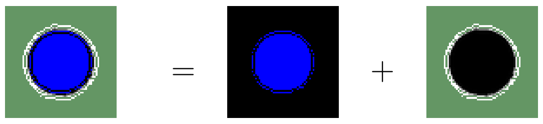


Figure 5.1: The decomposition of an image

We consider the decomposition of an image $\boldsymbol{x}$ into a set of foreground pixels $\boldsymbol{f}$ and background pixels $\boldsymbol{b}$ such that

$$\boldsymbol{x} = \boldsymbol{f} \odot \boldsymbol{m} + \boldsymbol{b} \odot (1 - \boldsymbol{m}),$$

where $\boldsymbol{m}$ is the binary mask of the foreground and $\odot$ denotes element-wise multiplication. As shown in Figure 5.1, every image can be decomposed into two parts mathematically, foreground (i.e. object) and background.

We propose a new model *Background and Foreground VAE* (BFVAE), which consists of two VAEs that can learn a disentangled representation between foreground information and background information. This can be useful when you want to generate images with the same object in different new backgrounds or vice versa. And the definition of background and foreground in an image should be based on human knowledge, for example, imaging a boy stands in a forest, it is hard to tell if the trees belong to foreground or background without human labelling. We use the whole image and the foreground image as inputs to the two VAEs. First, we use VAE-F to represent the VAE having foreground images as

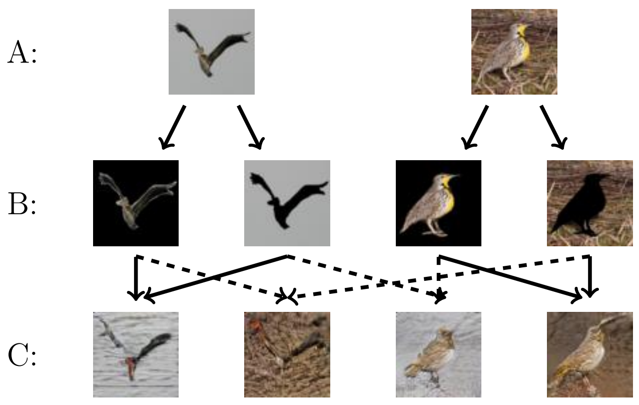


Figure 5.2: Row A is the original images. Row B is the decomposition of Row A. Row C is the generations of combining different factors. Between row B and row C, the solid arrow means reconstruction and dashed arrow means generation.

inputs, and VAE-F is a vanilla VAE. Then we use VAE-B to represent VAE that use whole images as inputs, we concatenate the latent space of the two encoders before we put it back into the decoder. The two VAEs are trained simultaneously. Due to the VAE-F only being able to learn a representation of foregrounds, it can force the latent space learnt by the encoder VAE-B to only preserves information about the background. This model can generate unseen images by exchanging the latent space. The reason that we choose the whole image as an input but not the background image is because the background image will limit the utility of this model. It is hard to obtain images with the foreground cropped out, by using whole images we can use the model to extract the background information from images without mask label. Also, using the whole image as input enables the model to generate a background image by setting the foreground image into all black but not generate an image with a blank area.

The drawback of the VAE-based model is the generated images tend to be blurry, thus we combine BFVAE with LSR-GAN (introduced in Chapter 3). In addition, we note that a different pre-processing operation of the images makes a significant difference to the FID scores. Thus, we claim that the details of how to compute FID scores in each paper are necessary to be clarified.

As shown in Figure 5.2, these images are the results from the combination of the BFVAE and the LSR-GAN. Row C is the generated images of our model by combining different factors in row B. The background and foreground of images in row C are not totally the same as images in row A and row B, this is due to a trade-off between the similarity and reality in our model which will be addressed in the following section.

5.2 Differences between BFVAE and Related Work

Not many works focus on compositing images in a background-foreground manner. The existing models are all GAN-based models that generate foreground and background separately and recursively, the generated images are stitched at the final stages (Yang et al., 2017a; Singh et al., 2019; Li et al., 2020), which means they cannot learn a factorised latent space of the training data, except MixNMatch (Li et al., 2020). Unlike these models, our model generates the whole image in one shot while it can still learn a continuous factorized latent space of the dataset.

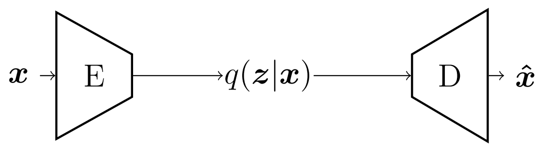
5.3 Model

In this section we describe the components of our model. Recall that a Variational Autoencoder (VAE) (Kingma and Welling, 2013) is a deep generative model that learns a distribution over observed data, $\boldsymbol{x}$, in terms of latent variables, $\boldsymbol{z}$. The original VAE approximates the intractable posterior by using a variational approximation to provide a tractable bound on the marginal log-likelihood called the evidence lower bound (ELBO)

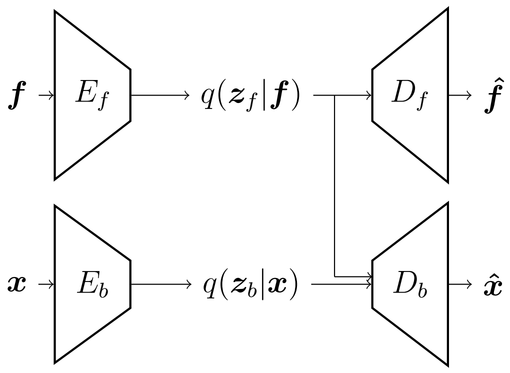
$$\log p_\theta(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z}\sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)\,. \tag{5.1}$$

Commonly, $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is the output of an inference network with parameters ϕ and $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ is generated by a decoder network with parameters θ. The architecture of the VAE is in Figure 5.3a.

Starting from a mask $\boldsymbol{m}$ of the foreground object we can extract the foreground $\boldsymbol{f}$ from the image $\boldsymbol{x}$ using $\boldsymbol{f} = \boldsymbol{x} \odot \boldsymbol{m}$. We use $\boldsymbol{f}$ and $\boldsymbol{x}$ as inputs to our two VAEs. The architecture of our model is shown in Figure 5.3. For simplicity, we omit symbols of the parameters. The top network is VAE-F and it acts like a vanilla VAE. The encoder E_f generates a probability distribution, $q(\boldsymbol{z}_f|\boldsymbol{f})$ that acts as a latent space representation of $\boldsymbol{f}$. A sample from this distribution, $\boldsymbol{z}_f$, is used by the decoder D_f to generate a reconstruction, $\hat{\boldsymbol{f}}$, of the input $\boldsymbol{f}$. The top VAE ensures the $\boldsymbol{z}_f$ contains representations of foreground objects. The bottom



(a) The architecture of a vanilla VAE.



(b) The architecture of our model, the top one is VAE-F and the bottom one is VAE-B.

Figure 5.3: The architectures of two models

network is VAE-B. The original image, $\boldsymbol{x}$, is given to the encoder, E_b that generates a probability distribution, $q(\boldsymbol{z}_b|\boldsymbol{x})$. A latent variables $\boldsymbol{z}_b$, is sampled from this distribution and concatenated with $\boldsymbol{z}_f$. This concatenated vector is sent to decoder D_b that must reconstruct the original image. Thus, we modify the ELBO for VAE-B to be

$$\mathcal{L}_b = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|(\boldsymbol{z}_b,\boldsymbol{z}_f))\right] - D_{KL}\big(q(\boldsymbol{z}_b|\boldsymbol{x})||p(\boldsymbol{z}_b)\big)\,. \tag{5.2}$$

Since the input $\boldsymbol{f}$ does not contain any information about the background, it is assured that the latent variables $\boldsymbol{z}_f$ only contain information about the foreground. For VAE-B, the encoder can extract information about both foreground and background from $\boldsymbol{x}$. When we train the decoder with both $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$, it can force $\boldsymbol{z}_b$ to discard the information about the foreground and only leave background information. This also enables us to extract the pure background from images by using $\boldsymbol{z}_f$ obtained from a pure black image. In the initial stages of training, $\boldsymbol{z}_b$ contain all the information about the image. This makes the decoder of VAE-B prone to ignore $\boldsymbol{z}_f$ especially when the dataset is complicated. There are two methods to alleviate this issue. The first method is to set the size of $\boldsymbol{z}_b$ to be reasonably small, it forces the $\boldsymbol{z}_b$ to discard information, but this design makes it hard to find an accurate size for $\boldsymbol{z}_b$. Thus, we recommend the second method which is turning the model into β-VAE,

$$\mathcal{L}_b = \mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|(\boldsymbol{z}_b,\boldsymbol{z}_f))\right] - \beta D_{KL}\big(q(\boldsymbol{z}_b|\boldsymbol{x})||p(\boldsymbol{z}_b)\big) \tag{5.3}$$

It is well known (see, for example, Hoffman et al. Hoffman and Johnson (2016) and Kim et al. Kim and Mnih (2018)) that the expected KL term in Equation (5.3) can be rewritten as

$$\mathbb{E}_{p_{data(x)}}[D_{KL}\big(q(\boldsymbol{z}_b|\boldsymbol{x})||p(\boldsymbol{z}_b)\big)] = D_{KL}\big(q(\boldsymbol{z}_b)||p(\boldsymbol{z}_b)\big) + I(\boldsymbol{x}, \boldsymbol{z}_b) \tag{5.4}$$

By setting $\beta > 1$, we penalize both terms on the right side of equation (5.4). Penalizing $D_{KL}\big(q(\boldsymbol{z}_b)||p(\boldsymbol{z}_b)\big)$ encourages factorization of the latent space, while at the same time it pushes $q(\boldsymbol{z}_b|\boldsymbol{x})$ towards a standard Gaussian distribution. But the most important part in BFVAE is that we penalize the mutual information term $I(\boldsymbol{x}, \boldsymbol{z}_b)$ which helps $\boldsymbol{z}_b$ to discard information about the foreground.

There is a situation where the VAE-F is unable to capture the foreground information. When the object is black and the dataset is simple enough, as shown in figure 5.1, the foreground image becomes a pure black image, this makes all the image with black foreground the same. To handle this, we add the fourth channel to the encoder of VAE-F, which is the binary mask label of images, it provides shape information about foregrounds and can solve this issue perfectly.

5.3.1 BFVAE with LSR-GAN

A prominent problem of vanilla VAEs is the blurriness of the output. Thus we use LSR-GAN, first introduced in Chapter 3. That can learn a latent space from BFVAE and generate high-quality images. Recall the idea is that we pass the output $G(\boldsymbol{z})$ of the generator G into the two encoders of BFVAE, and ask the two encoders to map $G(\boldsymbol{z})$ to the latent vectors $\boldsymbol{z}$ that we used to generate $G(\boldsymbol{z})$. By doing this, the generator will generate an image with a latent space encoding, $\boldsymbol{z}$, of the pre-trained BFVAE. It can be seen as a simple regularization term of the normal GAN loss function for the generator

$$\mathcal{L}_G = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(D(G(\boldsymbol{z})))] + \lambda \log(q(\boldsymbol{z}|(E_b(G(\boldsymbol{z})), E'_f(G(\boldsymbol{z})))) \tag{5.5}$$

where E_b means the encoder of VAE-B, and the $(E_b(G(\boldsymbol{z})), E'_f(G(\boldsymbol{z}))$ in the second term represents the concatenation of $E_b(G(\boldsymbol{z}))$ and $E'_f(G(\boldsymbol{z}))$. We train a new encoder E'_f that can extract the $\boldsymbol{z}_f$ from $G(\boldsymbol{z})$, we freeze all the other parts of BFVAE and replace E_f with E'_f, then train the encoder in the original manner. Note that when training the LSR-GAN we freeze the weights of the E_b and E'_f. The constant λ is an adjustable hyper-parameter providing a trade-off between how realistic the image looks and the similarity between reconstructions and real images. Although the idea is simple, it provides a powerful method to improve the image quality of BFVAE. The generator G can either be a new network or a pre-trained decoder of BFVAE. The pre-trained decoder can be a strong initialization for the generator when the generator meets with model collapse.

5.4 Experiments

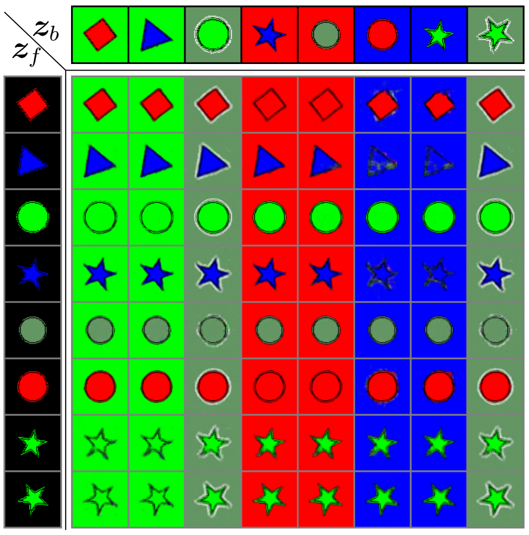
In this section, we show the results on factorised latent space in two steps, the BF-VAE without LSR-GAN and the BFVAE with LSR-GAN. Then we demonstrate the BFVAE can also work on other desired factors.

5.4.1 BFVAE without LSR-GAN

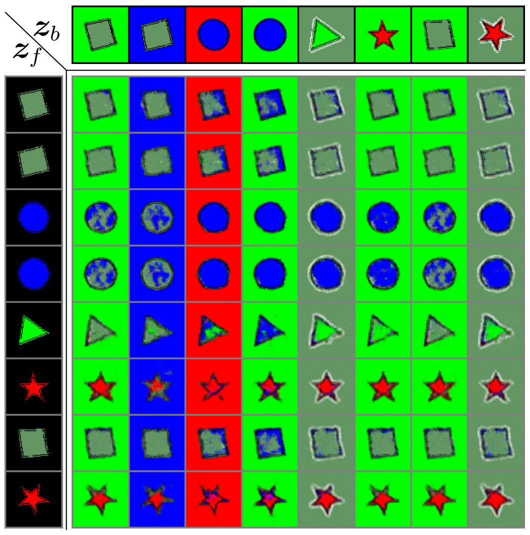
We evaluate our model on a toy dataset, COCO and CLEVR. The toy dataset consists of 4 shapes with 4 colours of objects and backgrounds. Each shape contains 2000 images in the training set. While COCO and CLEVR are datasets with multi-objects in one single image. We show the results quantitatively and qualitatively. The architecture of the encoder is 4 layers CNN network with Batch Normalization and 2 layers fully connected network, and decoder consists of 5 layers CNN network. The optimizer is Adam with a learning rate of 2×10^{-4} and $\beta_1 = 0.5$ (Kingma and Ba, 2015).

Toy dataset. The toy dataset originally contains 4 shapes: square, star, circle, and triangle. We re-render the colour of the object and the background to red, green, blue and dark green. Three-quarters of colours of each object are used as training set. The colour of the object and background in one single image is different. And we down-sample the image to size 64x64. We also show the results when we change one of the four colours of objects in the training set to black. The dimensions of the latent space of two VAEs are both 32. We choose $\beta = 1$ in this experiment.

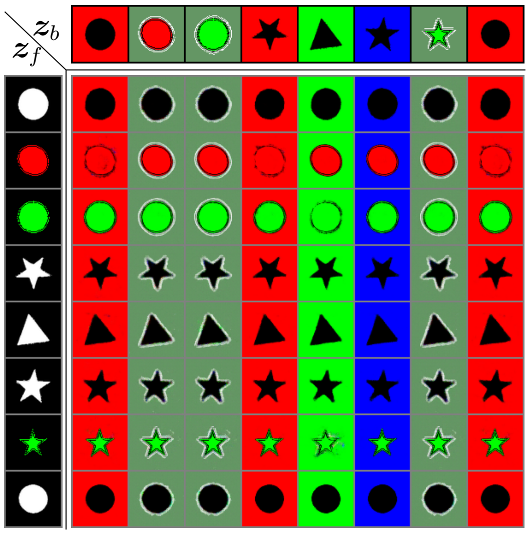
Figure 5.4 shows the generations by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$. The model can generate the image with the same colour of objects and backgrounds successfully, which has never been seen in the dataset. And by adding a mask channel, it can handle black objects. But VAE-F fails to pass the colour of objects in the test set correctly for some generations in (b) when the input is unseen but the elements can be found in the training set. And it can still pass the shape



(a)

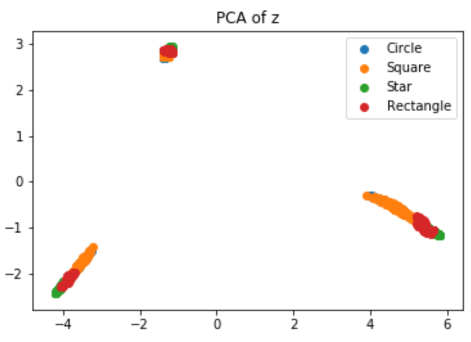


(b)

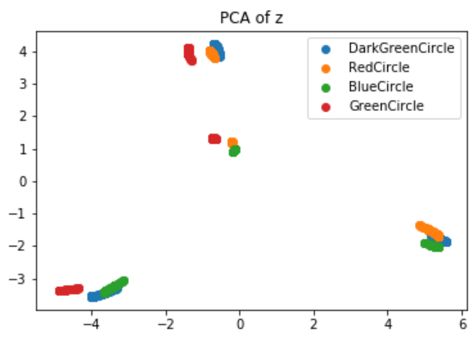


(c)

Figure 5.4: Generation by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$. The top row is $\boldsymbol{x}$ and the first column is $\boldsymbol{f}$. (a) is the result of training set without black colour, images with the same colour of objects and backgrounds have never been seen before. (b) is the result of the test set and the colour of these objects are unseen in the training set. (c) is the result of the training set with black colour, the black object images are shown as binary masks.



(a)



(b)

Figure 5.5: The results of PCA operation applied to the latent space. (a) contains 4 shapes with the same colour and 3 different colours of backgrounds. (b) contains 1 shape with 4 different colours and backgrounds.

information correctly. This finding seems to support the observation of Zhao et al. (2018) who found poor generalisation when the training set was too small (i.e. 12 combinations for shapes and colours). So the VAE-F fails to generalise to some images in the test set. Also, the VAE-B does not discard the foreground information totally which makes the result maintain the colours from $\boldsymbol{x}$ when the VAE-F cannot generalise to $\boldsymbol{f}$. The reason is that the object in this dataset is always situated at the centre part of the images so the encoder has no chance to infer a pure background image easily.

The results in Figure 5.4 can also be reflected after applying PCA to the $\boldsymbol{z}_b$. The foreground information in this dataset contains shapes and colours, we test the shape and colour separately because the number of shapes, backgrounds and colours are the same. The first situation is that the inputs contain all the shapes with only one colour. As shown in Figure 5.5a, if we visualize the $\boldsymbol{z}_b$ by the top-2 principal components, there are only 3 clusters which are the number of backgrounds but not shapes. Figure 5.5b is the plot when inputs are all circle with different colours, it consists of 4 clusters which depend on the backgrounds while in each cluster it has 3 small clusters which depend on the colour of objects. Thus, the colour of objects is a less important part of $\boldsymbol{z}_b$ while it has been ignored by the decoder when generating images.

We also set a quantitative measurement on the latent variables. We train a two-layer MLP classifier to classify both $\boldsymbol{z}_b$ and $\boldsymbol{z}_f$. This is a good indicator to show the amount of information in the latent space. Table 5.2 is the classification results. The first row is for shape information and the second row is for colour information. Ideally, $\boldsymbol{z}_b$ should be as low as possible since we want it to lose the foreground information While $\boldsymbol{z}_f$ should performs well on both tests. However, the accuracy of $\boldsymbol{z}_b$ on the training set is quite good but the classifier fails to generalize to test data. The classifier is overfitted on $\boldsymbol{z}_b$ while $\boldsymbol{z}_f$ keeps 100% accuracy. $\boldsymbol{z}_b$ and $\boldsymbol{z}_f$ have similar performance for the colour classification which is unexpected. But this corresponds to the result we have shown before. $\boldsymbol{z}_b$ still contains information about the colour due to the data speciality.

Table 5.2: Quantitative results for this toy dataset. Lower is better for $\boldsymbol{z}_b$ and higher is better for $\boldsymbol{z}_f$.

	$\boldsymbol{z}_f$ train acc.	$\boldsymbol{z}_b$ train acc.	$\boldsymbol{z}_f$ test acc.	$\boldsymbol{z}_b$ test acc.
All shapes and one colour	100%	86.7%	100%	36.3%
All colours and one shape	100%	100%	88.3%	88.27%

Multi-object datasets. CLEVR and COCO are both multi-object dataset, but the background of CLEVR is fixed. Except the original CLEVR dataset, we also change CLEVR to a 2D image dataset with red, green and blue backgrounds. COCO is full of different types of background and foreground, due to the blurriness problem of the VAE, the generation images are hard to be recognized. However, it can still be noticed that foregrounds have been swapped if the colour is vivid. We set $\beta = 5$ in this experiment.

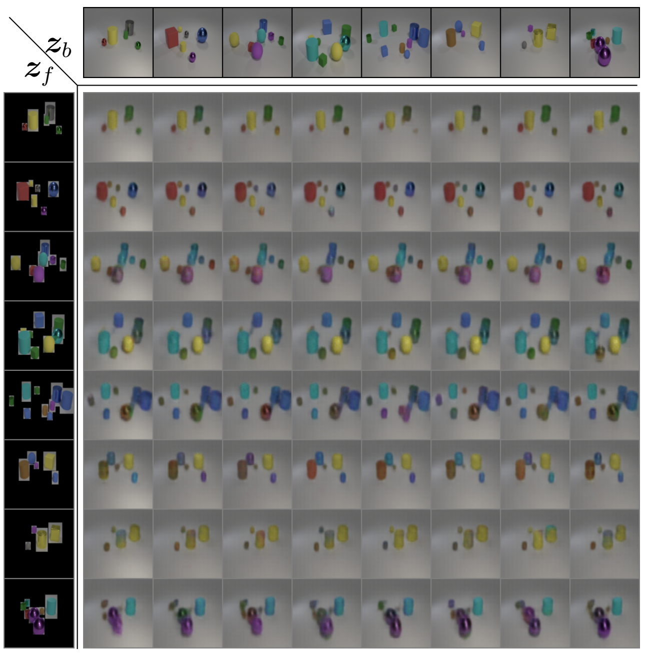


Figure 5.6: Generation by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$ on CLEVR. The top row is $\boldsymbol{x}$ and the first column is $\boldsymbol{f}$.

We first show the result of generation by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$. As the background is fixed, each row in Figure 5.6 is the same. But it can generate multiple objects correctly. Moreover, our model can generate multiple objects gradually by adding objects in $\boldsymbol{f}$.

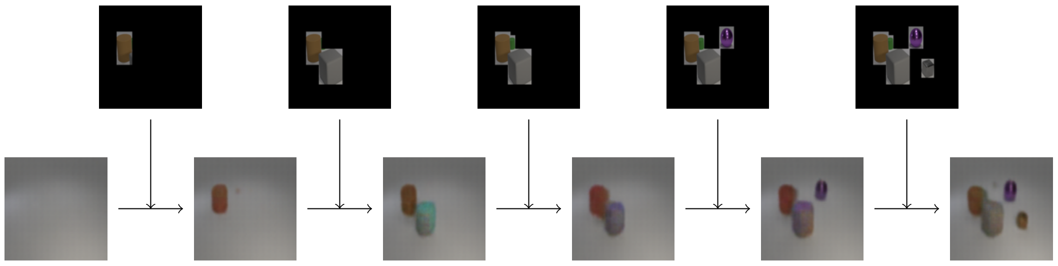


Figure 5.7: The gradual generation by adding objects in $\boldsymbol{f}$. The first blank image is generated by using a random $\boldsymbol{x}$ and with a whole black image as $\boldsymbol{f}$ (represents no objects in the image).

Images at the top row are $\boldsymbol{f}$ and images in the bottom row are $\boldsymbol{x}$. We get the first blank images by using a random $\boldsymbol{x}$ and with a whole black image as $\boldsymbol{f}$. Because $\boldsymbol{z}_b$ ignores the foreground information, we cannot add objects in the image by using a $\boldsymbol{f}$ containing only one object each time. We need to add those objects together before putting them into VAE-F. So it is the same if we only use the first blank image as $\boldsymbol{x}$ in this scenario. This shows the VAE-F and VAE-B is capable to handle multi-object images and control the number of objects.

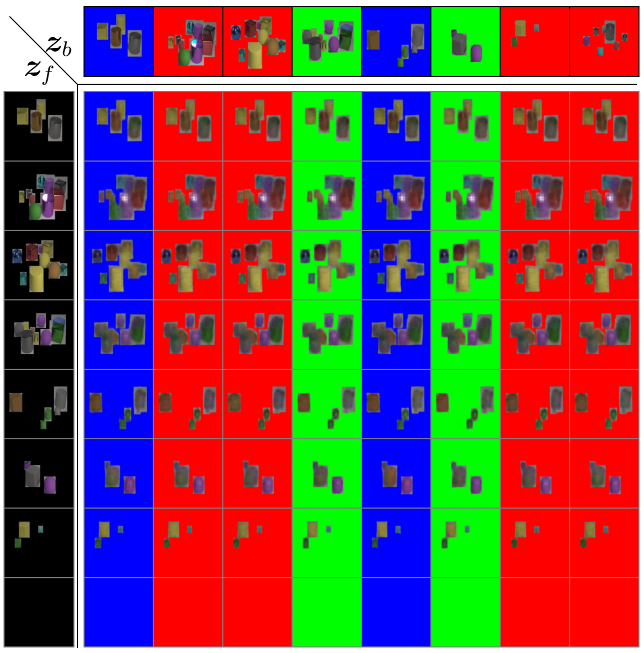


Figure 5.8: Generation by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$ on CLEVR with RGB backgrounds. The top row is $\boldsymbol{x}$ and the first column is $\boldsymbol{f}$.

The background of the original CLEVR dataset is the same for all images, which makes the VAE-F output the same thing for all the inputs. Thus, we change the background to red, green and blue. The generated images by swapping $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$ is shown in Figure 5.8. We remove all the objects in the input of VAE-F which makes $\boldsymbol{z}_f$ contains no information about the foreground and the generated images in the last row is exactly the background images without foreground. And these pure background images have never been seen by neither VAE-F or VAE-B. Unlike the toy dataset, objects in the CLEVR dataset is flexible, which provides the possibility for the encoder to inference a pure background image. This also supports our claim that the VAE-B is capable to extract the background information from whole images and discard the information about the foreground.

Here we also present the generations on COCO dataset, this dataset is more complex than ImageNet, since it is a multi-object dataset and each image in COCO contains different types of objects, which is hard to be factorised so we still need a higher β. We still set $\beta = 5$, larger β will make the images more blurry. In Figure 5.9, the images are blurry but they can still be recognized that the foreground and background have been exactly swapped.

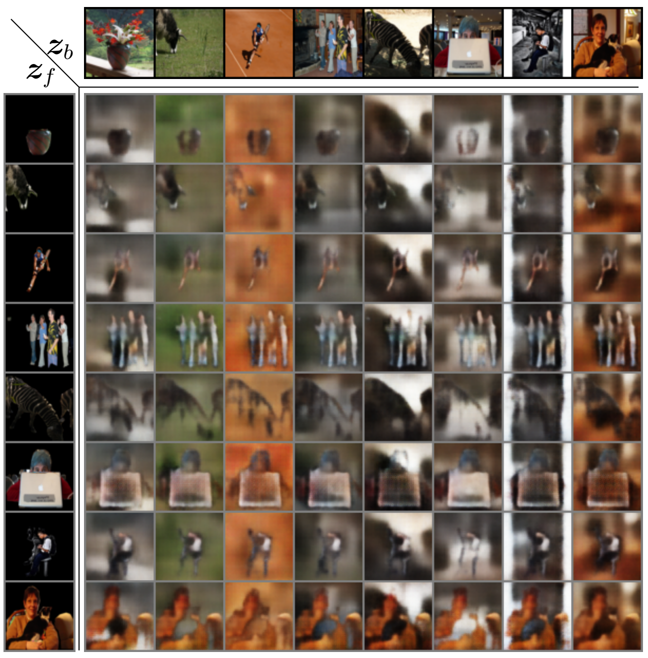


Figure 5.9: Generation by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$ on COCO. The top row is $\boldsymbol{x}$ and the first column is $\boldsymbol{f}$.

But some images are hard to tell whether the original foregrounds have been turned to the background with the same colour or just stack the foreground from other images on its own.

5.4.2 BFVAE with LSR-GAN

Due to the blurriness of images created by the VAE decoders, BFVAE cannot perform well on natural images. Thus, we use the LSR-GAN described in section 5.4.1 to generate high-quality images. We first train a new encoder E'_f before training the new generator. We find it is better to choose the same β value for VAE-B and VAE-F when we combine BFVAE with LSR-GAN. Our experiments use three datasets, the whole dataset of CUB, the training set of *Stanford cars* and 12000 images from *Stanford dogs* (120 classes $\times$100 images).

When we combine the BFVAE with LSR-GAN, we use 4 residual blocks in both encoder and decoder of BFVAE with downsampling and upsampling operation respectively. And we add one extra linear layer at the end of the encoder and the beginning of the decoder. The generator of LSR-GAN is similar to the decoder of BFVAE and the discriminator consists of 4 residual blocks with spectral norm

Table 5.3: The FID scores (lower is better) of saving resized images (left part) and feeding resized images directly (right part) at 64×64 scale.

	FID (saved)			FID (directly)		
	CUB	Cars	Dogs	CUB	Cars	Dogs
SNGAN	41.63	42.67	54.85	53.45	43.83	69.54
LR-GAN	35.22	30.30	86.67	51.85	38.80	104.45
FineGAN	24.51	31.32	**33.66**	**16.79**	23.61	**39.43**
MixNMatch	28.25	37.42	36.62	20.63	25.53	44.42
LSR-GAN	**19.12**	**18.01**	44.22	28.15	**18.99**	61.54

(Miyato et al., 2018). We apply orthogonal initialization to both LSR-GAN and the new Encoder E'_f.

We optimize the model using the ADAM optimizer(Kingma and Ba, 2015) with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We train the BFVAE and the new encoder E'_f for only 100 epochs and train the LSR-GAN for 600 epochs.

BFVAE+LSR-GAN obtains superior FID scores (Heusel et al., 2017) compared to previous models for images of size 64×64. It is well-known that the number of images and the implementation (Pytorch or Tensorflow) we use to calculate FID can strongly affect the results. Moreover, we notice an additional pre-processing operation that can make difference to the results. Commonly, when we calculate the FID scores of the dataset we need to resize the original images to the same size as our generated images, whether we save the resized images and reload them or feed the resized images to the inception model directly makes a significant difference to the FID scores we obtain. This difference is caused by the round operation when we resize the image at a different scale. In detail, the value of the original output from the network is from 0–1, and the size of the output is either 64×64 or 128×128. The input size of the inception model is 299×299. For the direct FID results, we interpolate the image in the range from 0–1. While for the saved PNG flow, we first save outputs as images which turns the value of each pixel from 0–1 to integers in 0–255, then when we load the saved images in either Pytorch or Tensorflow, the function interpolates the image at the 0–255 scale and round all the values. The two round operations make the difference in the FID score results.

Given the extreme sensitivity of FID scores to these details, it is necessary for the process of computing FID scores to be fully documented to make meaningful comparisons. Thus, we report two FID results of feeding resized images directly and saving resized images. The outputs of our model are saved as PNG image.

Table 5.4: The FID scores (lower is better) of saving resized images (left part) and feeding resized images directly (right part) at 128×128 scale.

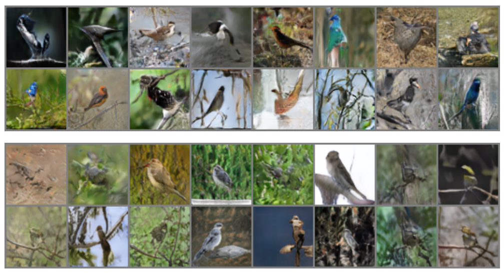
	FID (saved)			FID (directly)		
	CUB	Cars	Dogs	CUB	Cars	Dogs
SNGAN	66.43	32.11	148.50	72.15	28.94	150.70
LR-GAN	139.04	155.00	218.85	138.49	151.39	216.67
FineGAN	**21.32**	26.21	**39.65**	**13.67**	22.38	**42.72**
MixNMatch	27.25	23.86	47.13	17.63	20.62	51.58
LSR-GAN	29.56	**23.36**	75.83	27.04	**20.31**	80.40

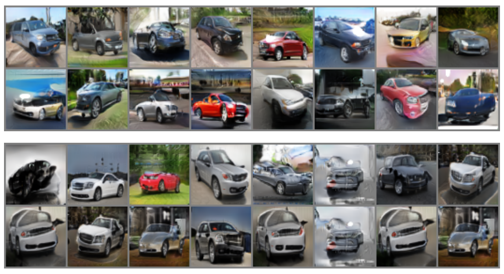
(Given these changes are not noticeable to humans it raises some concerns about how seriously we should take FID scores. However, given these are the standard metric in this field we present the results as honestly as we can.)

Quantitative results. We evaluate FID on 10K randomly generated images of size 64×64 for three different datasets. For LR-GAN, FineGAN and MixNMatch, we use the authors' publicly-available code. For a fair comparison, we use the same architecture of our LSR-GAN to train a SNGAN. We also tried replacing the original discriminator of other models with the same discriminator we use in the LSR-GAN, but it does not improve the results for either FineGAN or MixNMatch.

Thus, we present results with those models' original architectures. As shown in Table 5.3, the results in the two halves of the table are different even though the only difference is whether we saved the images or kept them in main memory (although saving images will introduce small errors due to truncation, these are not observable to a human viewer). This shows that the small difference in FID scores is not that meaningful. Comparing to previous models, our model is the best overall when we save the resized images while FineGAN is the best one otherwise. But our model has, by a considerable margin, the smallest number of parameters and training time. The size of LRGAN, FineGAN and MixNMatch's saved models are 65.6MB, 158.7MB and 336.7MB respectively. The size of our BFVAE+LSR-GAN is only 22.1MB. Also, we reported the FID scores on 128×128 scale in Table 5.4; our model can only beat previous models on the Cars dataset and gets worse results on the other datasets, FineGAN and MixNMatch perform better on larger size images because their hierarchical architecture is advantageous in learning details of images.

Alleviate mode drop. We demonstrate that the LSR-GAN can alleviate the mode drop problem by learning a latent space from a pre-trained VAE. In


(a) CUBs


(b) Cars

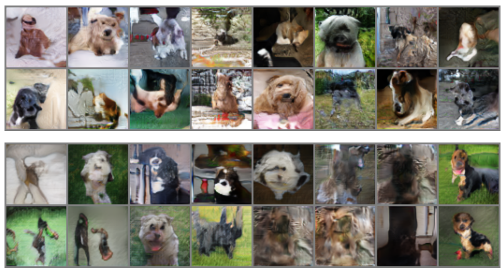

(c) Dogs

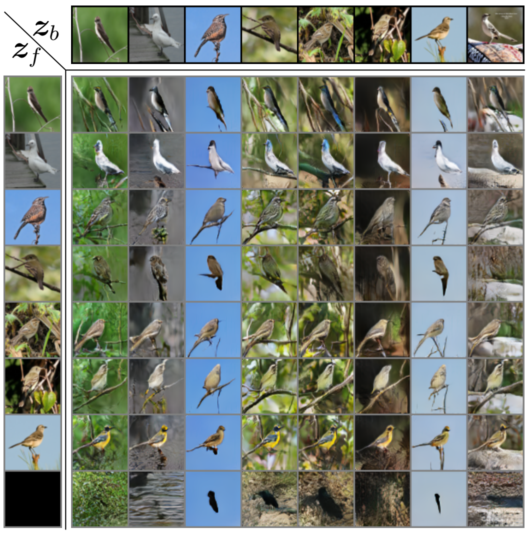
Figure 5.10: Images from SNGAN and LSR-GAN, top two rows are from LSR-GAN and bottom two rows are from SNGAN

Figure 5.10, the top two rows are from BFVAE+LSR-GAN, and the bottom two rows are from SNGAN with the same architecture. All the images are generated with a batch of random Gaussian noises, it can be observed that in the bottom two rows, there is always at least one pair of images looks visually similar. And in the images from the top two rows have more diversity.

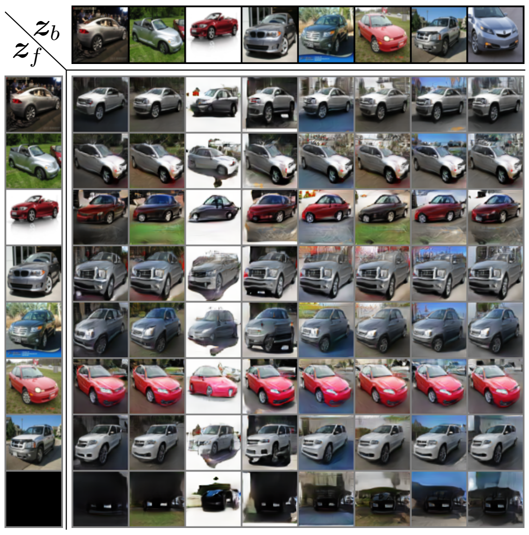
Conditional generation. In Figure 5.11 we show images generated by our method on three datasets. The top row and the first column are both the input images of two encoders. The other images are generated by combining factors from the two different images. As mentioned before, there is a trade-off between the realism and the similarity of generated images when we train the LSR-GAN, so there is a slight difference between reconstructions and input images. In the last row, some images are not pure background images. Because the discriminator has never seen images without foreground and can easily classify a pure background image as fake, this prevents the generator from generating a pure background image for some backgrounds especially pure colour background (e.g. sky). For the same reason, the background changes a little bit in the same column when the foreground is not harmonious with the background. This demonstrates how the LSR-GAN reduces its similarity when it tries to learn an approximate latent space of the pre-trained BFVAE. The nice part of this phenomena is the generator can adjust details in the background, such as the orientation of branches, to fit the foreground. For the Cars and Dogs dataset, we use bounding box instead of masks. From the results especially the fourth column in Figure 5.11(b), our model can extract the foreground from the bounding box, the foreground in the pure white background is not rectangular. But the model cannot extract background well when the input of E'_f is a blank image, this is because the objects in the Cars and Dogs datasets are much larger than CUB dataset (especially cars), the VAE-B can only learn a small part of the background information. The results of Dogs are not as good as the results of the other two datasets, because this dataset contains images like dog face or dogs with humans. This complication makes the Generator prone to generate high-quality but meaningless images (Such as dog without eyes).

New training scheme. We discovered one training scheme that can improve the pure background images but bring negative effects on the quality (FID scores) of images and make training unstable. The scheme is simple, when we train the LSR-GAN, we need to sample a batch of $\boldsymbol{z}$ from the normal distribution for each iteration. Under this scheme, we split one batch into two half batches, one-half batch contains $\boldsymbol{z}$ that consists of fixed $\boldsymbol{z}_f$ and random $\boldsymbol{z}_b$, and another half batch contains $\boldsymbol{z}$ that consists of fixed $\boldsymbol{z}_b$ and random $\boldsymbol{z}_f$. This scheme restricts that half batch image have the same foreground and another half batch images have the same background. As shown in Figure 5.12, the top one is from the original training scheme and the bottom one is from the new training scheme. It is obvious that the bottom one has less noise in the middle of background images, although it does not capture details of the backgrounds.

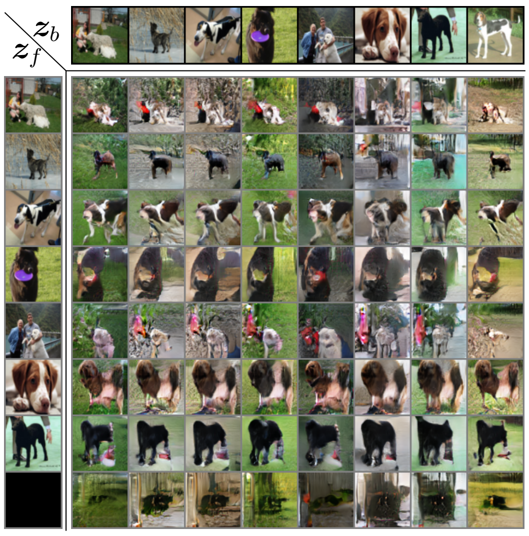
Continuous latent space. We demonstrate the continuity of the latent space learnt by BFVAE in Figure 5.13 where we show the interpolation between two images. The top-left image and the bottom-right image are the original images. Other images are the interpolations between the two images. As we move along the axes, we change $\boldsymbol{z}_f$ or $\boldsymbol{z}_b$. Both transitions between real images and fake images are smooth in the latent space, but it is obvious that even if we do not change $\boldsymbol{z}_b$ for each column, the birds (foreground) are slightly changing, this also happens for background in each row. The two reasons for this change are the same as above: firstly, the approximate latent space loses some similarity; and secondly, the discriminator can classify the unreal images like waterbirds on the branch or non-aquatic bird on the water as fake images, then the discriminator forces the generator to generate non-aquatic bird on a branch or waterbirds with



(a) CUBs



(b) Cars



(c) Dogs

Figure 5.11: Generation by concatenating different z_f and z_b. (a)(b)(c) are from 3 different datasets. The top row and the first column are both the input images of two encoders.

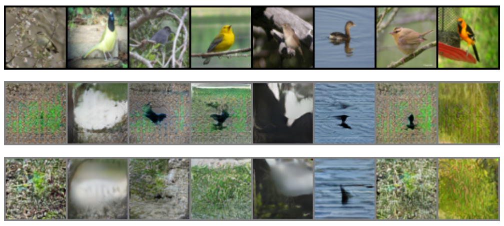


Figure 5.12: Extracting backgrounds from input images. The first row is the original image, the second row is from the normal training scheme and the last row is from the new training scheme.

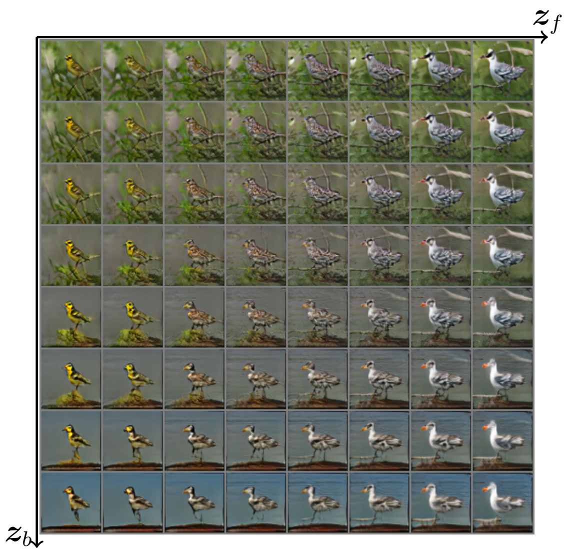


Figure 5.13: Interpolation in the latent space, the left-top image and the right bottom image are the original images, others are the interpolations between the two images. X-axis represents $\boldsymbol{z}_f$ and Y-axis represents $\boldsymbol{z}_b$.

water, which results in the slight change of both foregrounds and background even when we do not change one of $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$. This change is also a trade-off between reality and similarity.

5.4.3 Substitute Attribute for Foreground

We notice that our model is able to disentangle any kinds of attributes in the image with labels as long as we change the input of VAE-F (See details in the following context). We evaluate this on MNIST and Celeb-A (Liu et al., 2015), and compare the result with previous works.

Apart from disentangling the background and foreground in the latent space, our model is also capable to disentangle attributes such as style and content. The

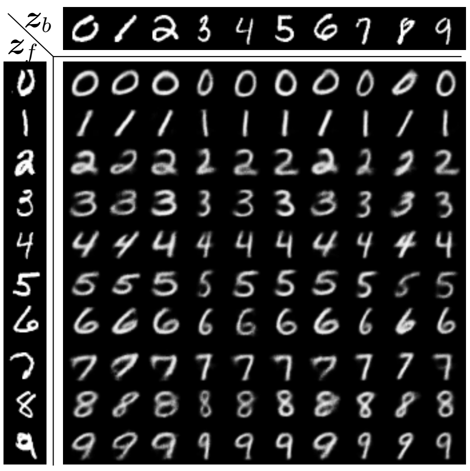


Figure 5.14: Generation by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$ on MNIST. The top row is $\boldsymbol{x}$ and the first column is $\boldsymbol{f}$.

Table 5.7: Classification results of $\boldsymbol{z}_b$. Lower is better.

	$\boldsymbol{z}_b$ train acc.	$\boldsymbol{z}_b$ test acc.
Szabó et al.	97.65%	96.08%
Mathieu et al.	70.85%	66.83%
Harsh Jha et al.	**17.72%**	**17.56%**
Ours	25.74%	26.51%

only thing we need to do is to substitute the $\boldsymbol{f}$ with images that represent each class of the single attribute. For example, if we want VAE-F to learn the information about the digit label of MNIST, we choose 10 images from 0 to 9 randomly and use these 10 images as fixed input of VAE-F. This differs from previous conditional works, we use images instead of one-hot vectors as labels. Then the VAE-B will learn a latent space about the style of images. In this setting, $\boldsymbol{z}_b$ represents the style of digit and $\boldsymbol{z}_f$ represents the value of digits. Figure 5.14 is the generation on MNIST, the same as before, the top row is $\boldsymbol{x}$ and the first column is $\boldsymbol{f}$. While the first column is fixed for all the $\boldsymbol{x}$ during training. It is obvious that the class is from the first column and the style is from the top row.

We also evaluate the classification accuracy based on the latent variables and compare the results with Szabó et al. (2017); Mathieu et al. (2016); Harsh Jha et al. (2018). The dimension of $\boldsymbol{z}_b$ is the same. The classifier is a two-layer MLP classifier with 256 hidden units. Ideally, the $\boldsymbol{z}_b$ should not contain information about the number of digits, thus, lower accuracy means better factorisation. The classification accuracy results are in Table 5.7. Our model is better than Szabó et al. (2017) and Mathieu et al. (2016) but worse than Harsh Jha et al. (2018),

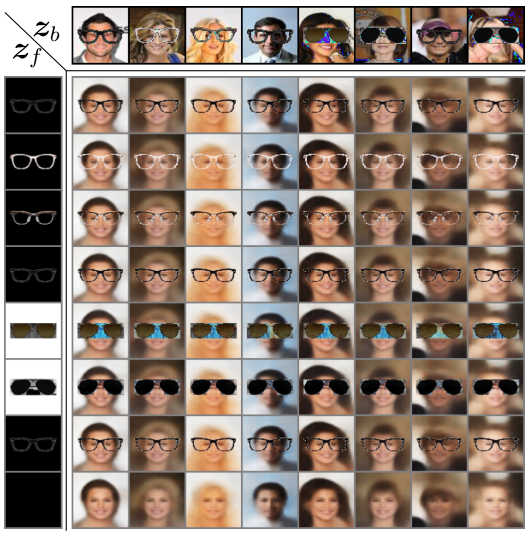


Figure 5.15: Generation by concatenating different $\boldsymbol{z}_f$ and $\boldsymbol{z}_b$. The top row and the first column are both the input images of two encoders.

although we note that in Harsh Jha et al. (2018) their model is designed to disentangle the class property while our model asks the decoder of VAE-B to extract the class information from $\boldsymbol{z}_f$ which are the latent representations of the whole images.

Also, we can replace foreground with glasses in human face images. Currently, no dataset provides human face and bounding box or mask of glasses, so we use the 10 pairs of glasses from Lin et al. (2018) and combine these glasses with images without glasses from Celeb-A. To avoid the situation that the image of black glasses is the same as a blank image, we add a binary mask as the fourth channel of the input of VAE-F. As in the previous figures, we set the last input of VAE-F as a blank image as shown in Figure 5.15. In this setting, $\boldsymbol{z}_b$ represents the human face and $\boldsymbol{z}_f$ represents the glasses. The BFVAE can generate eyes automatically when the glasses switch from sunglasses to normal glasses even the network has never seen those faces with normal glasses before, and it can also generate images without glasses when we set the foreground image into a blank image (which also it has never been seen before). The problem of generated images is the location of glasses is fixed, this could potentially be solved by using ST-GAN (Lin et al., 2018).

5.5 Conclusion

Overall, The main contributions of this Chapter are threefold:

- We propose a new VAE-based model called BFVAE that can composite the foreground and background of images and generate images by combining different factors.
- We demonstrate that BFVAE can factorize the foreground and background representation and generate high-quality images when combined with LSR-GAN. Moreover, we show our model can obtain FID scores that are comparable to the state-of-the-art model.
- We demonstrate that BFVAE is able to factorize other factors when we have additional information available for training (like class labels).

Although several works have shown great success by representing scenes using their components (Burgess et al., 2019; Greff et al., 2019; Lin et al., 2020), what defines a good object representation is still in discussion. We argue that a good object representation should also benefit the image generation task. We believe that enabling generative models to generate certain objects with random backgrounds should also be a property of good object representations. Comparing with previous models, there are several advantages of BFVAE. First, as a VAE-based model, BFVAE learns a generalized object representation from scenes, which is more similar to how human brains work on realistic scenes (Spelke, 1990). Second, BFVAE takes less training time due to the lighter architecture. Lastly, BFVAE can generate the whole images in one shot which is more efficient than generate the images part by part. However, there are also limitations. First, our model prefers to learn a representation of all the objects rather than of a single object, theoretically, this could be solved by combining with a decomposition network (Burgess et al., 2019; Greff et al., 2019). Second, BFVAE loses similarity between reconstructions and the original images after combining with LSR-GAN. Third, the generator tends to lose details of the texture in the background, this potentially can be solved by using a more powerful GAN model.

Moreover, conditional image generation tasks such as the one discussed here are useful in clarifying what we require of a good image representation. After all the ability of dreaming and imagining scenes seem to be an intrinsic human ability. In the mammalian visual system there is plenty of evidence that scenes are disentangled in different areas of the visual cortex and later re-integrated to obtain a complete understanding of a scene. Although very much simplified the BFVAE makes a step towards learning such a disentangled representation of the foreground and background.

Chapter 6

Unsupervised Representation Learning Via Information Compression

Being segmented into foreground and background is not the end of the factorisation of representation, for example, the foreground can be segmented into different objects with different attributes. In this chapter, we dive into the topic of object-centric representation learning. Unlike common generative models which learn representations for the whole images, we try to decompose the representation into different single object representations and background representations in an unsupervised way.

6.1 Introduction

In the last few years there has been a significant research effort in developing unsupervised techniques in deep learning. A very prominent example of these methods is the variational auto-encoder (VAE) (Kingma and Welling, 2013; Rezende et al., 2014) that attempts to find latent representations to efficiently encode a dataset. A drawback of VAEs is that they represent the entire image. This makes sense for images that represent a single object, but it is unlikely to lead to an efficient representation for many real-world images that depict multiple objects. Following the development of VAEs there have been a number of attempts to use unsupervised techniques for object location and segmentation within an image (Burgess et al., 2019; Engelcke et al., 2021, 2020; Greff et al., 2017; Lin et al., 2020; Locatello et al., 2020).

In this chapter, we will explore the use of a minimum description length cost function together with an information bottleneck to achieve unsupervised image understanding and object discovery in simple scenes. The evidence lower bound (ELBO) of a VAE can be interpreted as a description length where the KL-divergence corresponds to a code length of the latent representation and the log-probability of the reconstruction error as the code length of the residual error (i.e. the error between the reconstruction and the original image). In our approach, we will use multiple glimpses of an image corresponding to a sequence of bounding boxes. These areas chosen by bounding boxes are resized to 8×8 patches and passed to a variational auto-encoder. The full reconstruction is built up by adding together the reconstructions from the VAE. This is done iteratively with each patch providing a correction between the current reconstruction and the true image. A spatial transformer is fed the current residual error and used to select the next bounding box. The overall cost function is the cost of the latent codes for all the bounding boxes together with the cost of the final residual error. We stop when the cost of transmitting the latent code is higher than the reduction in the cost of transmitting the residual error. The spatial transformer and VAE are trained end-to-end by minimising the description cost of the images in a dataset.

Although we expect our approach to be very different to human eye movement nevertheless, there is a rough correspondence due to the restricted size of the fovea requiring multiple fixations of an image around areas of high interest and possible interpretational ambiguity (Stewart et al., 2020). We deliberately avoid building in any bias towards glimpsing complete objects, however, as we will see later, at least, in simple scenes this behaviour emerges. Our aim is not to build a state-of-the-art unsupervised object detector, but rather to investigate how a minimal implementation using minimum description length and an information bottleneck will glimpse images. As we will demonstrate these glimpses can sometimes be used to solve downstream tasks that have competitive results with much more sophisticated approaches.

6.2 Model

In this section we introduce our model in details.

6.2.1 Glimpsing Network

In our approach we iteratively build up a reconstruction. We use an glimpsing network that consists of a spatial transformer network that proposes the location of a bounding box and then resamples the image within that bounding box to create (in our case) a low-resolution patch of the original image. In our network at each iteration, the spatial transformer network is given the residual error between the current reconstruction and the input image. The glimpsing network selects a bounding box, we show the diagram in Figure **??**, where $\mathbf{z}^{where}$ is the parameter that indicates size and location of the bounding box. Then the residual error, $\boldsymbol{\Delta}(t) = \boldsymbol{x} - \hat{\boldsymbol{x}}(t)$, within the bounding box is down-sampled to an 8×8 patch (with 3 colour channels) and fed to a VAE. The VAE produces a latent code $q(\boldsymbol{z}|\boldsymbol{\Delta}(t))$. This is used to create a reconstruction using the standard reparameterisation trick, which is then resized to the size of the original bounding box. This results in a reconstructed correction, $\hat{\boldsymbol{\Delta}}(t)$, which is then added to the reconstruction to obtain an new reconstruction $\hat{\boldsymbol{x}}(t+1) = \hat{\boldsymbol{x}}(t) + \hat{\boldsymbol{\Delta}}(t)$ (note that $\hat{\boldsymbol{\Delta}}(t)$ only has non-zero values within the bounding box selected by the glimpsing network).

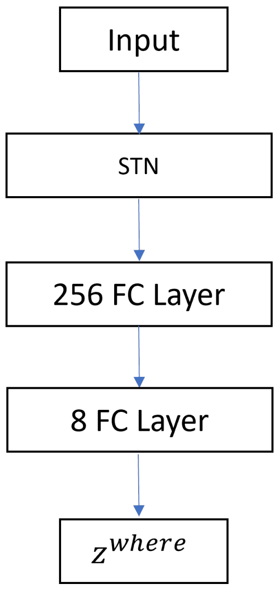


Figure 6.1: The diagram of glimpsing network, STN represents a spatial transformer network.

We use the standard information theoretic result that the cost of transmitting a random variable with a distribution $q(\boldsymbol{z}|\boldsymbol{\Delta}(t))$ relative to a distribution $p(\boldsymbol{z})$ is given by the KL-divergence (or relative entropy) $D_{KL}(q\|p)$. In our case we use the standard latent encoding $q(\boldsymbol{z}|\boldsymbol{\Delta}(t)) = \mathcal{N}\Big(\boldsymbol{z}|\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma})\Big)$ and standard prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\mathbf{0}, \mathbf{I})$. The cost of communicating the residual error is given by

$-\log\Big(p(\boldsymbol{x}-\hat{\boldsymbol{x}})\Big)$. We use the standard assumption that the residual errors are independent at each pixel and colour channel and normally distributed with mean 0 and standard deviation σ. To minimise the communication cost we choose σ^2 to be the empirical variance. In this case the cost of communicating the residual errors is, up to a constant, equal to $N\log(\sigma)$, where N is the number of pixels times the number of colour channels.

Note that provided both the sender and receiver have the same VAE decoder we can communicate an image by sending the set of latent codes (plus the position of the bounding box) and the residual error. (We assume that the dataset we are sending is so large that the cost of transmitting the VAE decoder is negligible). To train our spatial transformer and VAE we attempt to minimise this communication cost for a dataset of images.

A critical component of our approach is that we use an information bottleneck. That is, we down-sample our bounding box and feed this to a VAE. We illustrate the effect of this for images taken from the CLEVR (Johnson et al., 2017) dataset in Figure 6.2. In the first row we show the original images. In the second row we show the reconstruction error after down-sampling the whole image to an 8×8 image and then up-sampling using bilinear interpolation to the original size (64×64). In the third row we show the reconstruction loss if we use a vanilla VAE without down-sampling. Finally, we show the reconstruction loss when we down-sample to 8×8 encode that through a VAE and then up-sample the VAE reconstruction.

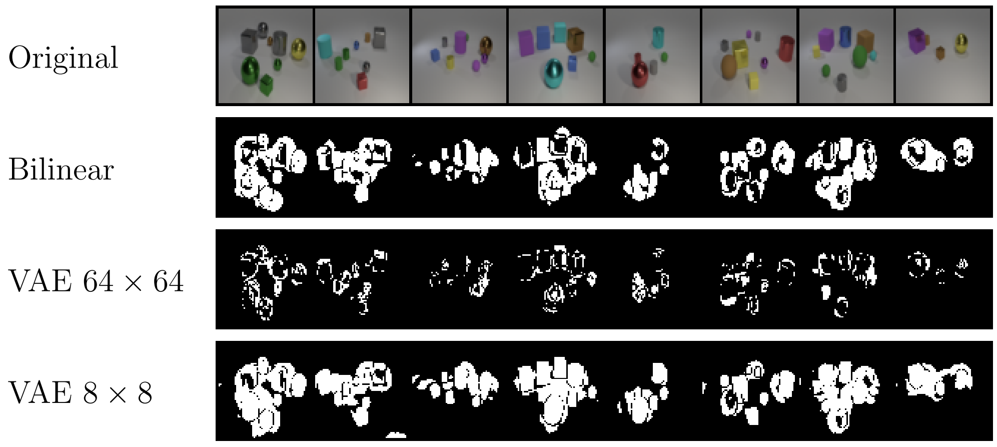


Figure 6.2: The error maps of different compression methods. The first row is the original image, the next three rows are the error maps of bilinear interpolation, a VAE for 64×64 image and a VAE for 8×8 image which takes the downsampled version of the original images as input

Figure 6.2 illustrates that due to the bottleneck we have high reconstruction error around the objects in the image. The original error is hard to find the difference visually, thus, we turn the error map into a binary map. By setting a threshold, if the error is higher than 0.15, the pixel value is set to 1, otherwise 0. The threshold can be any value, but too high or too low cannot reflect the difference between different parts in the image, and we found that 0.15 is a suitable value to show the observation. This error will drive the glimpse model towards parts of the scene of high complexity.

6.2.2 Object Discovery Based on Error Maps

Here we start the test without adding VAE into the network. The network consists of 3 layers of CNN and 2 layers of fully-connected network. There will be K iterations for each image. For each iteration, the network decomposes the input into a latent $\mathbf{z}_i^{where}$, which denotes the scale and position of each glimpse, and i is the index of iterations. For the latent variable $\mathbf{z}_i^{where}$, we assume it as Normal distribution: $\mathbf{z}_i^{where} \sim \mathcal{N}(\mu^{where}, \sigma^{where})$, where the Normal parameters are hyper-parameters. The input is changing over the iterations. For the first iteration, we take the error of the absolute value between the original image $\mathbf{x}$ and a degraded image $\mathbf{x}'_i$ (where $i = 0$) which has been downsampled and upsampled by bilinear interpolation method. The original size of the image is 64×64 and the downsampled size is 8×8. For the following iterations, the $\mathbf{x}'_i$ is changing. We first crop out the glimpse area and also downsample and upsample the area, this ensures the glimpse to be lossy. And the lossy glimpse is then stitched on $\mathbf{x}'_{i-1}$ via the spatial transformer (Jaderberg et al., 2015) to obtain the $\mathbf{x}'_i$. The network is updated for each iteration, and the loss function is a simple mean square error, where

$$\boldsymbol{L}_i = \|\mathbf{x} - \mathbf{x}'_{i+1}\|^2.$$

The last $\mathbf{x}'_{i+1}$ is only for the loss calculation and will not be fed into the network. The idea behind the loss function is succinct. Since the error maps show the high error areas belong to objects, the mean square error can lead the model to find the area that reduces the error most, especially when there is a bottleneck during the image compression process. Otherwise, the model will simply produce a bounding box that covers the whole image to make the error to be 0 at the first iteration. For the CLEVR dataset, the maximum number of objects in images is 10. So we set $K = 9$. Figure 6.3 shows the results of three different kinds of images: the first two rows show the image with small number but big size objects, the second

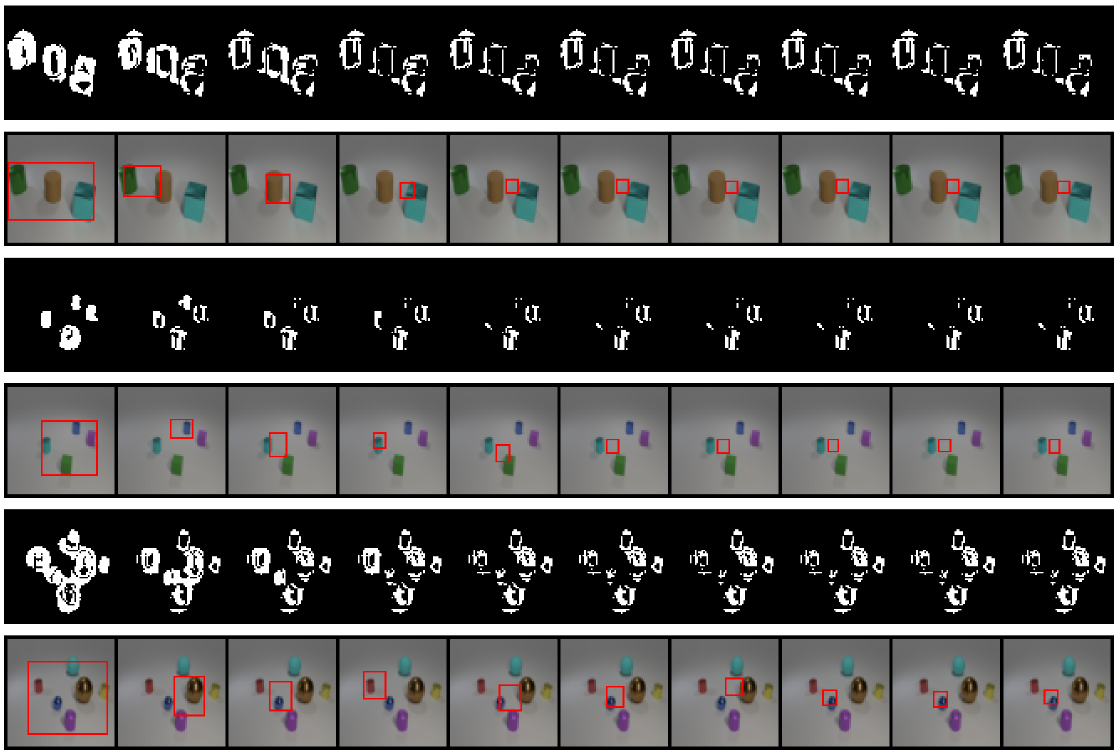


Figure 6.3: The results of the simple network. The binary images are the error map with a threshold 0.15 before each iteration and the colorful images below with bounding boxes are the corresponding results

two rows show the image with a small number and small size objects and the last two rows show the image with a big number and mixed size objects. The binary images are the error map with a threshold 0.15 (which is the same as Figure 6.2) before each iteration and the colourful images below with bounding boxes are the corresponding results. We draw a bounding box to show the area indicated by $\mathbf{z}_i^{where}$. For all the images, the first glimpse always covers almost all the objects. There is a trade-off in the choice of the scale of glimpses. We can assume the error for each pixel is the same during the image compression and define the error of each pixel as E_s, where the s is the scale of the glimpse. $E_s = 0$ when s equals the downsampled size (8×8 here) and E_s is maximum when s equals the image size, which we can define as E_{ini} since the input of the first iteration can be regarded as a glimpse that covers the whole image. Then the total error after the first iteration should be

$$(N - n_s)E_{ini} + n_s E_s.$$

N is the number of pixels of an image and n_s is the number of pixels of the glimpse. The model tends to produce a big glimpse at the first step when the difference between E_{ini} and E_s is small. Moreover, the pixels of objects are supposed to have higher errors, so the big glimpse area always covers the foreground area rather than the background area. Although the model can find the foreground at the first glimpse, the following glimpse is not accurate at discovering objects. Also, the model fails at discovering all the objects inside the foreground area in some iterations. But the reasons for the failure are different for those images. In general, the bounding boxes that have been drawn in the centre in the latter iterations can be regarded as a default bias of the spatial transform network, which is acceptable if all the objects have already been discovered. For the first two rows, the error map covers three objects well at the beginning. After the big glimpse, the error of the right object is too small to be recognised by the network, which is the same as the top object in the third two rows. For the second and the third two rows, the model failed at discovering the bottom object even if the corresponding area in the error map is still relatively larger than other objects that have been discovered by the model. The good side of the last row is the model tries to find objects in descending order according to the size of the objects.

6.2.3 Model Architecture and Training

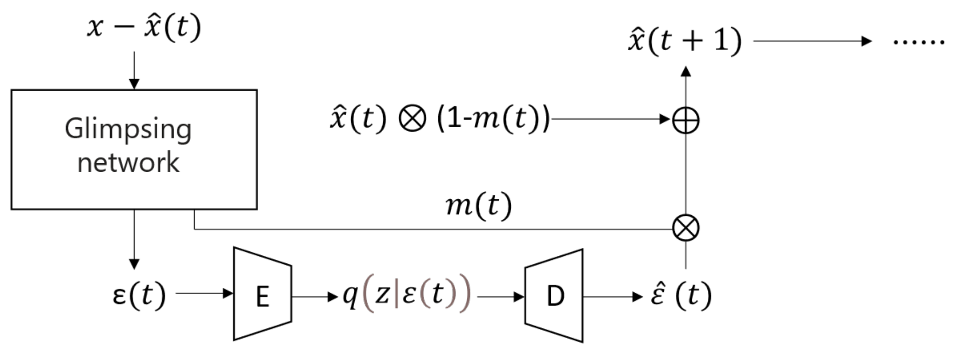


Figure 6.4: The diagram of one iteration in SAID, where we omit the operation from position code $\boldsymbol{u}$ to $\boldsymbol{\epsilon}(t)$.

In keeping with our philosophy or keeping our model simple. Our glimpsing network consists of 3 layers of CNN and 2 layers of MLP. Each image is presented K times. At each presentation the input to the glimpsing network is the residual error, $\boldsymbol{\Delta}(t) = \boldsymbol{x} - \hat{\boldsymbol{x}}(t)$. The output of the glimpsing network provides the coordinates of the bounding box that is then resampled to create an 8×8 RGB image patch, $\boldsymbol{\epsilon}(t)$ that is used as the input to a standard VAE. In Figure 6.4, we illustrate the structure of one iteration in our model. In the experiment, $\boldsymbol{\epsilon}(t)$ is the original image within the bounding box rather than the residual error (which is better for the downstream tasks). This also makes $\hat{\boldsymbol{x}}(t+1) = (1-\boldsymbol{m}(t))\otimes\hat{\boldsymbol{x}}(t)+\boldsymbol{m}(t)\otimes\hat{\boldsymbol{\epsilon}}(t)$. where $\boldsymbol{m}(t)$ is a mask equal to 1 in the bounding box and 0 otherwise that is obtained from the position code, and $\otimes$ denotes elementwise multiplication. Additionally, $\boldsymbol{m}(t)$ can be an alpha mask produced by the decoder of the VAE, the importance of the alpha mask will be investigated in the ablation study. The VAE reconstruction, $\hat{\boldsymbol{\epsilon}}(t)$, is reshaped to the original bounding box to create a correction to the reconstruction. Both encoder and decoder contain 4 layers of CNN with ReLU and 2 layers of MLP. We used a 10-dimensional latent representation. To train the VAE we minimise the standard ELBO loss function

$$\mathcal{L}_{vae} = -\log\big(p(\boldsymbol{\epsilon}(t)|\hat{\boldsymbol{\epsilon}}(t))\big) + D_{KL}\left(q(\boldsymbol{z}(t)|\boldsymbol{\epsilon}(t))\middle\|\mathcal{N}(\boldsymbol{x}|\mathbf{0},\mathbf{I})\right) \tag{6.1}$$

where $q(\boldsymbol{z}(t)|\boldsymbol{\epsilon}(t))$ is the distribution describing the latent code. Note that the reconstruction $\hat{\boldsymbol{\epsilon}}$ is generated by sampling a latent vector from $q(\boldsymbol{z}(t)|\boldsymbol{\epsilon}(t))$ and feeding this to the decoder of the VAE.

Recall the glimpsing network predicts the position of the bounding box. We encode this through a position parameter $\boldsymbol{\mu} = (\mu_x, \mu_y)$ and width parameters $\boldsymbol{\sigma} = (\sigma_x, \sigma_y)$. Adding a bounding box at iteration t reduces the cost of communicating the residual error by

$$\mathcal{L}_{res} = -\log\big(p(\boldsymbol{\Delta}(t)\big) + \log\big(p(\boldsymbol{\Delta}(t-1)\big) \tag{6.2}$$

but requires an additional cost

$$\mathcal{L}_{kl} = D_{KL}\left(q(\boldsymbol{z}(t)|\boldsymbol{\epsilon}(t))\middle\|\mathcal{N}(\boldsymbol{x}|\mathbf{0},\mathbf{I})\right). \tag{6.3}$$

Summing these two terms provides a loss function for the network that measures (up to an additive constant) the reduction in cost of communicating the image using the new latent code describing the correction to the residual error. Note that in using the trained network when this difference becomes positive then we stop glimpsing. In practice, training the glimpsing network with just these term leads to poor performance. To improve this we assume the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ represent parameters of a normal distribution where we regularise them with an additional loss term

$$\mathcal{L}_{pos} = D_{KL}\left(\mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu},\boldsymbol{\sigma})\middle\|p(\boldsymbol{u})\right). \tag{6.4}$$

This acts as a regularisation term for $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. We call the whole network *Spatial*

Attention for Information Detection (SAID). We also considered two modified network architectures. In the first we learn an alpha channel so that the corrections are only applied to particular regions within the bounding box. In the second case we include an additional channel as the input, which we called *scope*. This is motivated by MONet Burgess et al. (2019) that uses the same idea. This scope channel can force the network to look at the area that has not been discovered even when there are areas that have been discovered before contain high error pixels. Initially we set $\boldsymbol{s}(0) = \mathbf{1}$. Recall that the mask, $\boldsymbol{m}(t)$, is defined to be 1 in the bounding box and 0 elsewhere. Then the scope is updated as

$$\boldsymbol{s}(t+1) = \boldsymbol{s}(t) \otimes (\mathbf{1} - \boldsymbol{m}(t)) \tag{6.5}$$

so the scope will be 0 where a bounding box has been proposed and 1 otherwise. Note that in our approach we learn after every iteration rather than build up a gradient over multiple iterations. This makes the learning problem for our system much simpler than methods such as MONet that applies the loss function only after making a series of bounding box proposals. We investigate the role of the alpha channel and scope in ablations studies described in Section 6.4.4.

6.3 Differences between SAID and Related Work

The Attend-Infer-Repeat (AIR) (Eslami et al., 2016) and Spatially Invariant Attend, Infer, Repeat (SPAIR) (Crawford and Pineau, 2019) infer the object representation as "what", "where" and "pres" variables, where SPAIR infers an additional variable "depth". While SAID only glimpse "what" and "where". Rather look at the image cell by cell as AIR and SPAIR do, SAID glimpse the area with high error directly.

PermaKey (Gopalakrishnan et al., 2021) is a model that aims to extract object keypoints from images which takes the error map between two feature maps as input. While SAID takes the error map between two images as input to infer the position of objects.

6.4 Experiments

In this section we attempt to quantify the performance. Recall that the objective is to find glimpses that allows an image to be efficiently encoded through a bottleneck, so it might fail at finding a complete object for single glimpse. We therefore compare our model to two models, SPACE (Lin et al., 2020) and SPAIR (Crawford and Pineau, 2019) designed to find multiple objects in an image. To evaluate our model, we use three commonly used datasets in unsupervised object-centric representation learning models. The first dataset is Multi-dSprites, which is developed from dSprites (Matthey et al., 2017) and each image consists of different shapes with different colour, the maximum number of objects in this dataset is 4. The second dataset is Multi-MNIST, which is developed from MNIST (LeCun, 1998). For Multi-MNIST, we render MNIST digits of size 20×20 on a 84×84 canvas. The maximum number of objects in this dataset is 5. The last dataset is CLEVR (Johnson et al., 2017), unlike other models that follow the setting of IODINE (Greff et al., 2019) to re-generate an easier dataset CLEVR6, we use the original CLEVR dataset directly. For all the datasets we resize images into 64×64.

We trained our network, SAID, using the ADAM optimizer(Kingma and Ba, 2015) with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We train our model for 200 epochs for all the datasets.

6.4.1 Quantitative Comparison

As a first test we consider object location on the Multi-MNIST dataset. Although object location is not an objective of our model, for the Multi-MNIST dataset the hand-written characters are well separated and so a natural choice of bounding box would be around each character. There are however situations where the bounding box only covers part of the objects or covers more than one object, as shown in Figure 6.9. We perform a quantitative comparison of results on Multi-MNIST using two metrics with SPACE and SPAIR. The two metrics are the Average Precision (AP) and the Object Count Error. AP is a commonly used metric in object detection task (Everingham et al., 2010) and the Object Count Error is the difference between the number of objects predicted by the models and the true number of digits (Crawford and Pineau, 2019). For SPACE and SPAIR, we set the grid size as 3×3. For our model, we use the index of the iteration that the KL

Table 6.1: Comparison with respect to the quality of the bounding boxes in the Multi-MNIST. Results are averaged over 5 random seeds.

	AP IoU Threshold $\in$ [0.1:0.1:0.9]	Object Count Error
SPAIR	0.501 ± 0.004	0.261 ± 0.032
SPACE	0.310 ± 0.041	0.031 ± 0.005
SAID	0.503 ± 0.005	0.810 ± 0.040

term of the VAE is smaller than improvements on the mean squared error as the number of objects, and we set $K = 5$ in training and $K = 9$ in AP measurement.

As shown in Table 6.1, our model achieves similar AP with SPAIR, SPACE has the worst AP. However, this result on Multi-MNIST does not reflect the ability of object detection. The reason is the ground truth bounding box we are using for MNIST in the AP calculation is larger than the digits, which degrades the AP result when the model returns a smaller bounding box while it still detects the objects well. In Figure 6.5, each row is a concatenation of 5 images. The first row is the ground truth bounding box we can obtain in Multi-MNIST dataset, this brings disadvantages to the SPACE model in the AP calculation since the third row shows the bounding box of SPACE model is tighter than the ground truth and still maintain the accuracy. Our model does not perform well on the Object Count Error, since there is no $\mathbf{z_{pres}}$ in our model and the objective of our model is to find high error areas rather than find objects. Objects of high complexity are often selected more than once leading to a count error. We note that our stop criterion is applicable to any image and was not chosen to given an accurate object counts.

6.4.2 Downstream Task

Obviously a glimpsing model is only of value if the glimpses can be used in some downstream task. Here we consider the task of returning the sum of all the digits in a Multi-MNIST image. Each image contains 5 digits. This is a task that has previously been used to test unsupervised multi-object detection. We show the results on 80k training set and 20k test set. We compare our results to the SPACE and SPAIR models. We run implementation of all three models to ensure consistency.

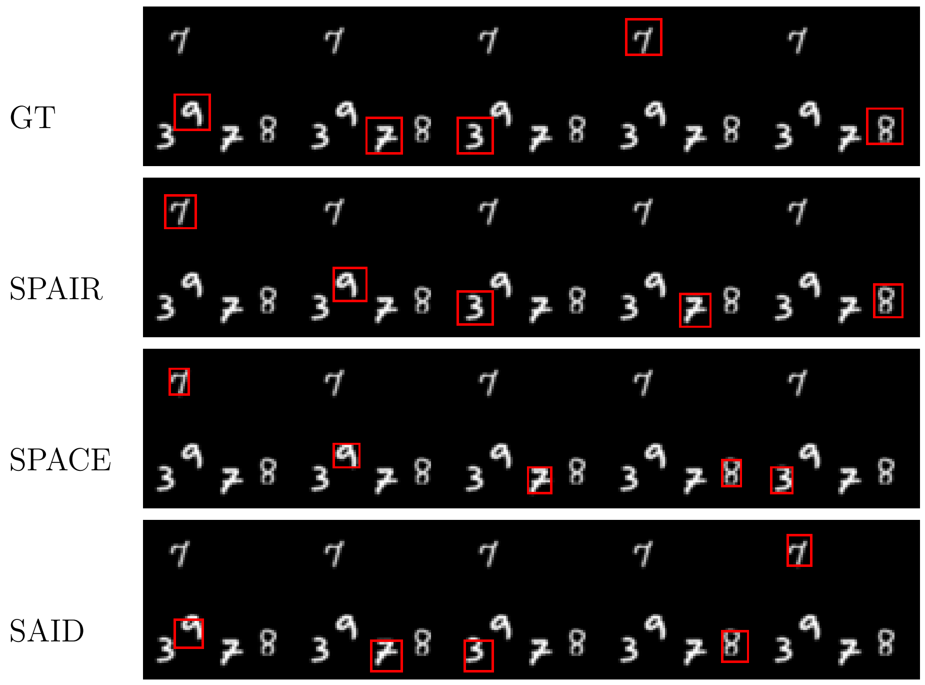


Figure 6.5: Qualitative comparison between the bounding boxes found for different models. GT is the ground truth bounding boxes, SPAIR and SPACE are models developed by other authors while SAID is our model.

For SPACE and our model, we use the same architecture of the encoder, and the channel of the latent space is 10. For SPAIR, we observered that the model tended to collapse at an early stage when we using these parameter setting. Thus, we maintain the original architecture of the encoder, but increase the channel of the latent space to 50 and the input size of the encoder is 15×15 rather than 8×8, which potentially brings benefits to the capacity of the encoder.

To compute the sum of the digits we construct a 3 layer MLP using the latent codes, $\boldsymbol{z}(t)$, as inputs. The output of the MLP has a single output which we train to have the same numerical value as the digit (that is, the digit 3 should have an output of 3). When testing we add the outputs from each glimpse and then round that number to the nearest integer. We trained the MLP for 100 epochs. This is considerably simpler than the set up described in Crawford and Pineau (2019) who use an LSTM to perform the addition. We note that our method explicitly treats the glimpse for this problem as a set, where the result is invariant to the ordering of the glimpses. We also demonstrate the results by feeding the ground truth bounding box into the encoder rather than the predicted bounding box, which we denote as GT. GT provides an estimated upper-bound on the performance we could achieve.

Table 6.2: The performance on the downstream task of summing the digits in the images in Multi-MNIST is shown using the ground truth (GT) bounding boxes and bounding boxes found by SPAIR, SPACE and our network SAID. The results are computed by averaging over 5 runs with different random seeds.

	Fixed		Unfixed	
	Train	Test	Train	Test
GT	30.3% ± 1.2%	29.2% ± 1.1%	97.5% ± 0.9%	92.3% ± 1.1%
SPAIR	25.8% ± 2.2%	24.0% ± 2.1%	24.6% ± 1.5%	22.0% ± 1.1%
SPACE	15.1% ± 2.5%	14.4% ± 2.2%	42.3% ± 1.5%	30.1% ± 1.2%
SAID	22.3% ± 1.8%	21.3% ± 2.0%	57.8% ± 1.6%	31.9% ± 1.5%

Table 6.2 shows the results of four models in 2 different conditions. Fixed represents a frozen encoder during the classifier training while Unfixed represents an encoder tuning with the classifier. Due to the architecture issue, SPAIR performs best under Fixed but worst under Unfixed. Our model performs better than SPACE in both situations but there is still a huge gap between our model and the ground truth bounding box.

We investigate the reason for the poor performance on this downstream task under the Fixed situation. In Figure 6.6, we show the prediction results of each digit within one image for the ground truth bounding box. For each row, the first image is the original image which is 64×64, and the following digits are upsampled from 8×8. The number under each digit is the prediction result from the downstream task. Most of the digits can get reasonable predictions. From the two examples, we can observe two reasons for the poor performance of all four settings, especially GT (which is supposed to be an upper bound). First, the 9 in the first image is similar to 5 after downsample to 8×8, it cannot guarantee if it is a 5 or 9 without any additional information. This is also the reason that the results can get a boost under the Unfixed situation. By using the label as the additional information, the performance can be enhanced even with 8×8 input images. Second, the first two digits in the second image are 2 and 8, the predictions are 2.37 and 8.33 which are correct if we round the two results separately. However, the training process is adding 2.37 and 8.33, which returns 10.70 that can be rounded to 11, or after adding 7.63 to get 18.33 which will also get an additional 1 for the results. Since we do not have the label for every single digit, we cannot try to predict each digit separately in this simple downstream test setting. By setting a larger bottleneck can potentially improve the downstream tests while it can also impact the quality of the predicted bounding box.

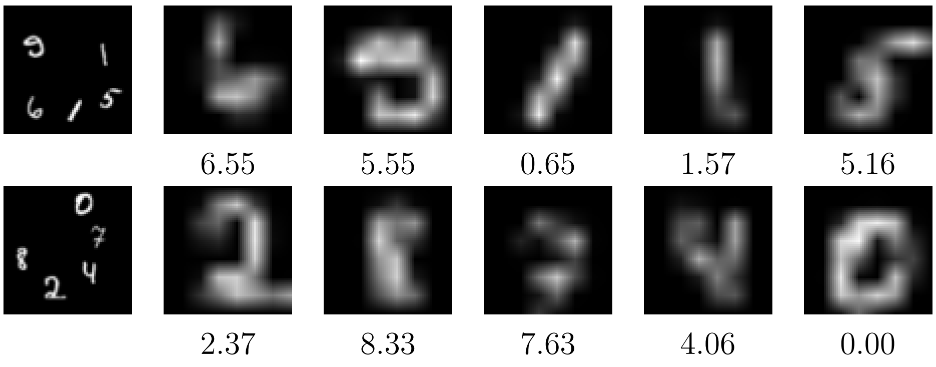


Figure 6.6: The prediction output of each digit within one image (GT). The value under each image is the prediction value of the corresponding image.

Rather than only looking at the ground truth set, we also investigate the reason for the poor performance of our SAID model. Apart from the two reasons mentioned before. There is another reason for the bad performance of SAID model. We show the predicted bounding boxes for one image in Figure 6.7, the first row is the first 5 predictions from SAID, and the second row is the first 8 predictions from SAID. The first 5 predictions are the inputs that we use for the downstream task while we do not have any presence parameters to indicate whether the bounding box has covered an object. There are a few images that SAID models cannot find the 5 digits in the first 5 predictions, this brings negative effect for the downstream test, although our model can still find the correct digits in the following predictions, we cannot point it out automatically, which is also a drawback of our model.

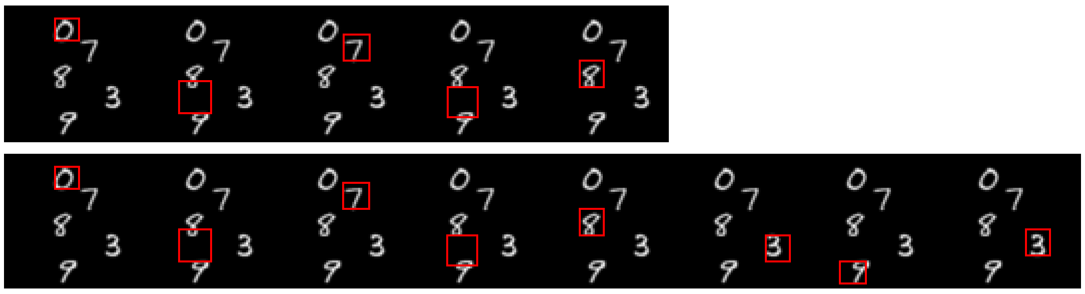


Figure 6.7: The predictions of bounding boxes.

6.4.3 Generalization

Our model uses a very general principle that we believe can be widely used in different contexts. To explore this we look at out-of-distribution generalisation.

That is, when we train on one dataset (here we use CLEVR) and use the model on a different dataset.

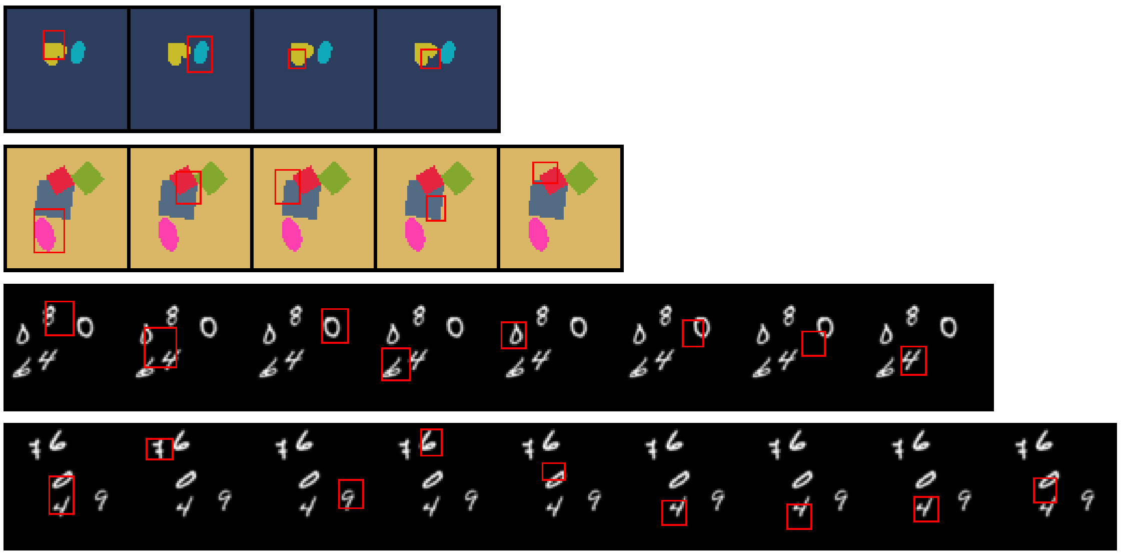


Figure 6.8: Examples of bounding boxes found on the Multi-dSprites and Multi-MNIST dataset are shown for a network trained of the CLEVR dataset.

We test the network on the Multi-dSprites and Multi-MNIST datasets. We set the maximum number of iteration $K = 10$ which is the same as we used when training CLEVR, but we stop the iteration when the KL is larger than the reduction in code length of the reconstruction error. Results are shown in Figure 6.8. The first two rows are the results for Multi-dSprites, the model trained on CLEVR can stop at a reasonable iteration (Close to the number of objects). But the model tends to infer more times on Multi-MNIST, we assume it is because the binary images are simpler to be transmitted than the RGB images. The VAE trained on CLEVR can transmit binary images efficiently no matter if the bounding box covers the digits correctly. The model can still locate the area of the objects although some of the bounding boxes failed at covering one object.

6.4.4 Ablation Study

The results we show in the previous section is trained with a scope channel and the input is the difference between the original image and a lossy image that has been downsampled to 8×8 and then upsampled to the original size through bilinear interpolation. Also, we have used an alpha mask instead of a binary mask when we blend the images. In this section, we show the importance of the scope channel and the alpha channel. We also show how the lossy image can affect the results when we use different methods to get the lossy version.

Table 6.3 shows the results on AP and Object Count Error when no alpha and no scope mean we remove the alpha channel and scope channel respectively. VAE8 means we use a 8×8 VAE to reconstruct the image after the downsample interpolation, as shown in Figure 6.2. It can be observed that the alpha channel does not make a huge difference to the model while the model gets a degraded performance when we remove the scope channel. Also, the model performs worse when we use a VAE to obtain the lossy version of the images. This is because after using a 8×8 VAE, the error map tends to cover more background.

Table 6.3: The performance of ablation studies carried out on the Multi-MNIST dataset. The average precision in the detection of bounding boxes is presented together with the error in the count of the numberof objects. Different versions of SAID are compared.

	AP IoU Threshold $\in$ [0.1:0.1:0.9]	Object Count Error
SAID (no alpha)	0.490 ± 0.008	0.731 ± 0.040
SAID (no scope)	0.341 ± 0.006	1.710 ± 0.110
SAID (VAE8)	0.452 ± 0.008	1.101 ± 0.031
SAID	0.503 ± 0.005	0.810 ± 0.040

6.4.5 CLEVR and Multi-dSprites

In the Multi-MNIST the objects are of approximately the same size and do not suffer from occlusion. Clearly, this is very different to real images. To explore these issues we tested our models on CLEVR and Multi-dSprites dataset.

Figure 6.9 shows the results on CLEVR dataset and Figure 6.10 shows the results on Multi-dSprites dataset, we set $K = 10$ for CLEVR and $K = 4$ for Multi-dSprites respectively. We stop the iteration when the KL divergences is greater than the reduction in transmitting the residual error. In Figure 6.9, the first two rows, the size of objects is close to 8×8, the model can stop at the correct iteration and all the bounding boxes covers different objects, although the bounding boxes are less accurate. For the last row, the size of objects is much larger than 8×8. Our model tends to infer more than the number of objects, this is due to the limited bottleneck failing at transmitting the whole object at the first transmission. But for those big objects, the model returns more accurate bounding boxes compared

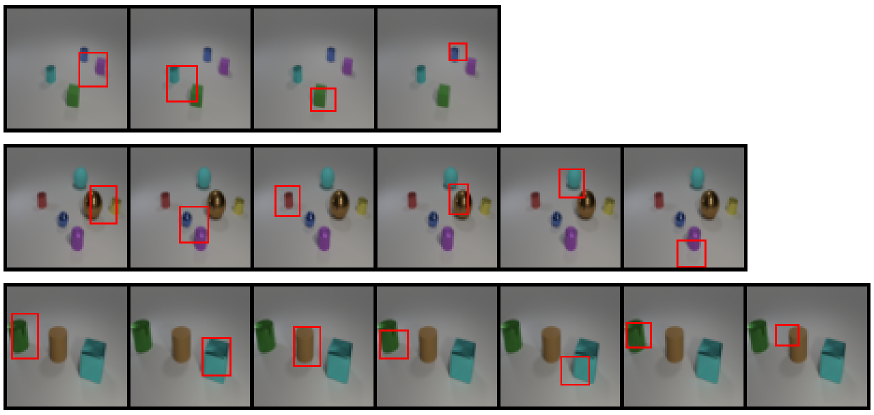


Figure 6.9: Examples of bounding boxes found by SAID on the CLEVR dataset.

to the first two rows. Since big objects tend to show a big error. In Figure 6.10, our model does not stop after in a reasonable number of iterations. It has the same issue as the CLEVR dataset that the bounding box tends to cover parts of shapes rather than the whole object. Also, our model cannot deal with overlaps properly. In part we attribute this failure to the weakness of the glimpsing network which struggles with finding bounding boxes of very different sizes.

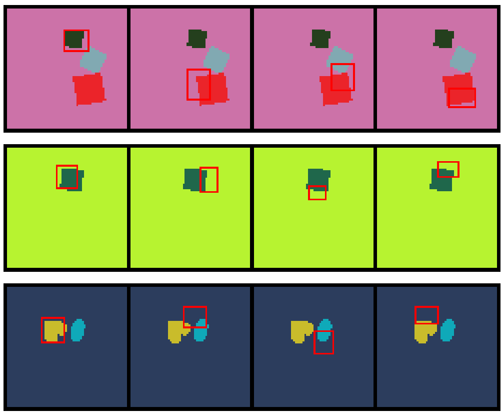


Figure 6.10: Examples of bounding boxes found on the Multi-dSprites dataset.

6.5 Conclusion

Information compression, which we implement as image compression in experiments, provides a powerful tool for recognising structure in complex data sources. In this work, we have combined this with an information bottleneck to produce a glimpsing network that encodes images through a series of glimpses. By feeding the network the current residual error we can generate a series of bounding box proposals around parts of the image with high uncertainty in its reconstruction. We combine this with a VAE that can learn the common structures within an image (e.g. objects, or rather typical residual errors associated with objects). As the bounding boxes are rescaled, the structures being learned by the VAE are translation and scale invariant. We have shown that following these principles it is possible to train a very simple network (with only 3 layers CNN, 2 layers fully-connected layers and a small vanilla VAE that takes inputs with a 8×8 size) that has comparable performance on object discovery tasks to much more complex networks like SPAIR and SPACE which asks a deeper architecture. Our objective was to test as simple a network as possible to prove the power of this learning paradigm.

SAID has shown the ability of learning object-centric representation in an unsupervised manner, it encodes different area of the image in an information descending order. The results on the downstream task show the advantage of this learning paradigm. However, it needs constraints to ask each iteration to glimpse a complete object if the input gets complicated, which we leave it as future work.

Having shown that this works, we believe there are many directions where the model can be improved. One clear weakness of the model is that the simple architecture of the glimpsing network used to propose bounding boxes was hard to train reliably on different datasets with different sized objects—a problem noted by other researchers (Lin et al., 2020). This possibly suggests that the CNN architecture that we used is not well suited to identifying interesting bounding boxes. We believe that a network better adapted to solving this problem is likely to improve performance and robustness of our approach. This work can potentially combine with the BFVAE introduced in Chapter 5, however, the performance of SAID on natural images is poor, it needs improvement.

There remain many open questions about pursuing this approach. Human gaze does not fixate just on areas of uncertainty, but also on areas of interest for survival purposes. We have used the ability to reconstruct a scene accurately as a proxy for regions of interest. Clearly in the highly simplified datasets that we have explored this is sufficient to solve many downstream tasks. We see this work as an opening exploration of using minimum description length with an information bottleneck for representation learning and image understanding, it starts a new direction on unsupervised representation learning for multi-object images, but a huge number of important issues remain to be explored.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The advantage of deep generative models is that they can learn a meaningful latent space while aiming at generation. Although realistic generations are attractive, it is more useful when we can obtain and control the latent space learnt from data. We first introduce LSR-GAN, a tool that can visualize the latent space of VAE by imagining sharper images. This VAE and GAN architecture can be replaced by other VAEs and GANs extensions, it will not damage the performance of GAN while we can keep the latent space learnt by VAE. Also, it can solve the problem of mode drop and stabilize the training of GAN. It can be regarded as a new hybrid of VAE and GAN but the original idea is to visualize the latent space of VAE. We should be careful when leveraging the β and λ value during training, the GAN still has a chance to mode collapse even when we add a regulariser. Then we explore VAE in the transfer learning field and find VAE is capable of using pre-trained models. Transfer learning helps convergence in the beginning, but this cannot improve the image quality of LSR-GAN because the quality is based on adversarial training but not ELBO.

We also investigate disentangled representation learning. Unlike previous work on disentangling attributes of images, our model tries to disentangle the foreground and background information. It is capable to generate brand new and meaningful images by swapping the latent variables of different images. And for non-object representations, VAE-B can extract information from the original image to fill the area that belongs to objects with a similar colour. The performance varies on different datasets. Moreover, our model can also learn disentanglement between other attributes of images, such as style and content by substituting foreground images with images that represent the labels.

However, the background and foreground information can still be subdivided into detailed concepts. Thus, we dive into this direction and propose a model to learn an object-centric model which aims at learning a representation that disentangles the foreground information. The model is based on the observation that the foreground area is used to contain more information which results in an error map with different errors in the background and foreground are after compression. However, this might not be suitable when the foreground object is highly close to the background. By combining the VAE with a small attention mechanism, it can find the areas around objects. This is a very light network requiring iterative training. But the drawback is also clear. The capacity of the bottleneck is hard to be tuned and the bounding boxes given by the k iterations from the model are not all accurate enough while those boxes are always larger than small objects.

Overall, based on the strong ability of the VAE on learning representation, we aim to learn a factorised representation from the shallow level to the deep level, since we believe that a highly factorised representation is more efficient and usable. As shown before, the original representation is not efficient to be used on generation or downstream tasks. Thus, the overflow of the whole work is to first factorise the image into foreground and background, while the definition of the foreground is subjective. The results show the ability to generate meaningful images with such a factorised representation, also, it brings more possibilities to applications such as conditional generation and background changing. To a deeper level of representation, we manage to segment the representation of a multi-object foreground into single-object representation and show the potential of the representation to be used on downstream tasks. The results of experiments show the usability of learning factorised representation. Currently, the quality of the deep level of representation is limited under both supervised and unsupervised manners, this work is only a small step on the way to learning a fully-factorised representation. We believe that when an image can be controlled by the representation according to what humans want, that would be a sign of a fully-factorised representation.

7.2 Future Work

There are different directions for future work:

- In our work, the object-centric representation can only appearance representation from the object representation. However, there are still other types of representations can be subdivided though a object and then help the desired downstream tasks, such as the category of the object. By factorising the category information of each object in a multi-object scene, it enables us to train a classification model that can classify different categories in one image.

- Generating images with manipulation is still a direction of interest. Currently, there are works that can achieve manipulated generation to a degree, such as our BFVAE or StyleGAN (Karras et al., 2019). However, when it comes to object-level manipulation, the dataset is highly restricted. As shown in StyleGAN, images in the dataset should be in one category for a better performance. Imagine a dataset contains only car images, then there is no need to learn a representation to show whether the car has a tail. However, when we add cat images into the dataset, then the model will need to learn the additional representation about tails. The complexity increases with the number of categories. A model that can only handle one category is always limited. By solving this issue, the model is expected to generate a black cat with seeing a black car and a white cat.

- The aggregation of common attributes for the background and foreground is also an issue that needs to be solved, like colour, texture or objects. Although what are foreground and background are highly task-depended. Many works disentangle the representation of objects successfully, however, the attribute in the background is ignored. As a human, we can draw a blue cat under the green sky by seeing a green cat under the blue sky, even when the green sky is unreal. This is hard for machines to draw. By achieving this, the representation of the background also needs to be disentangled and the factor should be shareable with the foreground.

- These types of works on images can be extended into videos. A snippet of video can be seen as a collection of sequential still images while it usually has more than one image per unit of time of the video. Thus, our model that works on 2D images needs to do spatial-temporal reasoning when we apply it to videos. This can be potentially solved by combining with the LSTM network (Hochreiter and Schmidhuber, 1997).

References

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168. PMLR, 2018.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433, 2015.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. *arXiv preprint arXiv:1710.11252*, 2017.

Marouan Belhaj, Pavlos Protopapas, and Weiwei Pan. Deep variational transfer: Transfer learning through semi-supervised deep generative models. *arXiv preprint arXiv:1812.03123*, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *Proceedings of International Conference on Learning Representations*, 2017.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. In *Proceedings of International Conference on Learning Representations*, 2017.

Tian Qi Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 2610–2620, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016a.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016b.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

Brian Cheung, Jesse A Livezey, Arjun K Bansal, and Bruno A Olshausen. Discovering hidden factors of variation in deep networks. *arXiv preprint arXiv:1412.6583*, 2014.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8789–8797, 2018.

Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3412–3420, 2019.

Bin Dai and David Wipf. Diagnosing and enhancing VAE models. *Proceedings of International Conference on Learning Representations*, 2019.

Fei Deng, Zhuo Zhi, Donghun Lee, and Sungjin Ahn. Generative scene graph networks. In *International Conference on Learning Representations*, 2021.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, 2018.

Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *arXiv preprint arXiv:1907.02544*, 2019.

Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. Genesis-v2: Inferring unordered object representations without iterative refinement. *arXiv preprint arXiv:2104.09958*, 2021.

Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. In *International Conference on Learning Representations.*, 2020.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, 2016.

Babak Esmaeili, Hao Wu, Sarthak Jain, Alican Bozkurt, Narayanaswamy Siddharth, Brooks Paige, Dana H Brooks, Jennifer Dy, and Jan-Willem Meent. Structured disentangled representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2525–2534. PMLR, 2019.

Patrick Esser, Ekaterina Sutter, and Björn Ommer. A variational U-Net for conditional appearance and shape generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8857–8866, 2018.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29:64–72, 2016.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

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.

Anand Gopalakrishnan, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Unsupervised object keypoint learning using local spatial predictability. In *International Conference on Learning Representations*, 2021.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR, 2019.

Klaus Greff, Sjoerd Van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, 2017.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein GANs. *arXiv preprint arXiv:1704.00028*, 2017.

Ishaan Gulrajani, Kundan Kumar, Faruk Ahmed, Adrien Ali Taiga, Francesco Visin, David Vazquez, and Aaron Courville. Pixelvae: A latent variable model for natural images. *arXiv preprint arXiv:1611.05013*, 2016.

Ananya Harsh Jha, Saket Anand, Maneesh Singh, and VSR Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 805–820, 2018.

Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–467, 2018.

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.

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.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *arXiv preprint arXiv:1812.02230*, 2018.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of International Conference on Learning Representations*, 2017a.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org, 2017b.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference*, 2016.

Huaibo Huang, Ran He, Zhenan Sun, Tieniu Tan, et al. IntroVAE: Introspective variational autoencoders for photographic image synthesis. In *Advances in Neural Information Processing Systems*, pages 52–63, 2018.

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.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information processing systems*, pages 2017–2025, 2015.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two pure transformers can make one strong gan, and that can scale up. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910, 2017.

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.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Proceedings of International Conference on Learning Representations.*, 2013.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Brian Kulis, Kate Saenko, and Trevor Darrell. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1785–1792. IEEE, 2011.

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.

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

Qi Li, Long Mai, and Anh Nguyen. Improving sample diversity of a pre-trained, class-conditional GAN by changing its class embeddings. *arXiv preprint arXiv:1910.04760*, 2019.

Yuheng Li, Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Mixnmatch: Multifactor disentanglement and encoding for conditional image generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727, 2015.

Chen-Hsuan Lin, Ersin Yumer, Oliver Wang, Eli Shechtman, and Simon Lucey. ST-GAN: Spatial transformer generative adversarial networks for image compositing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9455–9464, 2018.

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

Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *Proceedings of International Conference on Learning Representations.*, 2020.

Bingchen Liu, Yizhe Zhu, Zuohui Fu, Gerard De Melo, and Ahmed Elgammal. Oogan: Disentangling gan with one-hot sampling and orthogonal regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4836–4843, 2020.

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.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems*, 2020.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, 2015.

Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. Auxiliary deep generative models. In *International conference on machine learning*, pages 1445–1453. PMLR, 2016.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. In *Proceedings of International Conference on Learning Representations*, 2016.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems*, pages 5040–5048, 2016.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset, 2017.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

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

Takeru Miyato and Masanori Koyama. cGANs with projection discriminator. *arXiv preprint arXiv:1802.05637*, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, 2010.

Charlie Nash, SM Ali Eslami, Chris Burgess, Irina Higgins, Daniel Zoran, Theophane Weber, and Peter Battaglia. The multi-entity variational autoencoder. In *NeurIPS Workshops*, 2017.

Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. *Advances in Neural Information Processing Systems*, 33, 2020.

Atsuhiro Noguchi and Tatsuya Harada. Image generation from small datasets via batch statistics adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2750–2758, 2019.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2642–2651. JMLR. org, 2017.

Lili Pan, Peijun Tang, Zhiyong Chen, and Zenglin Xu. Contrastive disentanglement in generative adversarial networks. *arXiv preprint arXiv:2103.03636*, 2021.

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

Rajesh Ranganath, Dustin Tran, and David Blei. Hierarchical variational models. In *International Conference on Machine Learning*, pages 324–333. PMLR, 2016.

Siamak Ravanbakhsh, Francois Lanusse, Rachel Mandelbaum, Jeff Schneider, and Barnabas Poczos. Enabling dark energy science with deep generative models of galaxy images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, pages 14866–14876, 2019.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *International Conference on Machine Learning*, pages 1060–1069. PMLR, 2016.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, 2014.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5), September 1978. ISSN 0005-1098.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226. PMLR, 2015.

Jose San Pedro and Stefan Siersdorfer. Ranking and classifying attractiveness of photos in folksonomies. In *Proceedings of the 18th International Conference on World wide web*, pages 771–780, 2009.

Lucas Seninge, Ioannis Anastopoulos, Hongxu Ding, and Joshua Stuart. Vega is an interpretable generative model for inferring biological network activity in single-cell transcriptomics. *Nature communications*, 12(1):1–9, 2021.

Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. *Advances in Neural Information Processing Systems*, 29:3738–3746, 2016.

Elizabeth S Spelke. Principles of object perception. *Cognitive science*, 14(1):29–56, 1990.

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.

Emma EM Stewart, Matteo Valsecchi, and Alexander C Schütz. A review of interactions between peripheral and foveal vision. *Journal of vision*, 20(12):2–2, 2020.

Attila Szabó, Qiyang Hu, Tiziano Portenier, Matthias Zwicker, and Paolo Favaro. Challenges in disentangling independent factors of variation. *arXiv preprint arXiv:1711.02245*, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5066–5073, 2019.

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

Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. *arXiv preprint arXiv:2007.03898*, 2020.

Aaron Van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

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

Sjoerd van Steenkiste, Karol Kurach, Jürgen Schmidhuber, and Sylvain Gelly. Investigating object compositionality in generative adversarial networks. *Neural Networks*, 130:309–325, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring GANs: generating images from limited data. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 218–234, 2018.

Satosi Watanabe. Information theoretical analysis of multivariate correlation. *IBM Journal of research and development*, 4(1):66–82, 1960.

Nicholas Watters, Loic Matthey, Christopher P Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019.

Gregory P Way, Michael Zietz, Vincent Rubinetti, Daniel S Himmelstein, and Casey S Greene. Compressing gene expression data using multiple latent space dimensionalities learns complementary biological representations. *Genome biology*, 21:1–27, 2020.

Taihong Xiao, Jiapeng Hong, and Jinwen Ma. DNA-GAN: Learning disentangled representations from multi-attribute images. *arXiv preprint arXiv:1711.05415*, 2017.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Jianwei Yang, Anitha Kannan, Dhruv Batra, and Devi Parikh. LR-GAN: Layered recursive generative adversarial networks for image generation. *arXiv preprint arXiv:1703.01560*, 2017a.

Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9877–9886, 2019.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. Improved variational autoencoders for text modeling using dilated convolutions. In *International Conference on Machine Learning*, pages 3881–3890. PMLR, 2017b.

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363. PMLR, 2019.

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*, pages 5907–5915, 2017.

Miaoyun Zhao, Yulai Cong, and Lawrence Carin. On leveraging pretrained GANs for limited-data generation, 2020.

Shengjia Zhao, Hongyu Ren, Arianna Yuan, Jiaming Song, Noah Goodman, and Stefano Ermon. Bias and generalization in deep generative models: An empirical study. In *Advances in Neural Information Processing Systems*, pages 10792–10801, 2018.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.

Joey Tianyi Zhou, Sinno Jialin Pan, Ivor W Tsang, and Yan Yan. Hybrid heterogeneous transfer learning through deep learning. In *Twenty-eighth AAAI conference on artificial intelligence*, 2014.

Weijin Zhu, Yao Shen, Linfeng Yu, and Lizeth Patricia Aguirre Sanchez. Gmair: Unsupervised object detection based on spatial attention and gaussian mixture. *arXiv preprint arXiv:2106.01722*, 2021.

Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. Heterogeneous transfer learning for image classification. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.