Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Vector Quantised-Variational Autoencoders (VQ-VAEs) for Representation learning

Author:
Harry Coppock

Supervisor:
Prof. Björn Schuller

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial Intelligence of Imperial College London

September 2020

ABSTRACT

This report details an investigation into the applicability of Vector-Quantised Variation Autoencoders (VQ-VAEs) for representation learning. The VQ-VAE's representations are investigated and a novel regularisation technique involving noisy vector quantisation is introduced. It is then shown, through a range of quantitative and qualitative experiments, that this regularisation technique forces the VQ-VAE to learn superior representations. A theoretical explanation into why this is the case is also provided. In addition to this a new prior is fitted to the VQ-VAE latent space in the form of an adapted Progressive Growing Generative Adversarial Network (PGAN) with self attention which yields a significant reduction in training and sampling time.

ACKNOWLEDGMENTS

I would like to thank my supervisor Prof. Björn Schuller for the invaluable feedback and support throughout the project. I would also like to thank my PhD supervisor Mr Georgios Rizos for our weekly zoom meetings and his constructive comments and endless supply of creative tips and tricks. I would finally like to thank my parents, sister, partner and friends for supporting me as always and helping take my mind off work when a break was needed.

Contents

Chapter 1

Introduction

Humans are a direct and beautiful source of inspiration for the field of Artificial Intelligence (AI). It has become clear that the biological brain relies on its structure and design as much as on its raw compute power. The brain efficiently accesses high level thought and reason through forming *representations* of abstract concepts (See Section 2.1.1) (Roy et al., 2018; Zucker, 1981). Using these high-level representations, humans form internal models of the world, known as *mental models*. It is in this constructed space that we experience the world (Ha & Schmidhuber, 2018). Models that are constructed by abstract representations result in smaller problem state spaces and present patterns, concepts and relations in a disentangled and simple to interpret manner (See section 2.1.1). As these models contain representations of the dynamics and laws of the world, they serve as a simulation that enables us to efficiently plan, to evaluate policies and to take appropriate preemptive actions for events which are yet to occur.

The recent progress in classical deep learning tasks such as classification and language modelling demonstrates[1] that AI models already extract high level representations of the world Bengio et al. (2012). However, these models still fall significantly short of the adaptability and robustness of biological computation. One can postulate that the explanation for this difference in adaptability is due to the nature of the representations formed. AI representations are implicit, hidden in the entanglement of the network. They are task-specific and the process of their construction looses a lot of the information about the data, over-fitting to the task at hand. This is in stark contrast to the explicit and general representations that the biological brain forms. The field of representation learning identifies these issues and endeavours to formulate better methods to address them. The family of techniques known as autoencoders has achieved considerable progress in this field. Autoencoders form explicit representations of the data in the latent space, and maximise the mutual information of the data and the representations (Vahdat & Kautz, 2020; Kingma & Welling, 2013). The latent space created by autoencoders represents all modalities and dependencies of the data, not just the ones relevant to a specific task.

This MSc project identifies the quality of explicit representations as a limiting factor in creating artificial mental models. It therefore aims to make progress towards forming better representations of the world using state of the art Vector Quantised Variational Autoencoders (VQ-VAEs) (van den Oord et al., 2017). This MSc project makes the following contributions:

1. We propose a new regularisation technique which injects uncertainty into the vector quantisation step in VQ-VAEs. We show that while it has an insignificant detrimental effect on the quality of reconstructed data instances, it dramatically improves the standard of the representations extracted from the data. This is demonstrated through a range of quantitative and qualitative experiments.

[1] in order to solve these complex tasks high level representations are required.

2. The mechanisms which result in the disentanglement and identification of data representations and generative factors for VQ-VAEs and Variational Autoencoders (VAEs) are directly compared. Their differences and similarities are highlighted and discussed. Inspiration and theoretical justification for improving VQ-VAEs is taken from this analysis.
3. A novel Progressive Growing Generative Adversarial Network (PGAN) with self attention is fitted to the VQ-VAE latent space to act as a powerful generative prior. To the best of our knowledge this is the first time the original autoregressive prior has been replaced. While yielding sub-optimal generative results, it serves as a proof of concept and significantly reduces training time and computation requirements.
4. A new subset of ImageNet is created, featuring only classes with a large instance count, keeping to the theme of animals. It serves as a bridge between the trivial and low resolution CIFAR and MNIST datasets and the much more complicated full ImageNet. We name it NaturalImageNet and it is stored in an open access repository so that is can be used for educational and research purposes at Imperial College London.
5. The source code for all experiments and models is released open source on GitHub and can be found here.

The ethics and legality of this report are considered and detailed in Section 6.2. The key factors which arose from this study were that datasets featuring sensitive information, in the form of human faces, were only legal to train on (and so classed as processing under the GDPR) as this work was motivated for purely research purposes and not for commercial gain. And that shockingly the approximated environmental impact of training the models, through indirect CO_2 released into the atmosphere, was more severe than an economy ticket return flight from London to New York (see Section 6.2.1).

Chapter 2

Background

2.1 Motivation

2.1.1 Evidence for representation learning in the biological brain

Works claiming to prove high levels of abstraction in the human brain are diverse in the stance that is taken. Some take a complexity point of view[1], that given the brain's known resources, of number of neurons, number of synapses and neuron signal transmission time, the observed phenomena of human reasoning and actions are not computationally viable unless high level representations are formed (Tsotsos, 2017; Roy et al., 2018). These ideas stem from Zucker (1981) detailing that *"One of the strongest arguments for having explicit abstract representations is the fact that they provide explanatory terms for otherwise difficult (if not impossible) notions"*. This alludes to the widely accepted concept that the brain constructs a hierarchy of concepts where higher level concepts are described and constructed through a combination of lower level concepts (George, 2008; Hawkins & George, 2006; Tsotsos, 2017)

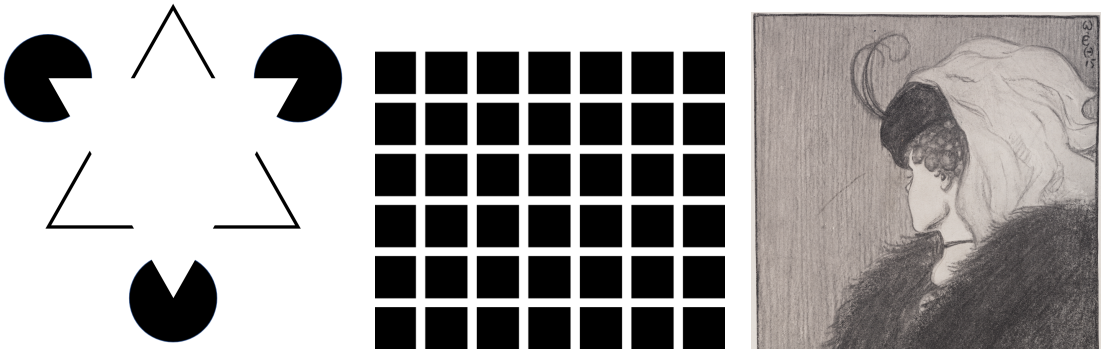
Others detail neurological findings, for example, that individual neurons in the medial temporal lobe (MTL) are only activated by very specific concepts, independent of the form in which this concept is presented to the patient. An example of this phenomenon is evidence of a single neuron only firing for a range of pictures that feature former American president Bill Clinton, but not for other images (Roy, 2017). These neurons are named by Roy (2017) as abstract modality invariant cells and lead to individual neurons being labeled by the high level concept that activates them, for example, the "Jennifer Aniston cell"(Quiroga et al., 2005). This effect was even seen to extend to when the patient observed words detailing the same concepts as the picture,[2] firing the same individual neuron in the MTL (Quiroga et al., 2005). Similar studies on monkeys have demonstrated an invariance to viewing angle, distance and size of the object, in neuron activity in the higher tier ventricular pathway, for specific objects such as faces or particular food objects (Logothetis & Pauls, 1995; D I Perrett, 1982). These findings detail a biological transform from sensory input signals to a high level concept that allows discrete neurons to "encode abstract representations"[3] of aspects of the world, in other words evidence for representation learning in the biological brain.

2.1.2 Mental Models in the Biological Brain

Humans reason, make decisions and experience the world through highly abstracted representations of our environment. When we plan actions or recall past events we do not process the world in all its spatial and temporal complexity, rather we focus on refined abstractions which embody the key concepts of the matter in question. What's more, we can play around with these representations

[1] Termed *complexity level analysis* in Tsotsos (1988)
[2] explicitly for this example, the string "Bill Clinton"
[3] (Quiroga et al., 2005)

to test our understanding and plan policies. We Imagine. Imagination is at the core of intelligence and requires a (mental) model of the environment which we can experience and with which we can interact. (Hassabis D, 2007; Hassabis et al., 2007; Neil Burgess, 2001; Dijkstra et al., 2017; Dentico et al., 2014).



((a)) Kanisza triangle **((b))** Hermann's grid **((c))** My Wife and My Mother-in-law

Figure 2.1: Figure detailing a sample of famous illusions, demonstrating our mind's ability to assert prior knowledge on observations to alter what is perceived, i.e. performing inference. **a)** Kanisza's triangle, a second white triange is seen overlapping a wire framed triangle, this is not the case, with the mind wrongly inferring the edges of the top triangle (Bradley & Petry, 1977). **b)** Hermann's grid where the mind perceives black objects at the cross hairs of the the grid despite nothing being there (Brewster, 1844). **c)** My Wife and My Mother-in-law illusion where the brain switches between percieving an old woman, the mother-in-law and a young woman, the wife (anonymous German postcard 1888)

The existence of the phenomenon of imagination (and dreams) is a demonstration of the presence of mental models. Mental models are also required for a range of other observed phenomena. One example is that what we perceive is determined by both the true input, i.e the sensory input, and by what our brain predicts (Nortmann et al., 2013; Maus et al., 2013): our brain performs inference so that what we experience is inferred from what is observed given our internal model of the world. This is shown best through illusions. Three example illusions that demonstrate this effect are given in Figure 2.1. Illusions demonstrate what is sometimes termed "unconscious inference"[4] where what is experienced is a combination of what is observed and what is subconsciously expected. Most of the time, given imperfect information, this effect is greatly beneficial to our understanding. It allows us to generalise well and helps us to infer which are the important factors and to predict likely consequences of events.

2.1.3 World Models

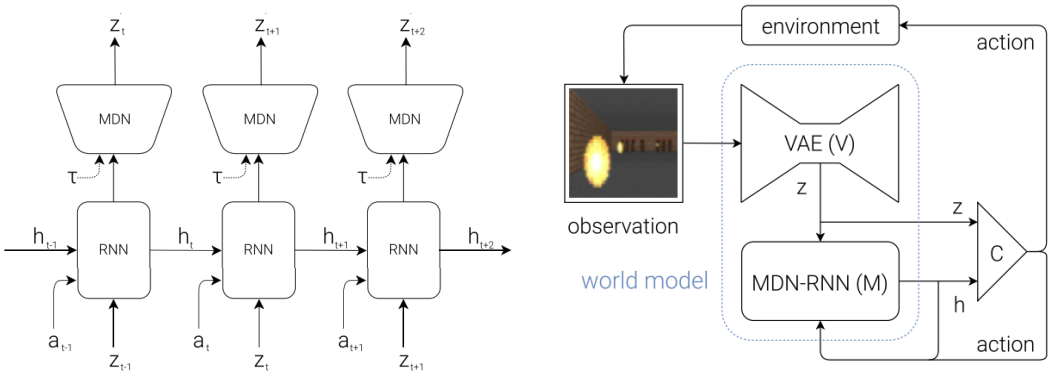
World models is the field of research which attempts to emulate, for artificial agents, the human brain's ability to construct a *mental model* of the world. Once a model of the world has been constructed, the agent can then train, test policies and plan in this compressed representation of the world. Two pieces of work that have made great progress in utilising mental models, namely Ha & Schmidhuber (2018) and more recently Hafner et al. (2019) construct spatial and temporal representations of the task and can then train the agent fully in its own "dream" world.[5] Both studies achieved (at the time) state of the art performances in a range of OpenAI gym (Brockman et al., 2016) bench mark challenges as a result of forming these models.

Ha & Schmidhuber (2018)'s approach (which was somewhat replicated in (Hafner et al., 2019)) consisted of a VAE which learnt a compressed representation, $\mathbf{z}$, of the environment at a given

[4] (Notredame et al., 2014)

[5] Once a compressed representation of the spatial and temporal dynamics of the task has been formed the agents can be trained solely in this "imagined" world

time step and a Long Short Term Memory (LSTM) Recurrent Neural Network (RNN) model, M, which learnt a temporal representation of the environment. This was achieved by predicting the next latent vector produced by the environment. The M model predicted the next latent state, $\mathbf{z}_{t+1}$ given the current action, a_t, hidden state, $\mathbf{h}_t$, and latent state, $\mathbf{z}_t$ and therefore attempted to model the dynamics of the world. The model took into account the stochasticity of the environment by sampling from $P(\mathbf{z}_{t+1}|a_t, \mathbf{h}_t, \mathbf{z}_t)$ with a Mixture Density Network and the level of uncertainty quantified by a temperature parameter, τ (similar to work done in Ha & Eck (2017)). Figure 2.2a) details a flow chart representing model M. In figure 2.2b) the flow diagram details the controller, C, using a combined representation of the true observed environment and M's future predictions, but it is important to note that the C model could also be trained purely in the M model's predictions, i.e. using no observation from the environment. This is what the authors refer to as dreaming.



((a)) M model **((b))** Agent model

Figure 2.2: a) Flow diagram of the M model in Ha & Schmidhuber (2018). Where $\mathbf{z}$ is the true and predicted latent representation of the environment and $\mathbf{h}$ is the hidden state of the world model. MDN is a gaussian mixture model which given an output from the RNN and τ, a temperature parameter (measure of uncertainty) samples a prediction for the next latent vector. It is trained to predict the next latent vector, $\mathbf{z}_{t+1}$, which is sampled from the environment given the current latent vector, $\mathbf{z}_t$, the agents action, a_t, and the hidden state, $\mathbf{h}_t$. In order to successfully do this it must model the dynamics of the world. Once the model has been trained, it provides a stand alone simulation which the agent can imagine/dream in and learn a good policy. **b)** Flow diagram of the agent model in Ha & Schmidhuber (2018). Where $\mathbf{z}$ is the latent compressed representation of the environment and $\mathbf{h}$ is the hidden state of the world model. C is the controller which decides on the action to take and is trained to act in a manner which results the maximum expected future cumulative rewards (Mnih et al., 2015; Sutton & Barto, 1998)

.

Both (Ha & Schmidhuber, 2018; Hafner et al., 2019) use Variational Autoencoders, VAEs (please see section 2.3) to form the compressed representations that make up the world model. As detailed in Section 2.3, this limits the resolution of the environment that the agent can be trained on, as currently VAEs are limited to low resolution images,[6] preventing the application of these models to more complex datasets. In addition to this the compressed representation is formed through minimising the means squared error (MSE) between the input and reconstructed images. This results in the world model, representing a lot of redundant information. Ha & Schmidhuber (2018) noted that this resulted in details such as the texture of the walls in the game environments being unnecessarily modelled. An example of where the MSE limitations are problematic is shown in Figure 2.3.

[6]However, very recent work by Vahdat & Kautz (2020) at Nvidia would excitingly suggest otherwise, please see section 2.3.4

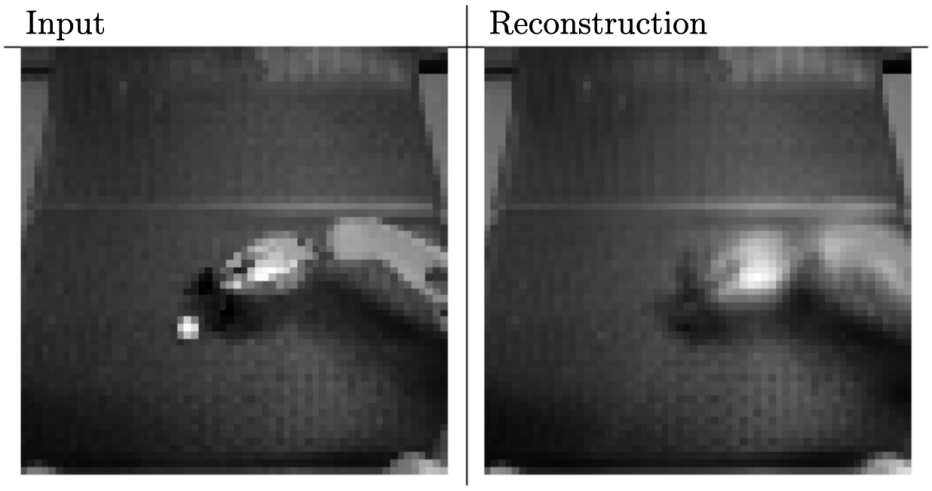


Figure 2.3: Figure detailing the input and reconstructed images from an autoencoder (Please see section 2.2.2) trained to minimise MSE. Here the autoencoder is tasked with forming a compressed representation of a scene where a robot arm is attempting to pic up a ping pong ball. The autoencoder has failed to form a compressed representation of the ping pong ball, arguably the most important aspect to the scene. This demonstrates how mean squared error (MSE) has it's limitations in forming compressed representations of the world. Image taken from (Goodfellow et al., 2016) which was kindly provided to the authors by Chelsea Finn.

2.2 AUTOENCODERS

2.2.1 THE MANIFOLD HYPOTHESIS

The manifold hypothesis postulates that data in the real world, that exist in high dimensional spaces, concentrate in a much lower dimensional space, known as a manifold, which is embedded/entangled within the high dimensional space (Bengio et al., 2012). This manifold corresponds to volumes of space which have a high data probabilistic density, because regions in the manifold are very likely to correspond to meaningful data points whereas points off the manifold are not. This concept can be illustrated, for example, by imagining a high dimensional space, S which represents all possible 30 second sound snippets where each dimension represents a particular frequency of sound that is produced at a given temporal slot. To demonstrate the presence of manifolds, we need to prove that the volume of space corresponding to meaningful sounds, for example music, M, where $M \subseteq S$, occupies a very small portion of S. To do this we can sample from the space S by creating a 30 second tune where each temporal slot is made up of a random note. The vast majority of these snippets will sound like white noise, i.e. nothing like music. As the chances of randomly sampling a 30 second sound snippet that sounds anything remotely like music from M is extremely small, it proves that M must occupy an extremely small proportion of the space of S (Bengio, 2019).

2.2.2 AUTOENCODERS

Extracting meaningful representations from data is a complex and highly non linear task with no current explicit objective function. Advances have been made however, through the family of techniques known as autoencoders. Autoencoders perform well at this task and are faced with the problem of manipulating a flow of information through a compressed space, known as a "bottleneck", with the objective of matching the output as closely as possible to the input (Kramer, 1991). In the bottle neck, the autoencoder is forced (limited by its capacity) to organise/present the information in a manner which summaries it well so that it can be accurately mapped back to its full complexity (Tishby et al., 2000). For example, in the context of human faces, if tasked with summarising someone's face, a poor strategy would be to list the first few pixel values of the image as the information for the majority of the face would be lost and therefore could not be reconstructed.

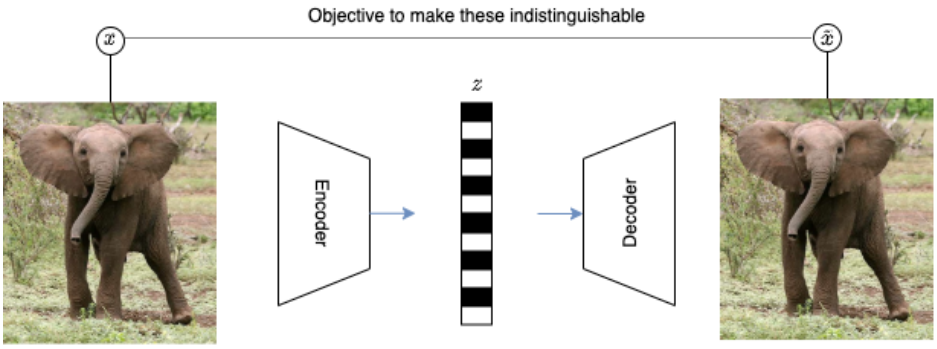


Figure 2.4: Schematic detailing a high level overview of an autoencoder. Here an image, which exists in a very high dimensional space, is compressed to a vector, $\mathbf{z}$, by an encoder which learns a highly non linear projection from the image space onto the latent space. The decoder then learns a mapping from this compressed representation back to the image space. Both the encoder and the decoder are trained to minimise the reconstruction loss, or the difference between the input and reconstructed image.

A better strategy would be to list the key attributes of the person for example, gender, age, ethnicity, emotion and pose. It is these complex concepts or representations that, under the right conditions, the model is forced to learn.

Autoencoders consist of an encoder $E()$ and a decoder $D()$, parameterised by deep neural networks, $E_{\theta_1}(), D_{\theta_2}()$. The encoder learns to project the high dimensional input, $\boldsymbol{x}_i$, onto the low dimensional latent space. The encoder performs the task of extracting the representations or features of the data (Bengio et al., 2012). These representations, $\boldsymbol{z}_i$ exist in the latent space and as a result are often referred to as latent vectors. The decoder then learns to map the latent representations back to the input space. Figure 2.4 details the high level overview of the autoencoder.

$$\boldsymbol{z}_i = E_{\theta_1}(\boldsymbol{x}_i) \tag{2.1}$$

$$\tilde{\boldsymbol{x}}_i = D_{\theta_2}(\boldsymbol{z}_i) \tag{2.2}$$

Where $\tilde{\mathbf{x}}_i$ is the reconstructed data. Both encoder and decoder are trained to minimise the reconstruction loss, which is simply the difference between the input and reconstructed data. In the case of images, this can be the mean squared error (MSE) in pixel values.

$$\min_{\theta_1,\theta_2} \sum_{i}^{N} \Big(\boldsymbol{x}_i - D_{\theta_2}(E_{\theta_1}(\boldsymbol{x}_i))\Big)^2 \tag{2.3}$$

Where θ_1, θ_2 represent the parameters of the encoder and decoder respectively and N details the number of samples in one minibatch.

The reconstruction loss results in the autoencoder being optimised to act as an approximate identity function to the data (Bengio et al., 2012). This results in the model easily *overfitting* to the data. This is an issue with regards to forming meaningful representations. Autoencoders have no explicit objective to form meaningful features. Left unconstrained they can learn to map each input feature to its own unique latent, but although this enables good reconstruction, it forms a latent space where locations refer to data instances rather than combinations of concepts (Vincent et al., 2008). This latent space will lack all the key attributes of a meaningful representation of the world, in that the data manifold (see section 2.2.1) will be highly entangled in the latent space, with the majority of the latent space failing to correspond to meaningful data points (or images), once decoded. This results in vanilla autoencoders being a poor choice for forming world models.

2.2.3 Denoising Autoencoders

To combat the limited applicability of autoencoders to representation learning a new approach to the training objective was taken, based on the idea that representations should be largely invariant

to noise and corruption.[7] Denoising autoencoders (Vincent et al., 2008) re-frame the standard autoencoder objective function from the identity function, detailed in Section 2.2.2, to an objective of decorrupting the input data, with the challenge of first compressing the data through a bottle neck. With this reformulation of the objective function, the model cannot simply learn the trivial identity function (Vincent et al., 2008).

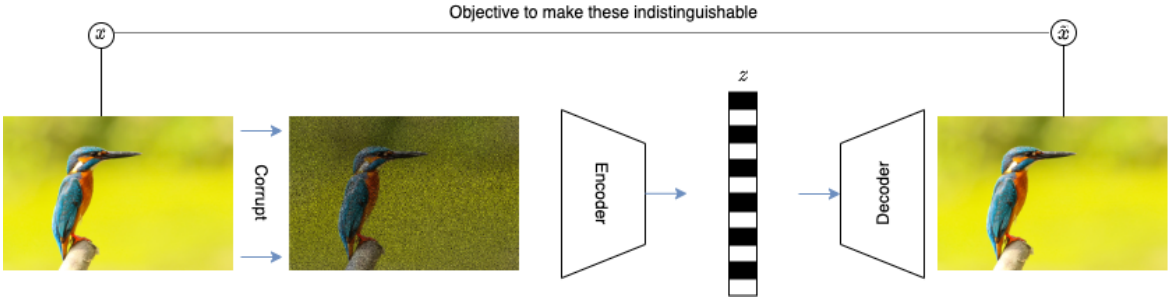


Figure 2.5: Schematic detailing a high-level overview of an denoising autoencoder. It takes the same structure as an autoencoder (Figure 2.4), however the input data is first corrupted before it is compressed. The objective to to minimise the reconstruction loss between the reconstructed data and the data before corruption.

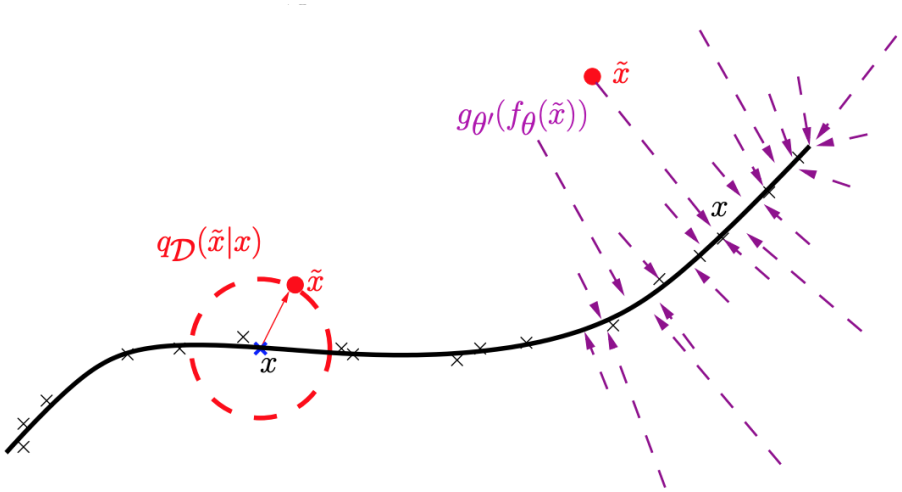


Figure 2.6: Schematic illustrating the manifold learning perspective of denoising autoencoders. Where the black line represents the data manifold in high dimensional data space. $\times$'s represent data instances, $\mathbf{x}$, which lie on the data manifold, and $\bullet$s represent corrupted images $\tilde{\mathbf{x}}$ which have been corrupted by some process $q_D(\tilde{\mathbf{x}}|\mathbf{x})$. Dotted arrows represent the model's learned mapping, $p(\mathbf{x}|\tilde{\mathbf{x}})$, where g and f represent the decoder and encoder respectively. Image taken from (Vincent et al., 2008)

Vincent et al. (2008) also explained the approach through a manifold learning perspective. Based on the idea that corrupted images will lie in an offset position relative to the manifold in data space and that the autoencoder is now tasked with learning a decorruption mapping from an area of low probability or off the manifold to an area of high probability or on the manifold. This perspective is detailed schematically in Figure 2.6. As the autoencoder first projects the data onto a compressed representation, this compressed representation can be viewed as a "coordinate system"[8] for the

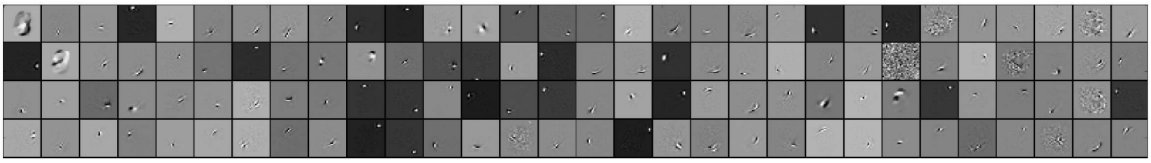
[7] i.e Mozart is still Mozart even with poor radio signal and a cat is a still a cat even if half of it's body is obscured.

[8] (Vincent et al., 2008)

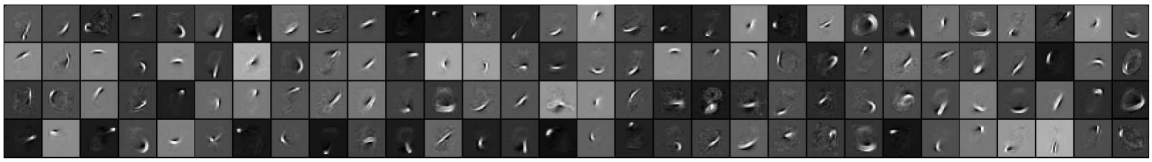
manifold, and is therefore a better representation of the patterns and variations of the data.

2.2.4 SPARSE AUTOENCODERS

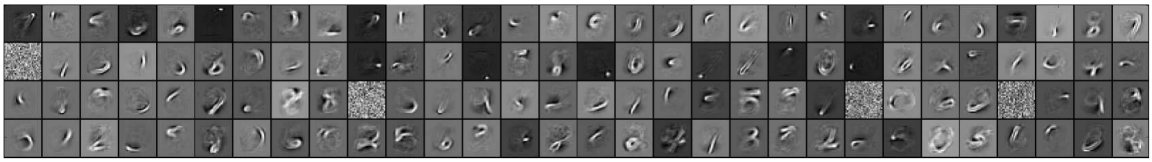
Sparse Autoencoders (Hinton et al., 2012; Lee et al., 2008; Makhzani & Frey, 2013) are built on the same principles as the well known regularisation technique, dropout. The autoencoder is sparse in that a constraint is placed on the autoencoder, limiting the amount of active neurons at a given time. This yields better representations as it prevents complex "co-adaption"[9] where models rely on all of the features of an input, resulting in over fitting and a latent space which represents data instances, not concepts. A successful variation on the sparsity constraint which yielded state of the art feature extraction for downstream tasks at the time, was k-sparse autoencoders (Makhzani & Frey, 2013). Here the k highest neuron activations in the hidden layer are selected, while the other activations are set to zero. This allowed for the explicit control of the sparsity of the model and demonstrated a direct correlation between the level of sparsity and the quality of the extracted representations. This can be seen in the Figure 2.7 where the learned features from the hidden layer are visualised. Low levels of sparsity (high k values) yielded very local feature extraction, being of little use for simple models in downstream tasks, however, increasing the value of k and making the model more sparse resulted in more global features being extracted for each high level neuron (Makhzani & Frey, 2013). It is important to note that limiting the bottleneck to the k highest activations results in a higher degree of compression and so favours more global feature extraction.



(a) $k = 70$



(b) $k = 40$



(c) $k = 25$

(d) $k = 10$

Figure 2.7: Learned filters for varying levels of k in a k-sparse autoencoder training on the MNIST dataset. Here we can see that the tighter the constriction on the bottle neck layer (smaller k value) the higher the level of representation is extracted (Makhzani & Frey, 2013).

[9] (Hinton et al., 2012)

2.3 VARIATIONAL AUTOENCODERS

Despite it's name and the final formulation containing an "encoder" and a "decoder", Variational Autoencoders (VAEs)(Kingma & Welling, 2013) have a completely different mathematical foundation to that of autoencoders (See Section 2.2.2)(Kingma & Welling, 2013; Doersch, 2016; Weng, 2018). VAEs were formulated with the aim of generating new data instances, utilising methods in the field of variational bayes. Formally, if we have some data, $\boldsymbol{X}$, distributed by some unknown distribution $P_{real}(\boldsymbol{X})$ we would like to learn/train a model, $P_{model}(\boldsymbol{X})$, where $P_{model}(\boldsymbol{X})$ and $P_{real}(\boldsymbol{X})$ are as similar as possible. Generative processes often first sample from a latent distribution, the reason for this is that the model then knows what type of data instance/structure to create, Doersch (2016) details that when training a generative model on a dataset with complex dependencies it helps to first decide the main concepts of the output before it is created. In that without this, the output could consist of a scrambling of concepts, not seen in combination in the real dataset.[10]. The objective is formulated as maximising the probability that the generative model will create every data instance, i.e. maximum likelihood[11]:

$$P(\mathbf{X}) = \int P_\theta(\mathbf{X} \mid \mathbf{z}) P(\mathbf{z}) d\mathbf{z} \tag{2.4}$$

VAE's solve Equation 2.4 through first assuming $P(\mathbf{z})$, the distribution dictating these "latent decisions"[12], to be a standard isotropic Gaussian, $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, in that all the data concepts and dependencies can be represented by a simple high dimensional Gaussian. And, rather than sample all possible $\mathbf{z}$ values to compute $P(\mathbf{X})$, which can be intractable in high dimensions, use a separate function $Q_\phi(\mathbf{z}|\mathbf{X})$, to sample only values of $\mathbf{z}$ that are likely to produce data instances. This sample space is likely to be considerably smaller than the total z space.

The formulation now somewhat resembles the structure of an autoencoder, in that the *encoder*, is $Q_\phi(\mathbf{z}|\mathbf{X})$ and the *decoder* is $P_\theta(\mathbf{X} \mid \mathbf{z})$. Figure 2.8 details the general structure of the vanilla VAE.

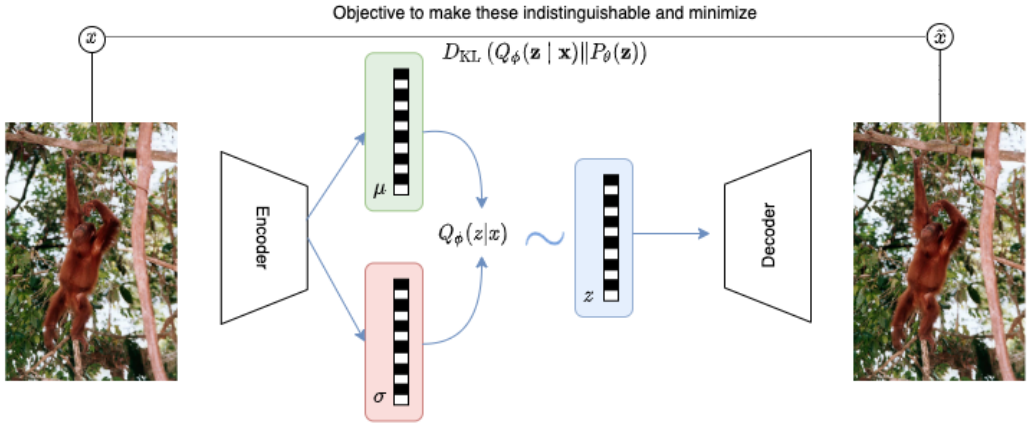


Figure 2.8: Schematic detailing the high-level structure and training principles of a VAE. The encoder takes an input and outputs a mean, μ and variance, σ. This defines the approximate posterior, $Q_\phi(\mathbf{z}|\mathbf{X})$ which is then sampled from using the reparamaterisation trick (see section 2.3.2). This sampled vector, $\mathbf{z}$ is then passed to the decoder which reconstructs the image. The VAE is trained to minimise the reconstruction loss and the divergence between the approximate posterior and sampling prior ($D_{KL}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P(\mathbf{z})\right)$) (Kingma & Welling, 2013)

[10] Take the digits 0-9, without first deciding which number to create, one may end up starting to write a 5 but then transition to writing a 7, yielding a generated output that doesn't resemble any data instance (Doersch, 2016)

[11] Following the logic that if the model can produce data instances it can also produce new, similar instances (Doersch, 2016)

[12] (Doersch, 2016)

2.3.1 VAE LOSS FUNCTION

The VAE loss function revolves around the relationship between $E_{z\sim Q}P(X \mid z)$ and $P(X)$ (Doersch, 2016). The final VAE loss function, which we want to minimise, is detailed below:

$$L_{\text{VAE}} = -E_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})} \log P_\theta(\mathbf{x} \mid \mathbf{z}) + D_{\text{KL}}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z})\right) \tag{2.5}$$

Here the first term represents the data likelihood, given latents sampled from the posterior, which we want to maximise and the second term is a measure of the difference between the approximate posterior and the sampling prior which we want to minimise. Equation 2.5 can be derived by starting with wanting to minimise the difference between the estimated posterior and the true posterior (Doersch, 2016; Weng, 2018). This can be achieved through minimising the Kullback-Leibler divergence, $D_{KL}(\bullet|\bullet)$ between the two distributions.

Formally we want to minimize:

$$D_{\text{KL}}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z} \mid \mathbf{x})\right)$$

Where $P_\theta(\mathbf{z} \mid \mathbf{x})$ represents the true posterior. Expanding out the Kullback-Leilber divergence yields

$$= \int Q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{Q_\phi(\mathbf{z} \mid \mathbf{x})}{P_\theta(\mathbf{z} \mid \mathbf{x})} d\mathbf{z}$$

As $P(A \mid B) = P(A, B)/P(B)$ and expanding the logs:

$$= \int Q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{Q_\phi(\mathbf{z} \mid \mathbf{x}) P_\theta(\mathbf{x})}{P_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

$$= \int Q_\phi(\mathbf{z} \mid \mathbf{x}) \left(\log P_\theta(\mathbf{x}) + \log \frac{Q_\phi(\mathbf{z} \mid \mathbf{x})}{P_\theta(\mathbf{z}, \mathbf{x})}\right) d\mathbf{z}$$

As the probability of all $\mathbf{z}$ given all $\mathbf{x}$ is 1 ($\int Q(\mathbf{z} \mid \mathbf{x}) d\mathbf{z} = 1$)

$$= \log P_\theta(\mathbf{x}) + \int Q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{Q_\phi(\mathbf{z} \mid \mathbf{x})}{P_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

$$= \log P_\theta(\mathbf{x}) + \int Q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{Q_\phi(\mathbf{z} \mid \mathbf{x})}{P_\theta(\mathbf{x} \mid \mathbf{z}) P_\theta(\mathbf{z})} d\mathbf{z}$$

$$= \log P_\theta(\mathbf{x}) + E_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{Q_\phi(\mathbf{z} \mid \mathbf{x})}{P_\theta(\mathbf{z})} - \log P_\theta(\mathbf{x} \mid \mathbf{z})\right]$$

collapsing back into the Kullback-Leibler divergence

$$= \log P_\theta(\mathbf{x}) + D_{\text{KL}}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z})\right) - E_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})} \log P_\theta(\mathbf{x} \mid \mathbf{z})$$

Then rearranging yields:

$$\log P_\theta(\mathbf{x}) - D_{\text{KL}}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z} \mid \mathbf{x})\right) = E_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})} \log P_\theta(\mathbf{x} \mid \mathbf{z}) - D_{\text{KL}}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z})\right)$$

To create a good generative model the left hand side of the equation needs to be maximised. This correlates to maximising the data likelihood, and minimizing the difference between the true and approximated posterior. The advantage of this reformulation is that the left hand side (which could not otherwise be trained via standard gradient descent techniques) is now described in terms that can be optimised via standard gradient descent techniques (the right hand side of the equation)[13].

2.3.2 REPARAMATERISATION TRICK

The formulation of the VAE requires drawing samples, $\mathbf{z}$, from the posterior and passing them to the decoder. This is an issue when training via stochastic gradient descent as sampling from a distribution represents a non continuous stochastic function which cannot be differentiated, preventing the back-propagation of gradients. The reparameterisation trick allows this problem to be circumnavigated.

[13]This is why the VAEs loss function is often termed the evidence lower bound (ELBO) as L_{VAE} equals the likelihood of the data, $\log p_\theta(\mathbf{x})$, plus a Kullback-Leibler divergence, which is always positive (Kingma & Welling, 2013; Doersch, 2016; Weng, 2018)

As the posterior is constructed as a Gaussian distribution, defined by the encoder outputs (mean μ and variance σ), the sampling procedure:

$$\mathbf{z} \sim Q_\phi\left(\mathbf{z} \mid \mathbf{x}^{(i)}\right) = \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I}\right) \tag{2.6}$$

Can be reformulated as a deterministic function with a stochastic input ϵ which can be differentiated.

$$\mathbf{z} = \mu + \sigma * \epsilon \tag{2.7}$$

This allows for gradients to flow back through the μ and σ nodes and train the encoder (Kingma & Welling, 2013; Doersch, 2016).

2.3.3 β-VAE

β-VAE builds on VAEs by enforcing constraint on the latent space to a greater extent, with the aim of forming better disentangled representations from the data as a direct result. This is achieved through a slight reformulation to the objective function, so that the data likelihood is maximised but is subject to keeping the Kullbeck-Leibler divergence between the approximate posterior and the sampling prior below a constant, ϵ.

$$\begin{aligned} &\max_{\phi,\theta} E_{\mathbf{x}\sim D}\left[E_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})} \log P_\theta(\mathbf{x} \mid \mathbf{z})\right] \\ &\text{s.t. } D_{\mathrm{KL}}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z})\right) < \epsilon \end{aligned} \tag{2.8}$$

Under the Karush–Kuhn–Tucker (KKT) conditions (Kuhn & Tucker, 1951; Karush, 1939) Equation 2.8 can be formulated as a Lagrangian with Lagrangian multiplier β [14](Irina Higgins, 2017; Burgess et al., 2018). This yields a modification of the VAE loss function, Equation 2.9[15], with the addition of the β parameter. It allows for the control of the balance between prioritising reconstruction quality, data likelihood, and quality of/ level of encouragement to form disentangled representations. The effect is that increasing the value of β narrows the bottleneck, reducing the capacity of the latent space, thereby forcing these high level concepts to be disentangled and represented in an easy to interpret manner. Increasing the value of β has a drawback however, because constricting the bottleneck results in a loss of information and therefore a reduction in data quality.

$$\mathcal{F}(\theta, \phi, \beta; \mathbf{x}, \mathbf{z}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = E_{Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log P_\theta(\mathbf{x} \mid \mathbf{z})\right] - \beta D_{KL}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P(\mathbf{z})\right) \tag{2.9}$$

The $D_{KL}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P(\mathbf{z})\right)$ term can be viewed as a contractive information bottleneck. This can be seen through investigating the conditions required to minimise it. In order to reduce $D_{KL}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P(\mathbf{z})\right)$ to zero for one data instance, all dimensions in the the estimated posterior need to have $\mu_i = 0$ and $\sigma_i = 1$ so that $Q_\phi(\mathbf{z} \mid \mathbf{x}) = N(\mathbf{0} \mid I) = P(\boldsymbol{z})$. This results in all data instances being mapped to identical uniform Gaussians, therefore transmitting no information, as the bottleneck has been squeezed so tight that it has completely shut off. In the VAE this is prevented from occurring by the reconstruction loss term, which forces the model to balance between reconstruction quality (keeping the information bottle neck as wide open as possible) and the formation of meaningful representations (extent to which the bottleneck has been constricted)[16]. β-VAE explicitly controls this balance (Irina Higgins, 2017; Burgess et al., 2018).

VAEs have two additional properties that the latent space possesses: *continuity*, i.e two points close together in the latent space should yield similar reconstructed samples and *completeness*, i.e a point sampled from latent space should always give a meaningful output once reconstructed Doersch (2016). This is thanks to the $D_{KL}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P(\mathbf{z})\right)$ squeezing the latent representation into a small space. Burgess et al. (2018) explains that this forces the posterior distributions to overlap. As a consequence

[14]Due to the KKT complimentary slackness conditions both ϵ and β are larger than 0 therefore as we are maximising a lower bound the $\epsilon\beta$ term can be dropped.(Irina Higgins, 2017; Burgess et al., 2018)

[15]note that when $\beta = 1$, 2.9 becomes the standard VAE loss function

[16]The more similar the posterior is to the sampling prior the tighter the information bottleneck. The capacity of the bottleneck is large when there is a large dispersion of posterior means, μ_i with narrow variances σ_i, i.e. the bottleneck can be extremely descriptive of the data (but differs considerably from the prior), and the capacity is small when all the means are close to zero, with large variances (resembling the sampling prior)(Burgess et al., 2018).

of this, if a latent vector is sampled from this overlapping area the decoder could reconstruct a data modality from either of the two (or more) overlapping posteriors[17]. In order to maximise data likelihood, and minimise decoder confusion resulting from the overlap of multiple posterior distributions, the latent space should be constructed in a manner where similar samples in the data space also exist in similar spaces in the latent space, thereby minimising the effect of these overlaps on the reconstruction quality. Figure 2.9 details this effect schematically.

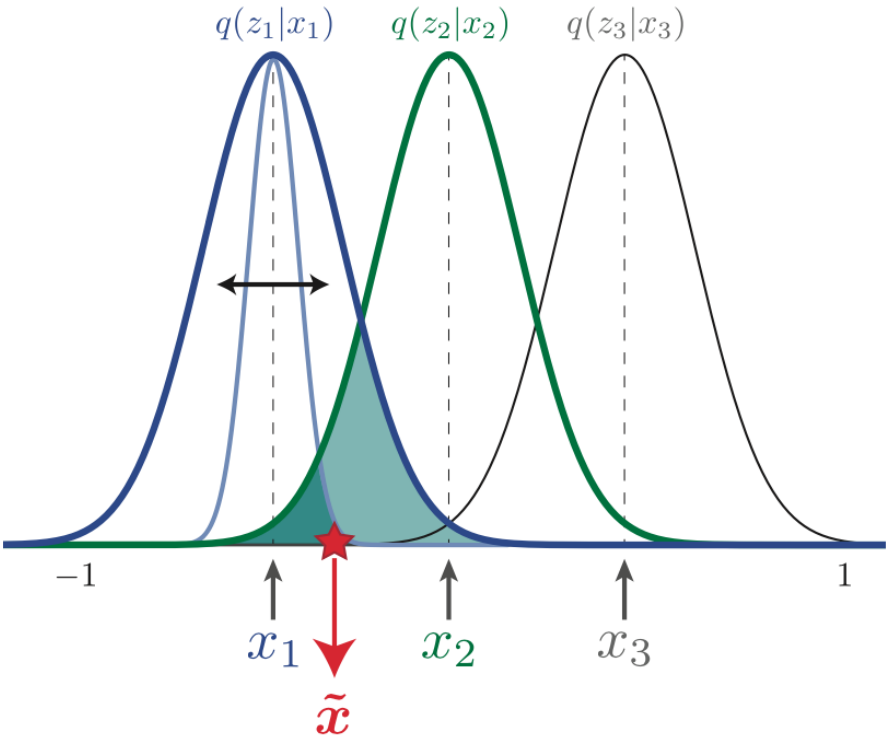


Figure 2.9: This schematic details the effect of overlapping posteriors on the construction of the latent space in VAEs. If a latent vector is sampled, $\tilde{x}$, such that it exists in an overlap region of two posteriors, $Q(z_1 \mid x_1)$ and $Q(z_2 \mid x_2)$, the decoder can be confused about whether the sampled latent should be decoded in a similar manner to a x_1 or x_2 data instance. Therefore, to maximise the data likelihood the encoder should construct the latent space in a manner which results in similar data instances being mapped to similar locations.(Burgess et al., 2018)

2.3.4 NOUVEAU VAE

Nouveau(N)VAE (Vahdat & Kautz, 2020), published in July this year, details the current state of the art generation capabilities for VAE objective based models. The work details a heavily engineered hierarchical VAE which yields great results through carefully designed architecture and training methods. It builds on works which feature a deep hierarchical VAE (Kingma et al., 2016), where the latent space is split into L groups $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_L\}$, with the aim of lower levels representing long range concepts (such as global structure) and higher levels representing short range concepts (such as texture). This structure hopes to enable the latent space to be more expressive of the data. The prior of each level is conditioned on all lower levels, therefore the prior of the model is given by $p(\boldsymbol{z}) = \prod_l p\left(\boldsymbol{z}_l \mid \boldsymbol{z}_{<l}\right)$ [18]. The approximate posterior, in addition to being conditioned on the data, x, is also conditioned on all lower levels, $Q(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_l Q\left(\boldsymbol{z}_l \mid \boldsymbol{z}_{<l}, \boldsymbol{x}\right)$. From this the hierarchical VAE

[17] The higher the value of β the more overlapp

[18] Lower levels in this formulation refer to a more constrictive latent space and so represent higher level/ long range concepts

objective function can be derived (Vahdat & Kautz, 2020; Kingma et al., 2016):

$$\mathcal{L}_{\mathrm{NVAE}}(\boldsymbol{x}) := E_{Q(\boldsymbol{z}|\boldsymbol{x})}[\log P(\boldsymbol{x} \mid \boldsymbol{z})] - \mathrm{D}_{\mathrm{KL}}\left(Q\left(\boldsymbol{z}_1 \mid \boldsymbol{x}\right) \| P\left(\boldsymbol{z}_1\right)\right) - \sum_{l=2}^{L} E_{Q(\boldsymbol{z}_{<l}|\boldsymbol{x})}\left[\mathrm{D}_{\mathrm{KL}}\left(Q\left(\boldsymbol{z}_l \mid \boldsymbol{x}, \boldsymbol{z}_{<l}\right) \| P\left(\boldsymbol{z}_l \mid \boldsymbol{z}_{<l}\right)\right)\right] \tag{2.10}$$

Where the first term is the standard data log-likelihood, the second term encourages the lowest level approximate posterior to fit to the lowest level prior and the third term encourages the higher tier posteriors to fit the higher tier priors, while taking into account the inter tier conditional relationships. It is important to note that in order to train this new formulation, a new model architecture is required. As lower levels refer to levels with a higher level of compression a bottom up and a top down model is required, referred to as a Bidirectional Encoder. This is because the posterior at each level is conditioned on the data **and** the posterior of all lower levels. The new architecture implemented by Vahdat & Kautz (2020) is detailed in Figure 2.10.

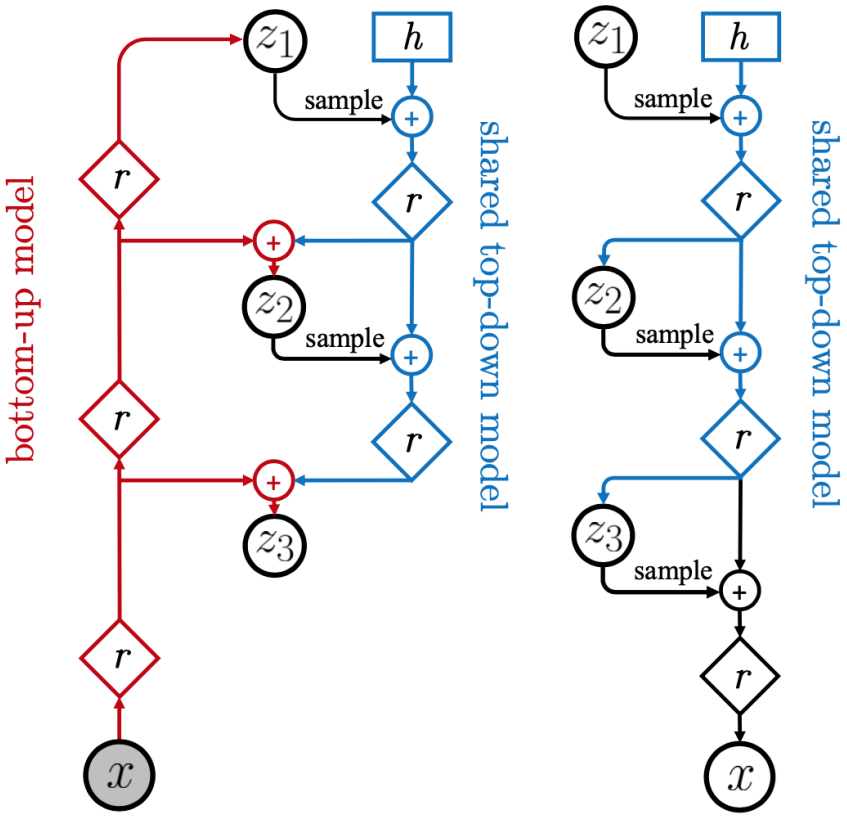
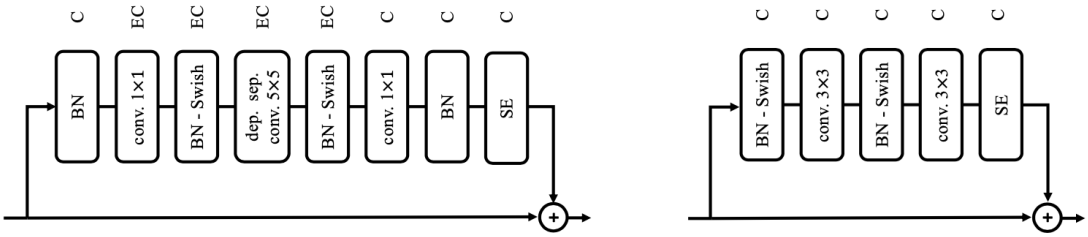


Figure 2.10: Schematic representing the Bidirectional encoder (i.e $Q(\mathbf{z} \mid \mathbf{x})$) on the left and the generative model (i.e $P(\mathbf{x} \mid \mathbf{z})P(\mathbf{z})$) on the right for the NVAE. Here r represents a residual block, $+$ represents concatenation and h is a learnable parameter outputted by the network. Note the skip connections in the top-down model which allows conditioning from all levels. The sampling procedure is achieved through the reparamaterisation trick (detailed in section 2.3.2)(Kingma & Welling, 2013) (Vahdat & Kautz, 2020).

While the theory remains largely unchanged from (Kingma et al., 2016), NVAE achieves state of the art results for non autoregressive likelihood based models through carefully designed engineering tweaks. This demonstrates that VAEs are still a powerful tool for generation and so representation learning tasks. It is not possible to pin down the exact reason why NVAE performs better than its VAE counter parts, rather it can be attributed to a number of tweaks made to the model and training procedure which all contribute in some way to the improved performance. The main point of the paper is that VAEs have historically taken standard deep-learning architectures and training methods

intended for other tasks such as classification, and that large improvements can be made by creating new methods and architectures, tailored to the VAE objective. An example of this is the empirical observation that having different architectures for the encoder and decoder yielded better results. Figure 2.11 details the different residual blocks used for encoder and decoder. This also nicely demonstrates the level of tuning that was implemented in the NVAE.



(a) Residual Cell for NVAE Generative Model

(b) Residual Cell for NVAE Encoder

Figure 2.11: Schematic detailing the different residual blocks for encoder **b)** and decoder **a)** which were empirically determined to be optimal for the NVAE (Vahdat & Kautz, 2020). The decoder revolves around depth separated convolutions with a large receptive field and the encoder is made up of standard 3x3 convolutions. BN refers to batch normalization, Swish refers to the swish activation $f(u) = \frac{u}{1+e^{-u}}$ (Ramachandran et al., 2017) (which in combination with BN was empirically shown to be better than the more standard ELU activations). It is important to note that these residual blocks were optimised even down to the order in which they appeared in the block. Image taken from (Vahdat & Kautz, 2020).

2.4 VQ-VAE

Vector Quantised VAEs (VQ-VAEs) (van den Oord et al., 2017) match standard autoencoder structures in that they consist of an encoder and a decoder which learn a forward and a reverse mapping from an input space onto a compressed space known as a "bottle neck". They differ from standard autoencoders, by replacing the continuous "bottleneck" layer with a discrete layer[19]. This is achieved through the model learning a set of K codebook vectors, $\mathbf{e}_K$, to which outputs from the encoder, $\mathbf{z}_e(\mathbf{x})$, are assigned via K-means clustering,[20] a process known as Vector Quantisation. Through forcing the information to be represented by the indexes of the nearest codebook vectors, a constraint of $\log_2(K)$ bits is applied to the output of the encoder, independent of $\mathbf{z}_e(\mathbf{x})$ dimensionality, D (Łańcucki et al., 2020). Figure 2.12 details the high level VQ-VAE structure. The encoder outputs a $W * H * D$ tensor $\mathbf{z}_e(\mathbf{x})$ which is then discretised to $W * H$ codebook vector indices. A $W * H * D$ tensor is then passed to the decoder, a stack of $W * H$ codebook vectors, corresponding to the selected codebook indices. It is important to note that as this operation is not differentiable, the encoder relies on the straight-through gradient estimator, $\frac{\partial L}{\partial \mathbf{z}_e(\mathbf{x})} \approx \frac{\partial L}{\partial \mathbf{z}_q(\mathbf{x})}$ (van den Oord et al., 2017; Łańcucki et al., 2020).

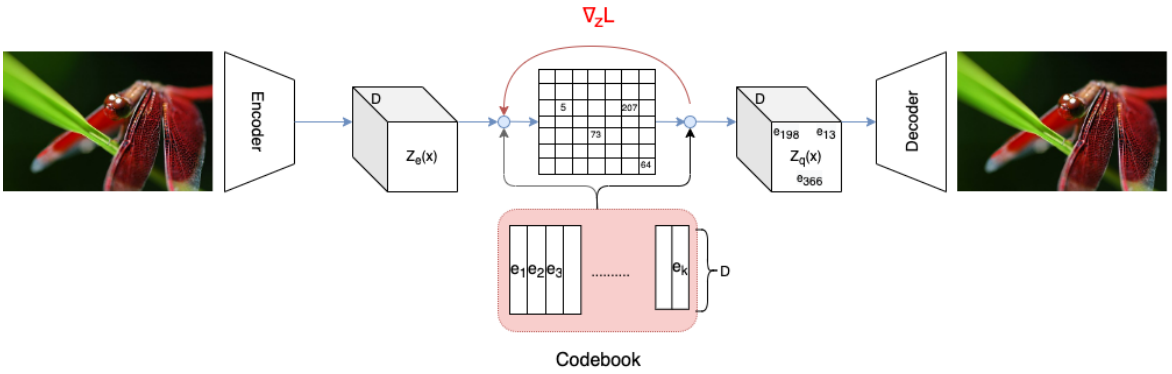


Figure 2.12: Schematic detailing the VQ-VAE structure. Here we can see the encoder mapping an image onto tensor of dimensionality (batch)xlatent_widthxlatent_heightxcodebook_vector_dimensionaliy. This represents latent_widthxlatent_height vectors which exist in the codebook vector space. These are then assigned to the nearest codebook vectors via the vector quantisation operation, detailed by the light blue circles. The data instance is now represented by a two dimensional tensor, latent_widthxlatent_height, with values corresponding to indices of the vectors in the codebook. Before decoding, the corresponding codebook vectors are inserted into the latent, forming once again a (batch)xlatent_widthxlatent_heightxcodebook_vector_dimensionaliy tensor. This tensor is then decoded back to the data space. It is important to note that vector quantisation is not a differentiable operation therefore to train the encoder, the gradients are estimated by the straight through gradient estimator, detailed by $\nabla_z L$

The VQ-VAE loss function has 3 components detailed in equation 2.11.

$$L_{VQ-VAE} = log(p(\mathbf{x}|\mathbf{z}_q(\mathbf{x}))) + ||sg[\mathbf{z}_e(\mathbf{x})] - \mathbf{e}||_2^2 + \beta||\mathbf{z}_e(\mathbf{x})| - sg[\mathbf{e}]|_2^2 \tag{2.11}$$

The first term, $log(p(\mathbf{x}|\mathbf{z}_q(\mathbf{x})))$, is the standard log likelihood, calculated through the expectation of the reconstruction error, commonly the mean squared error (MSE) between the input and reconstructed data. The log likelihood applies to the decoder and through the straight through gradient estimator, to the encoder. It provides a signal by which the VQ-VAE can learn to construct better reconstructions of the data. Because the straight through estimator skips out the discretisation step (due to its non differentiable nature), gradient signals from the reconstruction error are not

[19] van den Oord et al. (2017) argues that real world concepts such as music, language and images are inherently discrete and that logic, reason and planning are built up on discrete concepts. For example "if it rains, I will use an umbrella". It is therefore reasonable to think that latent representations should also be discrete

[20] formally, assigning to the codebook vector corresponding to $\mathbf{z}_q(\mathbf{x})_{ij} = \mathbf{e}_k$ where $k = \arg\min_m ||\mathbf{z}_e(\mathbf{x})_{ij} - \mathbf{e}_m||$) (van den Oord et al., 2017)

passed to the codebook. An additional loss is therefore required to train the codebook embedding space. This is achieved through the second loss term, labelled by the authors as the *codebook loss*, which applies only to the codebook vectors [21]. The codebook loss, $||sg[\mathbf{z}_e(\mathbf{x})] - \mathbf{e}||_2^2$, is simply the L_2 loss of the euclidean distance between output vector of the encoder and the codebook vector to which it was assigned. The gradient signals allow the codebook to alter its vectors so that they point to the average position of all the encoder vectors which are currently mapped to them. The third loss is the *commitment loss*, $||\mathbf{z}_e(\mathbf{x})| - sg[\mathbf{e}]||_2^2$, and only applies to the encoder. This loss encourages outputs from the encoder to move closer or *commit* to the codebook vector to which they were assigned. van den Oord et al. (2017) explains that this confines the encoder outputs to the embedding space which otherwise would be left unconstrained. In addition Razavi et al. (2019) adds that it also assists with encoder-codebook assignment fluctuation, where encoder outputs frequently switch between codebook vector assignments, complicating the decoders task and on a high level preventing each codebook vector from acquiring a clear and well defined semantic meaning.

2.4.1 Autoregressive Prior

The VQ-VAE uses an autoregressive prior to generate new images. Fitting powerful neural network priors to latent codes has yielded great improvements in VAE performance (Chen et al., 2016). It has the benefit that latent variables sampled at test time match the training latents more closely (compared with a static prior), resulting in the decoder producing more reasonable outputs as it is more familiar with the sampled latent variables. In addition, it can be viewed at the lossless compression of the latent space, as fitting the prior to the posterior results in a latent space which matches the true distribution more closely[22] (Razavi et al., 2019). The VQ-VAE utilises Pixel autoregressive models to fit the generated latent space and serve as a powerful prior.

2.4.2 Autoregressive Models

The Pixel family (RNN, CNN, SNAIL)[23] of autoregressive models work by estimating the distribution of a collection of images[24]. This distribution can then be used to generate new images that fit the collection and can also estimate the likelihood of a newly presented image being within the distribution. The key concept is that the model scans the image, pixel by pixel, and predicts the conditional probability of all the possible pixel values, given the context of the previously predicted pixels. To assign a probability $p(\boldsymbol{x})$ to each image, $p(\boldsymbol{x})$ is estimated to be the product of conditional probabilities of all the pixels in the image (van den Oord et al., 2016b;c).

$$p(\boldsymbol{x}) = \Pi_{i=1}^{n^2} p(x_i|x_1, x_2, ..., x_{i-1}) \tag{2.12}$$

Where n is the dimension of the image and $p(x_i|x_1, x_2, ..., x_{i-1})$ is the probability of the pixel i given all previous pixels. This results in the current pixel value being predicted conditioned on all the previous pixels which have come before, and the task can now be viewed as a sequencing problem. It is important to note that the Pixel family models the pixel space as a discrete distribution rather than a continuous one, and this makes it particularly useful for modeling the discrete latent space generated by VQ-VAEs.

To implement this complex sequencing problem, with highly non linear and long range dependencies, three main architectures have been suggested. The first is the PixelRNN(van den Oord et al., 2016b), which uses methods from the Recurrent Neural Network (RNN) field of research to address the problem. Namely a Long Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997; Graves & Schmidhuber, 2009) recursive unit, which recursively alters a hidden state at

[21] The *codebook loss* doesn't effect the encoder through the stop gradient, sg[], operator. Implemented in PyTorch through the `.detach()` function.

[22] i.e why should you force the latent space to take the shape of a simple gaussian? when a more complex structure may suit/compliment the true structure of the data better.

[23] Here autoregressive models are explored with respect to image generation only, however, have been used to model a range of different data modalities for example audio (van den Oord et al., 2016a)

[24] or in the case of VQ-VAE a collection of latents

each new input state through a series of learnt gates. Figure 2.13 details two LSTM variants implemented in the original paper (Row LSTM and Diagonal BiLSTM). This has the benefit that the receptive field completely encompasses all previous states in the sequence, however due to the model having to compute each pixel likelihood at each pixel location it is extremely slow to train and sample.

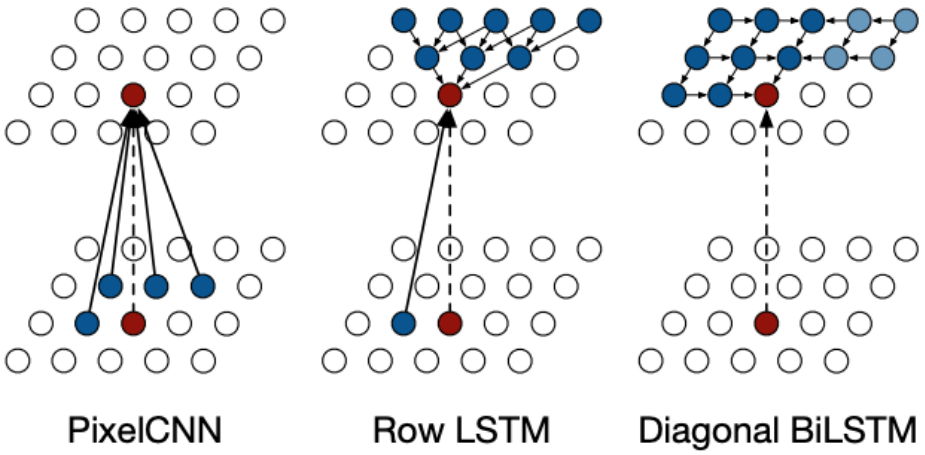


Figure 2.13: Schematic detailing the original formulations of different methods to model the conditional dependencies in the Pixel family of autoregressive models. Here the bottom and top row represent the input and hidden states respectively. Solid black lines detail conditional dependencies, modeled by the architecture. Image taken from (van den Oord et al., 2016b)

The second is PixelCNNs, which rather than modelling the dependencies in the image with recurrence, utilised convolutional operations (van den Oord et al., 2016c;b). Stacks of masked convolutional layers yield large receptive fields, allowing the model to compete with PixelRNNs. van den Oord et al. (2016c) achieved this through introducing a novel formulation of two convolutions which when combined, result in a receptive field which encompasses all previous pixels, please see Figure 2.14. This strategy was an improvement on PixelRNNs in that it allowed for parallelism during training without this leading to a drop in performance (it also avoids the problem associated with LSTMs of catastrophic forgetting (Sodhani et al., 2018)).

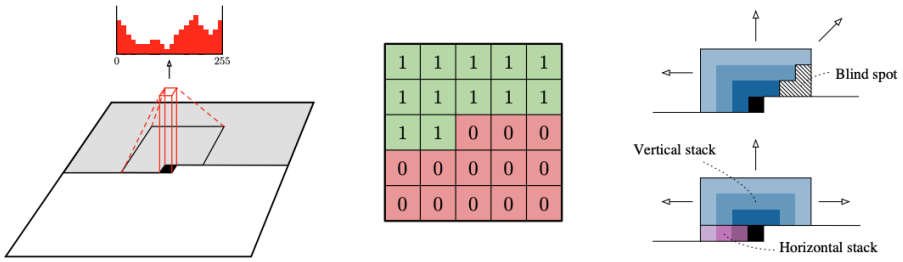


Figure 2.14: Schematic detailing the convolutional operation implemented in the reformulated PixelCNN. On the **left** is a high level overview of the convolutional operation which predicts the value of hidden state i based on the masked convolutional operation of the pixels/hiddens states of the layer below, which through multiple layers has a (indirect) receptive field which encompasses all previous predicted pixels. **middle** details an example mask for a 5x5 convolution which ensures that only the previously predicted pixels are incorporated into the conditional prediction. **bottom right** shows how a novel combination of two convolutions, one purple, one blue, can be used to circumvent the issue of the blind spot when only one convolution is used (**top right**) where pixels to the right of the current position are not taken into account in the prediction. Image taken from (van den Oord et al., 2016c)

The final formulation of the Pixel family is PixelSnail (Chen et al., 2017). This builds on the PixelCNN by combining the gated convolution architecture with self attention (Vaswani et al., 2017). Pixel predictions are conditioned on the previous sequence of pixels through a combined operation of convolution and attention. This yielded state of the art performance in likelihood based image generation tasks, thanks to self attention's superior performance in modeling long range dependencies of sequences. It is this model which was used subsequently in Razavi et al. (2019), see section 2.4.3.

2.4.3 Hierarchical latent space

Razavi et al. (2019) introduced VQ-VAE-2. VQ-VAE-2 differs from VQ-VAE in one main aspect: it has a hierarchical space. The basic structure consists of a bottom and a top latent space, tasked with representing local (e.g. texture) and global information (e.g. structure) respectively. Figure 2.15 depicts the high level structure of the VQ-VAE-2 along with the general training principles. It can be useful to compare this figure with the original VQ-VAE structure detailed in Figure 2.12. Independently focusing on short and long range patterns/structures allowed the VQ-VAE-2 to compress data to a further extent, allowing for high resolution images to be compressed by over 30 times (Razavi et al., 2019) while maintaining all the important information.

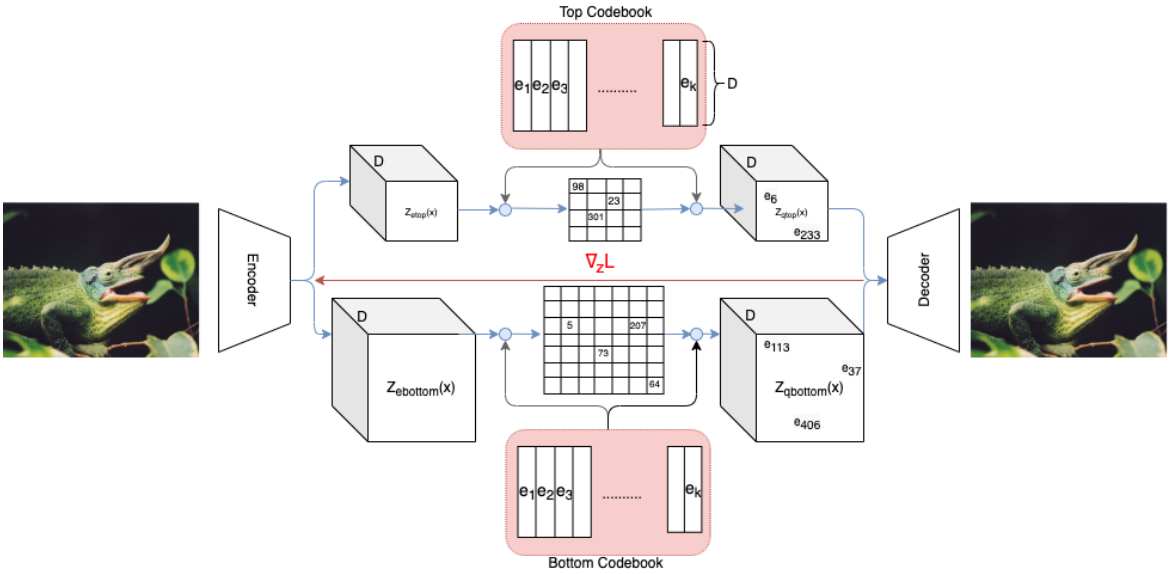


Figure 2.15: Schematic detailing the VQ-VAE-2 hierarchical structure. It revolves around the same principles as the VQ-VAE, depicted in Figure 2.12 apart from that there are now two tiers, representing varying levels of compression. The top tier is intended to extract long range concepts such as structure where as the bottom tier is supposed to focus on short range dependencies such as texture.

When two autoregressive Pixel Snails were fitted to the top and bottom tier latent spaces, generated images rivaled that of Generative Adversarial Networks with the added benefit of not being subject to mode collapse as the Pixel Snails were trained via log likelihood methods (therefore tasked with modelling the whole data distribution)(Brock et al., 2018; Razavi et al., 2019).

2.4.4 Codebook loss replacement

Codebook loss replacement was originally suggested as an improvement in the original VQ-VAE paper (van den Oord et al., 2017) and implemented in (Razavi et al., 2019; Dhariwal et al., 2020). Since the *codebook loss* only applies to the codebook, rather than update it with stochastic gradient techniques, it can be simply updated with the exponential moving average of the encoder outputs. Here the codebook vectors are updated to match the exponential moving average of the encoder output vectors which were assigned to them during VQ. The new value of codebook vector $\mathbf{e}_i$ at time t is calculated

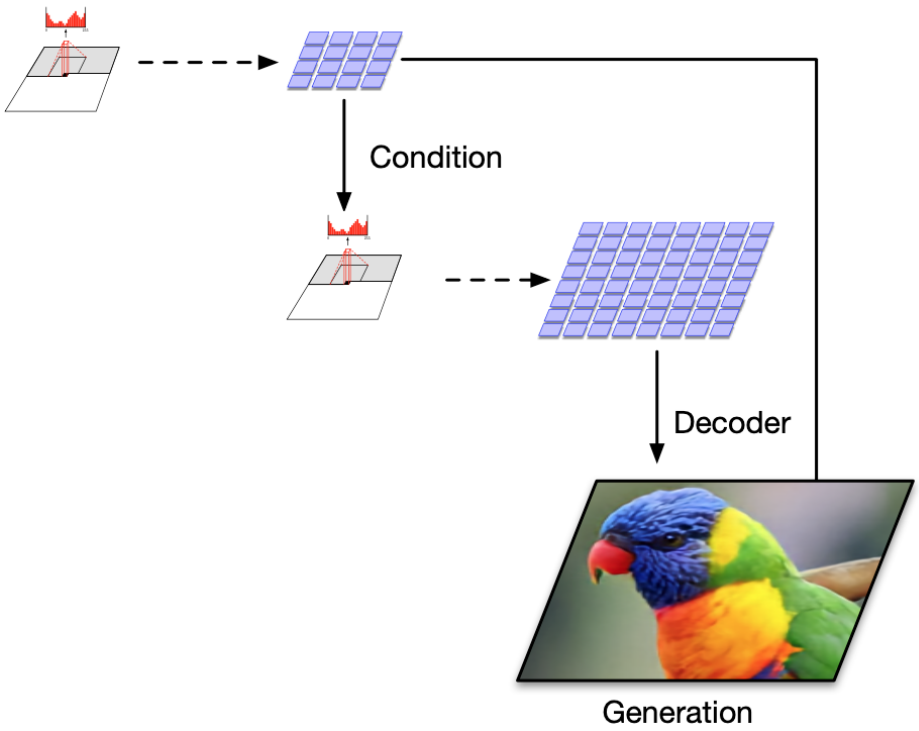


Figure 2.16: Schematic detailing the hierarchical sampling procedure to generate new images. Here two PixelCNNs conditioned on the image class (the bottom PixelCNN is also conditioned on the top latent) sample each discrete latent one position at time, before both top and bottom latents are passed to the decoder.

via Equation 2.13, where $E(\mathbf{x})$ are the vector outputs from the encoder which were assigned to that particular codebook vector, n_i^t is the number of encoder output vectors assigned in that minibatch, and γ is a hyper-parameter which controls the rate at which the average is updated by new entries. The value of γ used universally in all VQ-VAE-2 related implementations is 0.99.

$$N_i^{(t)} := N_i^{(t-1)} * \gamma + n_i^{(t)}(1-\gamma), \quad m_i^{(t)} := m_i^{(t-1)} * \gamma + \sum_j^{n_i^{(t)}} E(\mathbf{x})_{i,j}^{(t)}(1-\gamma), \quad \mathbf{e}_i^{(t)} := \frac{m_i^{(t)}}{N_i^{(t)}} \tag{2.13}$$

2.4.5 Forcing the use of all latent tiers

Recently in (Dhariwal et al., 2020), it was observed that VQ-VAE-2 models often learn to rely completely on the bottom latent. This phenomenon occurs because the bottom bottleneck is much less restrictive on the information flow than the top tier bottleneck, and as a result the model simply pushes all the information through the bottom latent. The reason for this is that the model can in this way avoid the task of learning the relatively harder representation required for information to be transferred through the top latent. This is an issue as in ignoring the top latent the VQ-VAE-2 can still achieve very low reconstruction errors while still failing to learn the highly abstracted representations that the top latent was intended for. This effect was also observed in this project and is detailed in section 4.8. To combat this, Dhariwal et al. (2020) trained separate autoencoders, with different sized latent spaces. As each had to independently achieve low reconstruction errors, each was forced to utilise the latent space and so learned data representations appropriate to the dimensionality constraint. A draw back to this however may be that as they are independent, they can no longer learn complimentary information and so information will be repeated, reducing the efficiency of the model.

2.4.6 PREVENTING CODEBOOK COLLAPSE

Codebook collapse is when the VQ-VAE resorts to using only a small proportion of the codebook vectors (Dhariwal et al., 2020; Łańcucki et al., 2020). This is an issue as it limits the expressability of the latent space at the bottleneck. The model often gets stuck in this sub optimal state for two reasons: the encoder, through the *commitment loss* (please see Equation 2.11), is encouraged to commit only to the codebook vectors to which it has already been assigned and therefore will not learn to start favouring other codebook vectors which have yet to be assigned. The second reason is that only the codebook vectors which have been assigned encoder output vectors will be updated through Equation 2.13, therefore if a codebook vector has not been assigned an encoder output its position in the embedding space will not change. As a result, a significant portion of the codebook will not be utilised. We the authors observed this effect in the current project, detailed in section 4.2

Dhariwal et al. (2020) addresses this problem through "random restarts", where the usage of each codebook vector is monitored, and if the usage is/falls below a certain threshold, it is randomly reset to one of the encoder outputs in that minibatch. This ensures that this codebook vector will now have a signal to learn by and that all the codebook vectors are used.

Łańcucki et al. (2020) takes a geometric approach. If the encoder outputs are smaller in norm than the codebook vectors, then they are often trivially assigned to the smallest codebook vector, leading to many codebook vectors not being utilised. This is shown in Figure 2.17 on the right. To address this problem they enforce (through batch normalisation) the encoder outputs to be larger in norm than the codebook vectors. They demonstrate this results in a better utilisation of the whole codebook. This can be explained through the assignment now being based on the radial distance (as opposed to just distance). Figure 2.17 geometrically details this new formulation on the left, demonstrating how the new formulation leads to more uniform asignment.

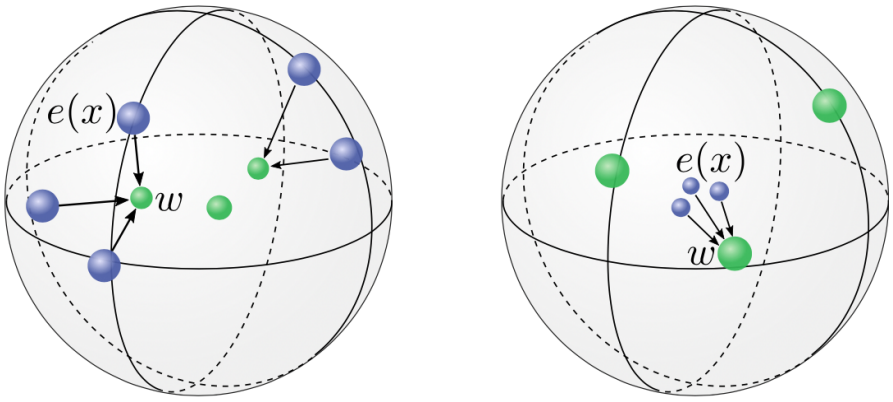


Figure 2.17: Schematic detailing Łańcucki et al. (2020)'s interpretation of the effect of the relative sizes of the encoder outputs and the codebook vectors. Through batch normalisation if the encoder outputs are set to have a larger norm than the codebook vectors (as seen on the left) an even distribution of assignments can be observed. However, if encoder outputs are smaller in norm than the codebook vectors, all are assigned to the smallest codebook vector. Image taken from (Łańcucki et al., 2020)

2.4.7 SOFT-VQ-VAE

soft-VQ-VAE (Wu & Flierl, 2020), with the aim of forming better representations, combines aspects of quantisation and denoising schemes. Bayesian optimisation is utilised to estimate the quantised based representation of a noisy encoder output. The output of the Bayesian estimator is a combination of all the codebook vectors in the codebook, weighted by the probability of assignment and is defined

by:

$$\hat{\mathbf{z}}_q = E\left[\mathbf{z}_q \mid \mathbf{z}_e\right] = \sum_{k=1}^{K} \mathbf{e}_{(k)} p\left(\mathbf{e}_{(k)} \mid \mathbf{z}_e\right) \tag{2.14}$$

The encoder outputs a noisy output along with a variance for each codebook vector. The Bayesian estimator then assigns a probability to each of the codebook vectors, based on the codebook vector variance and its distance from the encoder output and outputs a convex combination of them. When tested on the MNIST data set, using a Multilayer Perceptron (MLP) architecture the soft-VQ-VAE yielded better representations than both VQ-VAE and VAEs. This was demonstrated through a range of downstream tasks. However, these results were not replicated when the dataset was scaled up in complexity to the CIFAR10 dataset (32x32). In order to prevent extremely poor results of the soft-VQ-VAE the number of codebook vectors (K) had to be reduced to 32. When this was done it produced better representations that the standard VQ-VAE however, this standard VQ-VAE had a codebook vector size of 512. Unfortunately, for this reason it is not possible to claim that the Bayesian Estimator caused the improved representations. As detailed in Section 2.2.2 reducing the capacity of the bottleneck often leads to better representations being formed and is therefore a potential source for the improved representations. In addition, I would argue that this formulation is no longer a VQ-VAE because discrete latents are no longer being fed to the decoder, rather the decoder receives a continuous combination of the discrete codebook vectors. With a standard VQ-VAE the codebook vectors represent discrete concepts which can be passed to the decoder but with soft-VQ-VAE they are never explicitly passed to the decoder and instead form a space where a continuous set of vectors can be sampled from and decoded (generated through Equation 2.14).

2.4.8 Hierarchical Quantised Autoencoders

During the data collection stage of this project, a new variation of the VQ-VAE was published, namely Hierarchical Quantised Autoencoders (HQAs) (Williams et al., 2020). They too, independent to this MSc thesis, identify deterministic quantisation to be an issue in forming representations with the VQ-VAE. HQA adapts VQ-VAE-2 through two alterations. The first is through stochastic quantisation. They make use of the differentiable Gumbel Softmax where the probability of an encoder output $\mathbf{z}_e(\mathbf{x})$, being mapped to a given codebook vector $\mathbf{e}_k$ is inversely proportional to the distance between them. (Jang et al., 2016)

$$q(z = k \mid \mathbf{x}) \propto \exp - \|\mathbf{z}_e(\mathbf{x}) - \mathbf{e}_k\|_2^2 \tag{2.15}$$

The extent of stochasticity is reduced over the course of training, through reducing the temperature parameter of the Gumbel Softmax operation. At low temperatures the expected value from the Gumbel Softmax becomes a hard assignment. At high temperature values the Gumbel Softmax is a smooth distribution over the codebook vectors, resulting in the quantisation becoming a soft assignment. The authors highlight that as $\mathbf{z}_e(\mathbf{x})$ now implicitly defines the categorical distribution over all codebook vectors, it carries more information about the data sample. Throughout training they reduce the temperature parameter until the problem reverts to the original quantisation assignment. This differs from the quantisation noise introduced as Contribution in this MSc thesis, described in section 3.1, in that it gradually reduces to a hard assignment during training and it doesn't enforce a semantic ordering with respect to the codebook indices.

The second alteration described by Williams et al. (2020) restructures the hierarchical structure of VQ-VAE-2 into independent VQ-VAEs which are built on top of the latent space created by the VQ-VAE of the tier below. The structure can be seen in Figure 2.18. This is also similar to the hierarchical structure developed in (Dhariwal et al., 2020) except that in this case the separate tiers were built up from the raw data space every time. These two reformulations resulted in a higher level of compression being possible for a given amount of reconstruction loss and as in (Dhariwal et al., 2020), allow for the constraint at each tier to be enforced, yielding better representations.

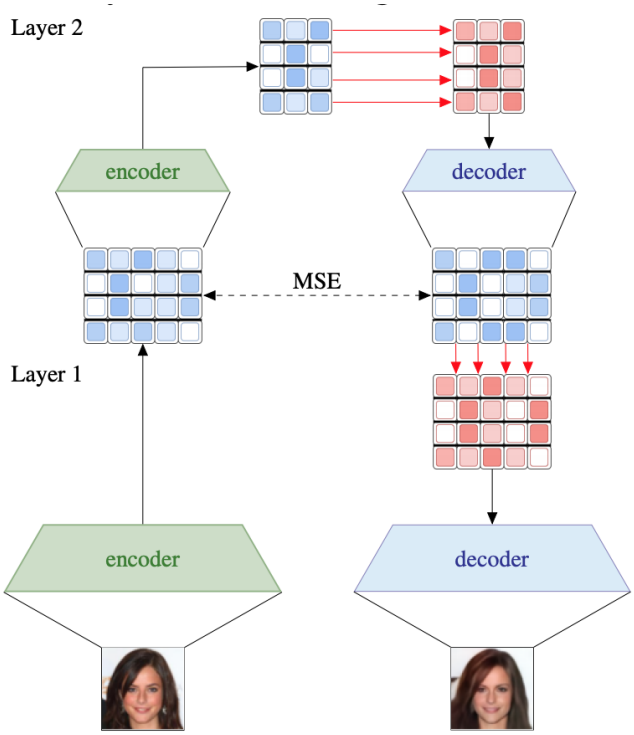


Figure 2.18: Schematic detailing the high level structure of HQA. The key difference in this formulation over the vanilla VQ-VAE is that each layer is trained separately and is trained to minimise the MSE between the input, whether that is an image or a latent representation, and it's reconstruction. Image taken from (Williams et al., 2020)

2.5 GENERATIVE ADVERSARIAL NETWORKS

Generative Adversarial Networks (GANs) are a form of generative modeling where the problem is formulated as a zero sum game between a generator, G, which models the data distribution, and a discriminator, D, which estimates the probability that a sample is from the real distribution (is real) or from a fake distribution (is fake) (Goodfellow et al., 2014; Pfau & Vinyals, 2016). The training objective is for the discriminator to accurately distinguish between real and fake instances and for the generator to maximise the miss classification of the discriminator i.e fool the discriminator (The discriminator is now the objective function for the generator). As both G and D improve at their respective tasks i.e. create more realistic data instance and get better at distinguishing between real and fake respectively, the result is that both are faced with an ever more challenging problem. This forces both the models to learn more complex and sophisticated aspects of the data distribution. The basic/original objective is detailed here:

$$\min_G \max_D E_{\mathbf{x}\sim p(data)}[logD(\mathbf{x})] + E_{\mathbf{z}\sim p(\mathbf{z})}[1 - log(G(\mathbf{z}))] \tag{2.16}$$

Training a GAN can be a very unstable process. GANs consist of two models that are playing a non-cooperative game with each other and due to the two models being updated independently, there is no guarantee of convergence (to the Nash equilibrium) (Salimans et al., 2016b). In addition, as GANs are implicit generative models, in that the model's distribution is not seen/controlled and that the loss function is defined by the discriminator and not by more conventional likelihood measurements, the generator can achieve a low loss by only modelling small areas of the distribution of the data. This is known as mode collapse, and is very easily identified as the same image is

repeatedly produced by the generator(Salimans et al., 2016a).[25]

GANs can also experience issues converging. A reason for this is that at the start of training the generated and real distributions will be very different, this means that the two distributions may not overlap at all, resulting in it being very easy to tell the difference between real and fake data. The result of this is that the gradient signals fed to the generator point in random directions, making it very hard for the generator to learn anything (Arjovsky & Bottou, 2017). To address this problem two major orthogonal contributions to improving to GANs have been made.

The first was to use Wasserstein distance as a distance metric between the two distributions instead of the the Jenson-Shannon divergence used in the original GAN paper (Goodfellow et al., 2014) (Arjovsky et al., 2017). The Wasserstein distance or Earth-Mover distance re-frames the comparison to (intuitively) what is the cost of optimally moving all the probability mass (or earth) from one distribution to the other. Key, is that the Wasserstein metric provides continuous and useful gradient signals no matter what the difference or distance between the two distributions is. This is extremely helpful for GAN training as even if the discriminator (named "critic" in this case) is easily distinguishing between real and fake images the generator can still learn. The difference in gradient signals produced by two optimal discriminators (original (Equation 2.16) and Wasserstein) distinguishing between two Gaussian distributions is demonstrated in Figure 2.19. It is clear that the Wasserstein discriminator provides useful gradients for all areas of the one dimensional explorable space, where as the standard GAN gradients quickly vanish/ are not useful for the majority of the search-able space.

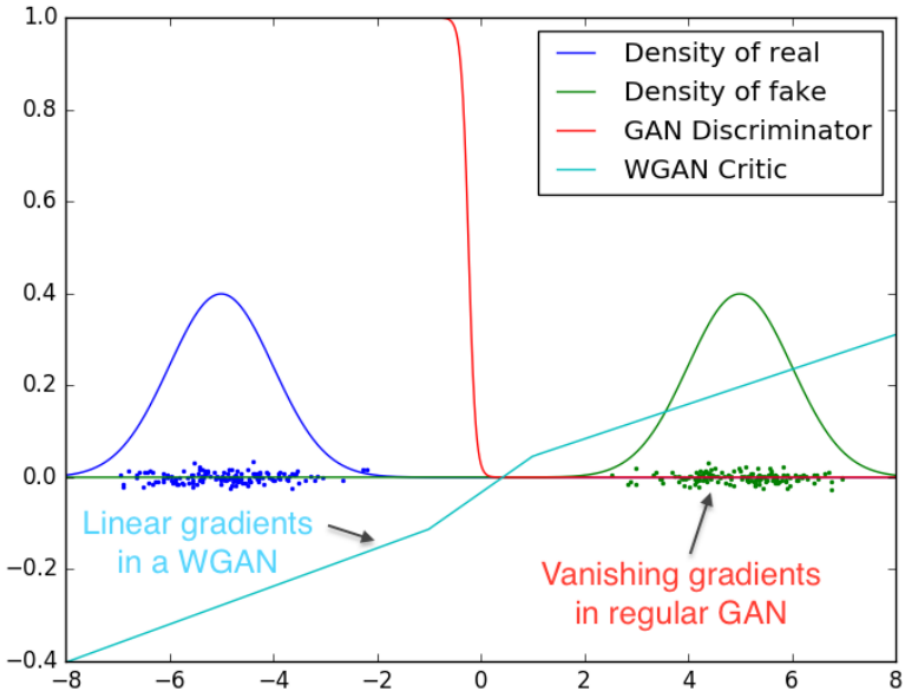


Figure 2.19: Schematic detailing the different gradient signals generated from optimal discriminators between a standard GAN with Jenson-Shannon divergence and a Wasserstein GAN (the discriminator has been labeled a critic in this case). Here the discriminators have been tasked with distinguishing between the blue (real) and green (fake) distributions. The teal and red lines represent the discriminator's/critic's output by which the generator has to learn. Image taken from (Arjovsky et al., 2017)

[25]Engineering techniques can reduce the impact of model collapse such as minibatch discrimination where variance within sampled batches is promoted(Salimans et al., 2016a) but, unlike likelihood based models, the model is still not explicitly pushed towards modelling the full data distribution.

The second was based on the observation that for high resolution image generation, generation was initially a harder task than discrimination, in that the generator had a much harder time creating realistic data instances compared to the discriminator's task of distinguishing between them (Odena et al., 2016). This meant that at the start of training the generator struggled to get a foot hold to learn by, exacerbating the gradients problem discussed above. To combat this Karras et al. (2017) developed a novel training paradigm which started the training process on low resolution images and *progressively* grew the resolution of the the image and the capacity of the model throughout training. This dramatically increased training stability and resulted in state of the art performance in a range of image generation tasks (Karras et al., 2017). A high level overview of the Progressive Growing GAN (PGAN) can be see in Figure 2.20.

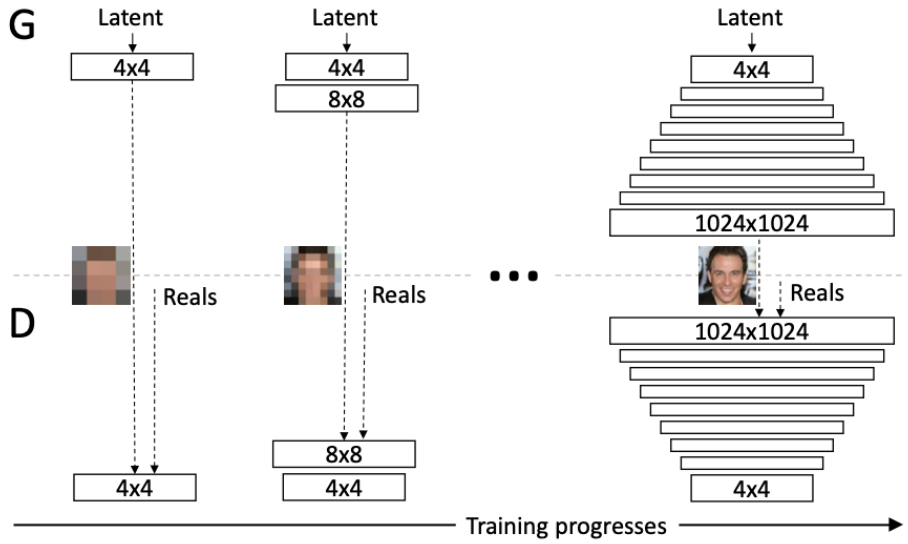


Figure 2.20: Progressive growing GAN training procedure, detailing the progressive addition of layers to the discriminator (D) and generator (G) with the corresponding increase in image resolution. Note that the X refers to the resolution of the image that the convolutional layers operate on. Image take from (Karras et al., 2017)

Additional improvements have been made surrounding the work related to DeepMind's BigGAN (Brock et al., 2018) however, due to the size of the model used and the compute power required to run, We decided not to pursue these techniques as their applicability to this project is questionable given our limited compute resources.

2.5.1 Video generation

Imagination and planning require the ability to construct scenarios which are convincing representations of genuine events in the real world. This can be interpreted as the ability to generate realistic videos. It can be achieved through training a model, such as a LSTM (Hochreiter & Schmidhuber, 1997; Graves & Schmidhuber, 2009) in a compressed representation, to model the dynamics of the world and allow an agent to interact with this model as is done with World models (See section 2.1.3). While far from being implemented as a method for agents to imagine future scenarios, due to little work exploring action conditioning, an important avenue to consider is the field of video GANs as they currently yield the state of the art performance in video generation.

Despite being state of the art, successes experienced by GANs with image generation have not been experienced with video generation. The main reason for this is the increase in data complexity. A one second video can correspond to many hundreds of frames (or a stack of hundreds of images). Not only does this create memory issues but it has been shown that using 3D convolutions (*widthxheightxtime*) results in a highly curved loss surface (compared to when 2D convolutions are used), resulting in a significant drop in stochastic gradient descent optimisation efficiency (Kahembwe &

Ramamoorthy, 2019).

Some of the most successful techniques/models that improve on the state of the art in GAN video generation decompose the task of video generation into smaller sub tasks. Temporal GAN's generator consists of a temporal generator which first samples a series of conditioned latent variables, one for each frame, representing the trajectory of the scenario, before then passing these latents to the image generator which using these latents to produce a video (Saito et al., 2016). Temporal GAN can be seen in the schematic in Figure 2.21. In addition to this they used Wasserstein loss for training stability, with a novel technique named *Singular Value Clipping (SVC)* to ensure 1-Lipschitz constraint by clipping all parameter values to have singular values less than or equal to one (Saito et al., 2016).

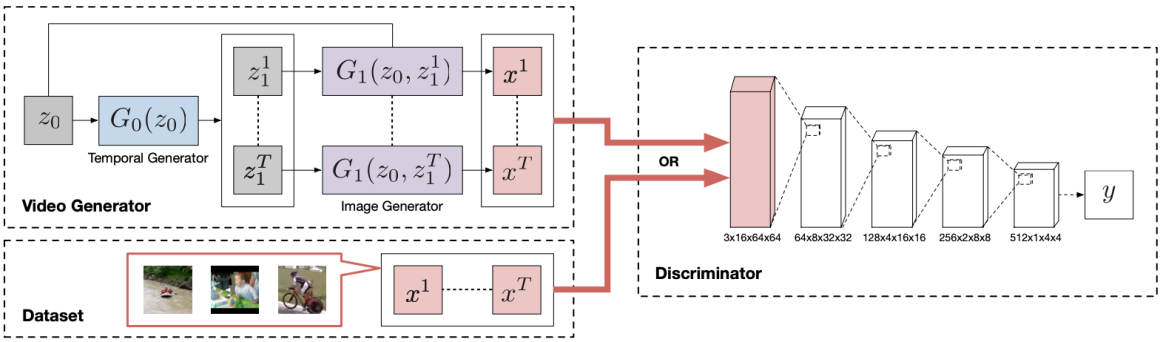


Figure 2.21: Schematic illustrating Temporal GAN. The key feature is the two step generation, first generating the trajectory of latents and then the images/frames conditioned on these latents. Image taken from (Saito et al., 2016)

MoCoGAN (Tulyakov et al., 2017) forms a decomposed representation of the video, made up of a motion (Mo) subspace and a content (Co) subspace. Generation is a two stage process. First a content vector/latent is sampled. This content vector/latent remains constant throughout the video generation process. It is concatenated with varying motion vectors/latents at every frame stage. These motion vectors/latents are generated by a recurrent neural network. These series of latents are then mapped to images by a separate neural network and then subsequently stacked to form the video. MoCoGAN produces state of the art results for low resolution videos. Figure 2.22 details the range of freedom and control on the generated videos this configuration provides, along with demonstrating the logic behind the architectural configuration.

Extremely deep and powerful residual models, leveraging the successes of BigGAN (Brock et al., 2018) have applied similar architectures to the challenge of video generation (Clark et al., 2019), achieving state of the art results in a range of complex video datasets. Along with throwing a massive amount of compute power at the task, a new form of discriminator was developed and termed the *duel discriminator*. The dual discriminator consists of a Spatial Discriminator D_s, which critiques single frame's/image's structure and content, with frames/images being sampled at random from the video, and a Temporal Discriminator D_T which critiques a temporally down-sampled version of the whole video, and thereby ensures temporal coherence.

Other works have addressed the added complexity of 3D kernels discussed above by replacing the 3D kernels with lower dimensional kernel approximations (Kahembwe & Ramamoorthy, 2019). This dramatically improves training stability, memory and computation efficiency, resulting in state of the art performance for single GPU video generation.

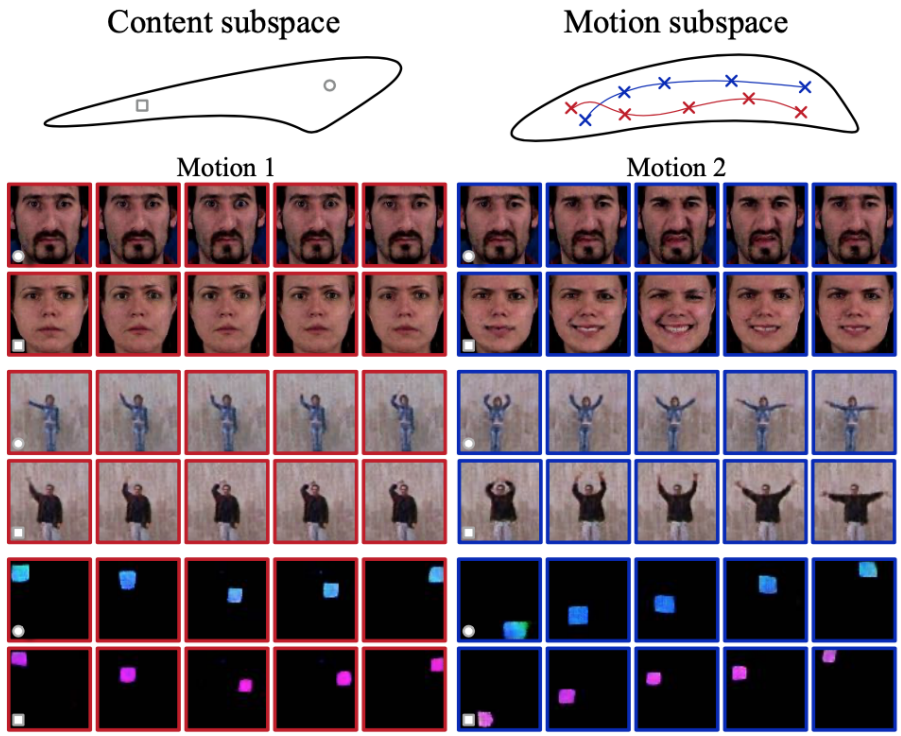


Figure 2.22: Schematic detailing the separate content and motion subspaces of MoCoGAN, illustrating how different latent content and motion vectors can be combined to yield intuitive results. Image taken from (Tulyakov et al., 2017)

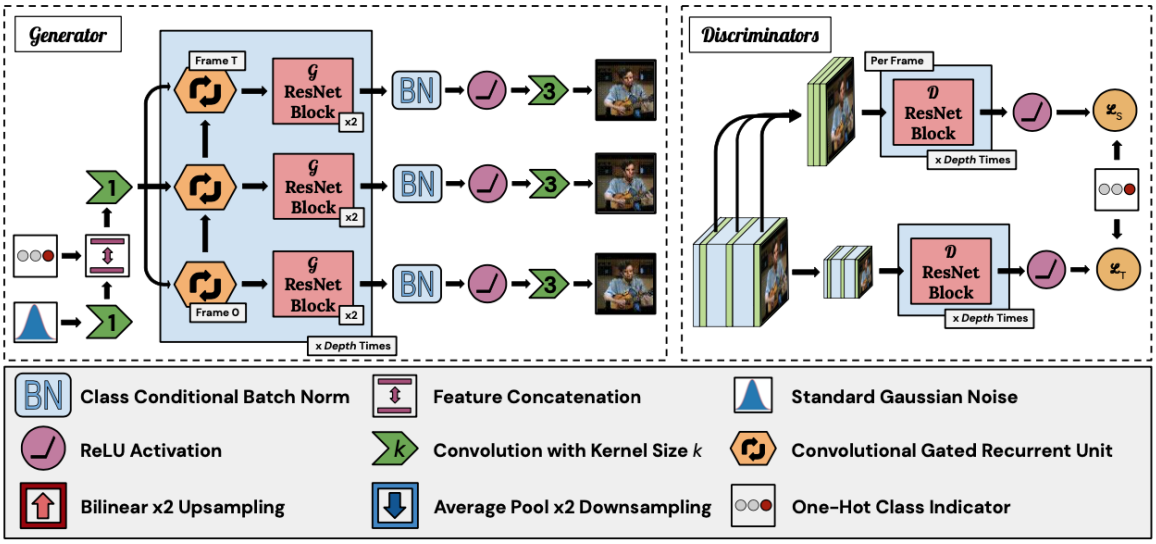


Figure 2.23: Schematic detailing the architect of DVDGAN. The most important part to note is the dual discriminator, consisting of D_s and D_T tasked with ensuring that samples fit the spatial and temporal distributions of the video data. Image taken from (Clark et al., 2019)

Chapter 3

Contribution

3.1 Noise Injection

Due to the formulation of the VQ-VAE, the codebook has no semantic structure. Codebook vectors which lie next to one another in the codebook order are no more semantically linked than two codebook vectors positioned far away from each other. The explanation for this is that the codebook vectors in the latent space are randomly initialised. There is therefore no correlation between codebook vector ordering and position. For this reason when fitting autoregressive priors on the discrete representations for data generation, the output layer has to be made up of a depth wise softmax layer, requiring a depth of, K (number of codebook vectors). In many cases this is as high as 512-1024. This adds a layer of complexity to the latent space which contrasts with operating in the pixel space, where pixel values that are of a similar value represent similar hues. Constructing representations in this unstructured space is counter productive as in order to use these representations one must first decipher the complicated and random relationships between the discrete latent vectors within the codebook. The unstructured nature of this data space can therefore be viewed as entangling the latent representations formed.

We attempted to combat this issue by employing a simple regularisation method. This method consists of a noisy codebook vector assignment in the Vector Quantisation stage. Figure 3.2 details how we added sampled discrete Gaussian noise to the quantised codes during training. We hypothesised that the addition of noise would force neighbouring codebook vectors to be more semantically related than those far away from each other, as there would now be a small chance that they could be interchanged. This can be linked to the high level explanation of why β-VAE is an effective method for forming well structured latent spaces, detailed in Section 2.3.3 and Figure 2.9. Figure 3.1 compares the normal and noisy codebook vector assignment in the embedding space. It is important to note that with this implementation we decided that codebook EMA updates should affect the final codebook vector that was assigned (after the noise had been added). This encourages the codebook vector which has been noisily assigned, to move towards the encoder output, thereby directly restructuring the codebook into a more semantically smooth space. In contrast, we set the *commitment loss*, as depicted in Figure 3.1, to encourage the encoder to commit to the original codebook vector (before noise was added) in other words the mean of the noise is favoured. Without this manipulation to the commitment loss, the encoder would receive a noisy learning signal and so would struggle to commit to an assignment.

We found that adding noise in the quantisation stage has an additional benefit. As alluded to in Section 2.4.6 VQ-VAEs often fall into a local optimum of not using all the codebook vectors (termed codebook collapse). This is due to the VQ-VAE learning signals only affecting the codebook vectors which are currently being used, therefore redundant codebook vectors are never updated and so remain unused. As discussed in Section 2.4.5, OpenAI's Jukebox (Dhariwal et al., 2020) combats this through "random restarts" where if a codebook vector remains unused it is randomly set to one of the

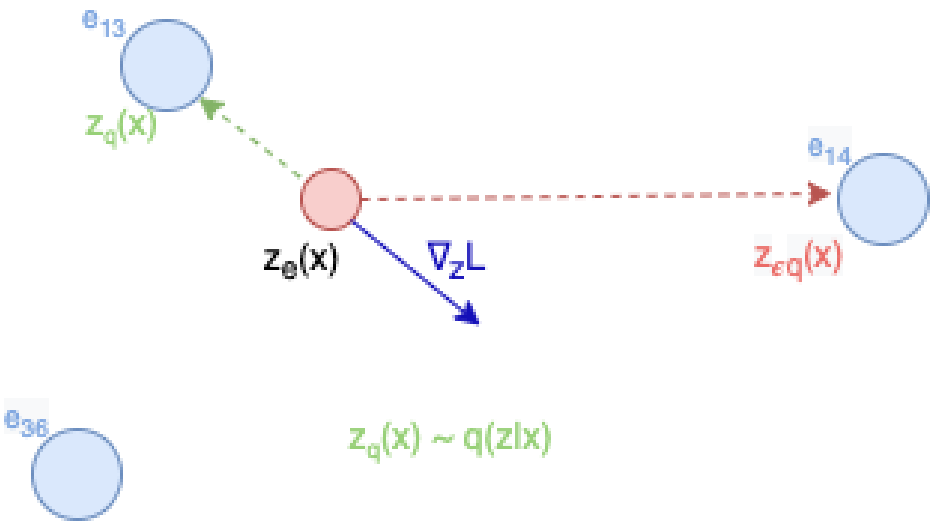


Figure 3.1: Schematic detailing standard (green) and noisy (red) vector codebook assignment in the VQ-VAE-2 for one of the discrete latents in embedding space. Here $\mathbf{e}_x$ represents the codebook vectors and $\mathbf{z}_e(\mathbf{x})$represents one of the outputs from the encoder. $\mathbf{z}_q(\mathbf{x})$ and $\mathbf{z}_{eq}(\mathbf{x})$ represent the non-noisy and noisy codebook assignments respectively. $\nabla_{\mathbf{z}} L$ is the commitment loss which encourages the encoder to map closer to the assigned codebook vector and stop it from fluctuating between assignments.

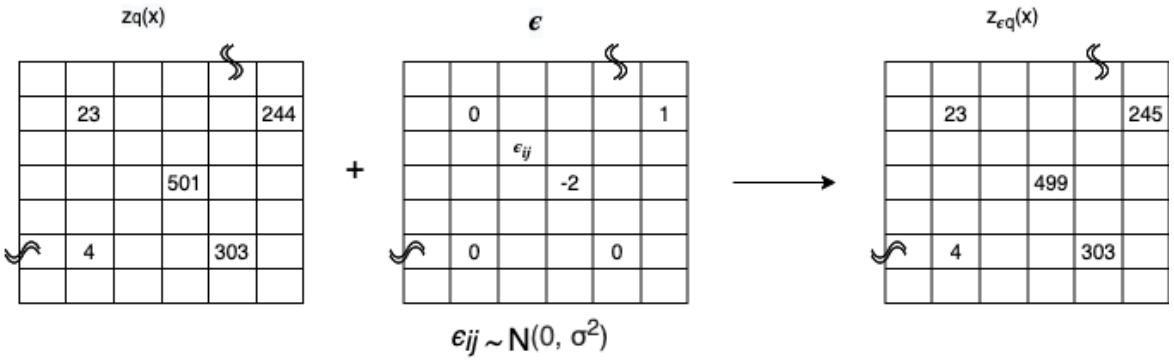


Figure 3.2: Schematic detailing the method to add noise to the quantisation assignment. $z_q(x)$ details the intended quantisation, if no noise is added. ϵ details the noise which is added. Here ϵ is sampled from a discrete Gaussian distribution with mean 0 and variance σ. $z(x)$ details the quantised latent after the noise has been added. Recall that the numbers refer to the indices of codebook.

encoder's outputs. We found that our regularisation technique of adding noise in the quantisation stage also has the effect of stopping codebook collapse. Imagine a codebook vector which is not being used. With our approach, there is a high chance that at some point during training, due to the quantisation noise, an encoder output will be re-assigned to it from its intended codebook vector (this codebook vector index will equal the currently redundant index minus the quantisation noise). Now that this codebook vector has been assigned to an encoder output it will have a signal via the EMA update to learn by and so alter its value. Due to the iterative nature of machine learning training, one can be confident that this will ensure that all codebook vectors are utilised and not left out.

3.2 FITTING A GAN PRIOR

With standard VAEs, data instances can be easily generated by sampling from the prior because the approximate posterior has been forced to resemble the static sampling prior (Kingma & Welling, 2013; Doersch, 2016), in most cases a simple Gaussian (see section 2.3). From this it can be inferred that the manifold occupies a large volume of the prior space. As the VQ-VAE objective lacks any constraint to fit the approximate posterior to a sampling prior, sampling from the latent space will not yield meaningful data instances. Therefore van den Oord et al. (2017); Razavi et al. (2019) fit a powerful autoregressive model (PixelSNAIL (Chen et al., 2017; van den Oord et al., 2016b)) to the latent space. The need for such a powerful model to serve as the prior is another demonstration of the level of entanglement in the latent space, discussed in Chapter 4.

In the field of fitting models to complex distributions, GANs are also a very successful technique. It was therefore deemed important to investigate the applicability of GANs to serve as a prior to the learned VQ-VAE discrete latent space. In addition to this, once a mental model of the world has been constructed, it may be used to perform a wide range of tasks, therefore, a range of generative techniques should be compatible with it. A preliminary study was conducted, implementing a Deep Convolutional GAN (DCGAN) (Radford et al., 2015) on each of the top and bottom latent spaces. Despite scaling up the size of the network along with a limited hyperparameter search, the DCGANs immediately diverged during training.[1] To combat this instability, two Progressive Growing GANs (PGANs) (Karras et al., 2017), were implemented on the top and bottom latent space. The combined effect of progressively increasing the resolution of the latents, combined with the Wasserstein loss [2] yielded a more stable structure which was able to capture the latent space dependencies[3]. The final structure and training procedures are detailed in Figure 3.3. Figure 3.3a) details the progressive growing nature of the training procedure for the PGAN fitted to the top latent space and Figure 3.3b) details the method to condition the bottom latent on the top latent using a stack of residual blocks with gated linear units. Each white block represents a stack of residual layers. This allowed for the easy control of the capacity of the model in the hyper-parameter search.

Along with introducing Wasserstein loss and progressively growing the generator and discriminator/critic, there were a few other important features which we implemented which resulted in better generated latents. The first was to condition the bottom PGAN at every structural level, as can be seen in 3.3b)[4]. One can hypothesise that the reason for this high level of conditioning being beneficial is due to the high degree of linkage between the two latents. As the VQ-VAE-2 decodes the two latent tiers concatenated together, if the two latents are modelling slightly different data modalities (even within the same class) this can scramble the output as the decoded image is now a combination of concepts which do not appear together in the true distribution.

It was hypothesised that the dependencies in the latent space were even more complex and long range than in the pixel space. To address this, inspired by (Zhang et al., 2018; Vaswani et al., 2017), a self attention layer was added to both the top and bottom generators and discriminators in the layer corresponding to 32x32 latents[5]. As in (Zhang et al., 2018), we implemented a residual connection around the self attention layer. The output of the self attention layer is first multiplied by a trainable scaling factor γ originally set to 0. This allows for the model to (or not) gradually learn to

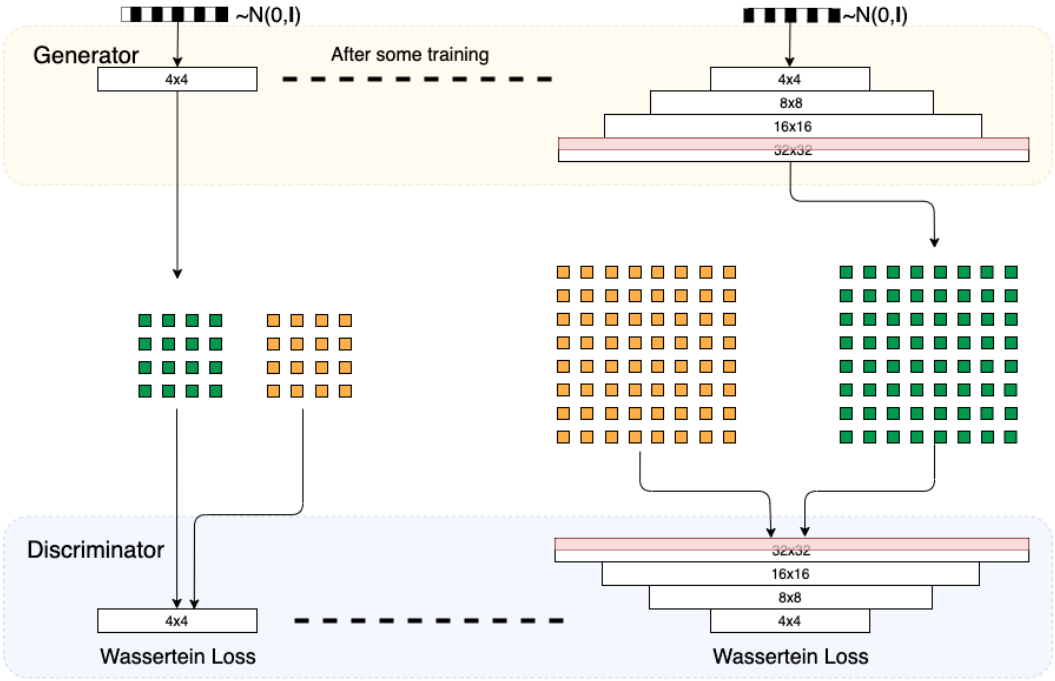
[1] The discriminator correctly classified every generated sample as fake, resulting in useless gradient signals for the generator to learn by. This effect is detailed in section 2.5

[2] As in Karras et al. (2017) we use the improved Wasserstein loss, where a gradient penalty is used to enforce the 1-Lipschitz constraint on the discriminator gradients rather than clipping the weights (Gulrajani et al., 2017).

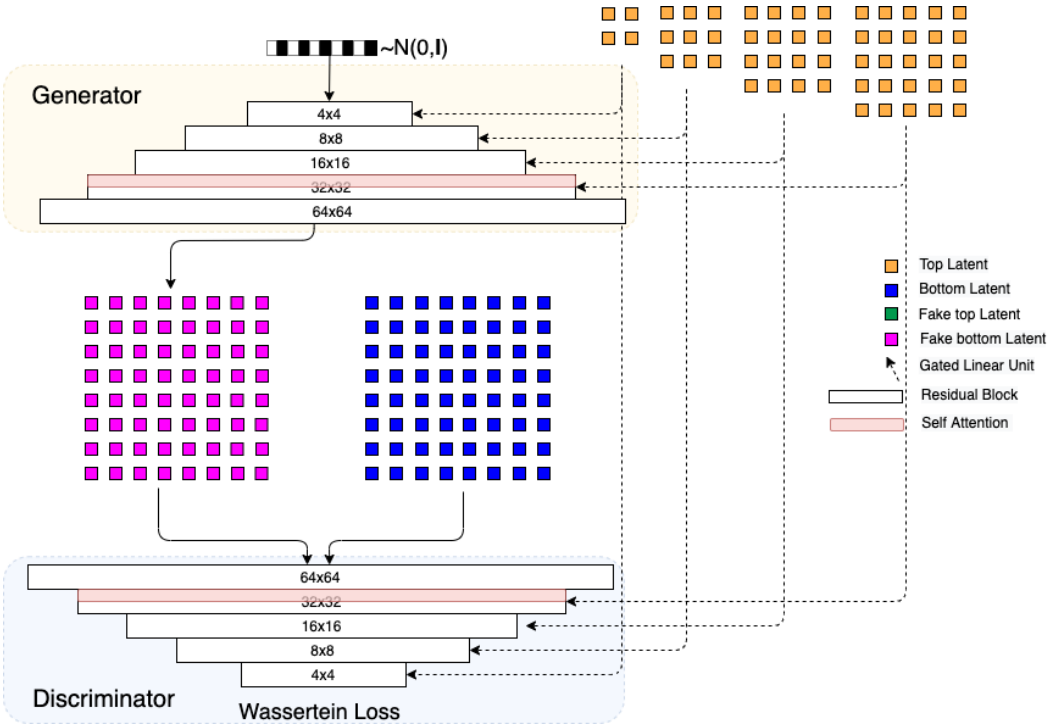
[3] As detailed in section 4.7, despite significant efforts, I was unable to fit the modified PGAN to the latent space of the vanilla VQ-VAE-2. It was only after adding the quantisation noise to the VQ-VAE-2 that the PGAN was able to model the distribution

[4] Taking inspiration from the repeated adaptive instance normalization used in STYLE-GAN where the model is conditioned on the sampled noise vector at every layer (Karras et al., 2018)

[5] Zhang et al. (2018) found that, taking into account the memory constraints of self attention being exponential with input length, that applying it to just the 32x32 resolution layer yielded the optimal trade off between performance and memory requirements



((a)) Top Progressive GAN prior featuring the training procedure



((b)) Bottom conditional Progressive GAN

Figure 3.3: Schematic detailing the final structure of the GANs which were fitted to the hierarchical discrete latent space generated by the VQ-VAE-2. This schematic depicts a batch size of 1. a) details the progressive growing nature of the GAN where as b) omits this to save space on the page.

include the information from the self attention layer. It is important to note that in all experiments, judging by the final value of γ after training, the model heavily relied on the self attention layer. Figure 3.4 details the structure of the self attention layer.

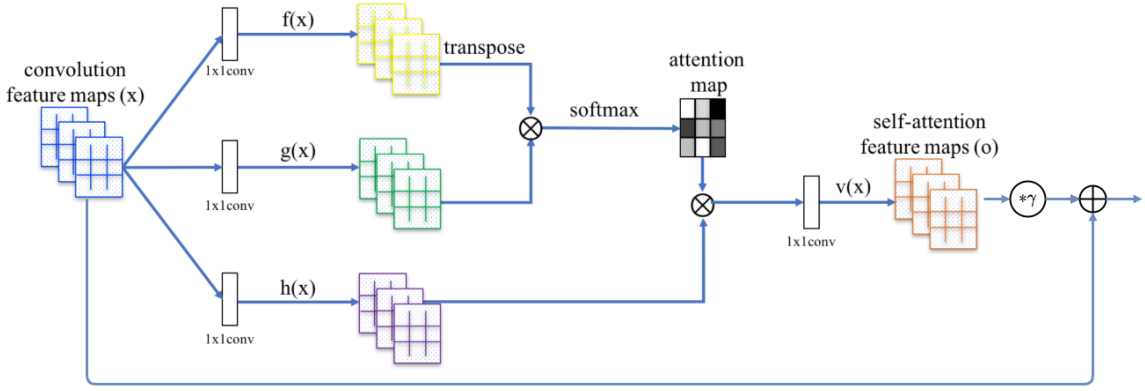


Figure 3.4: Schematic detailing the self attention layer which was implemented in the PGAN at the layer corresponding to a latent resolution of 32x32. Here we can see the residual connection around the self attention layer, with the learnable parameter γ, which allows the model to control the extent of influence has attention layer has. γ as in (Zhang et al., 2018) is set to 0 at the start of training, allowing the model to slowly incorporate attention. The figure was taken and adapted from (Zhang et al., 2018)

3.3 NATURALIMAGENET

For reasons detailed in Section 4.4 a dataset with a relatively small amount of classes was required. While CIFAR and MNIST provide exactly this, both comprise sets of inherently poor resolution images, 28x28 and 32x32 respectively. Therefore, only minimal benefits could be gained from compressing them into a latent space, especially since the bottom tier latent space in the VQ-VAE-2 is 64x64. Additionally, from a personal point of view, using the CIFAR and MNIST dataset is not satisfying since the MNIST images are simply hand written figures and the CIFAR images are so granular that they barely resemble their classes. To progress to a more complex dataset with classes, one might turn to ImageNet. However, the draw backs with using ImageNet are:

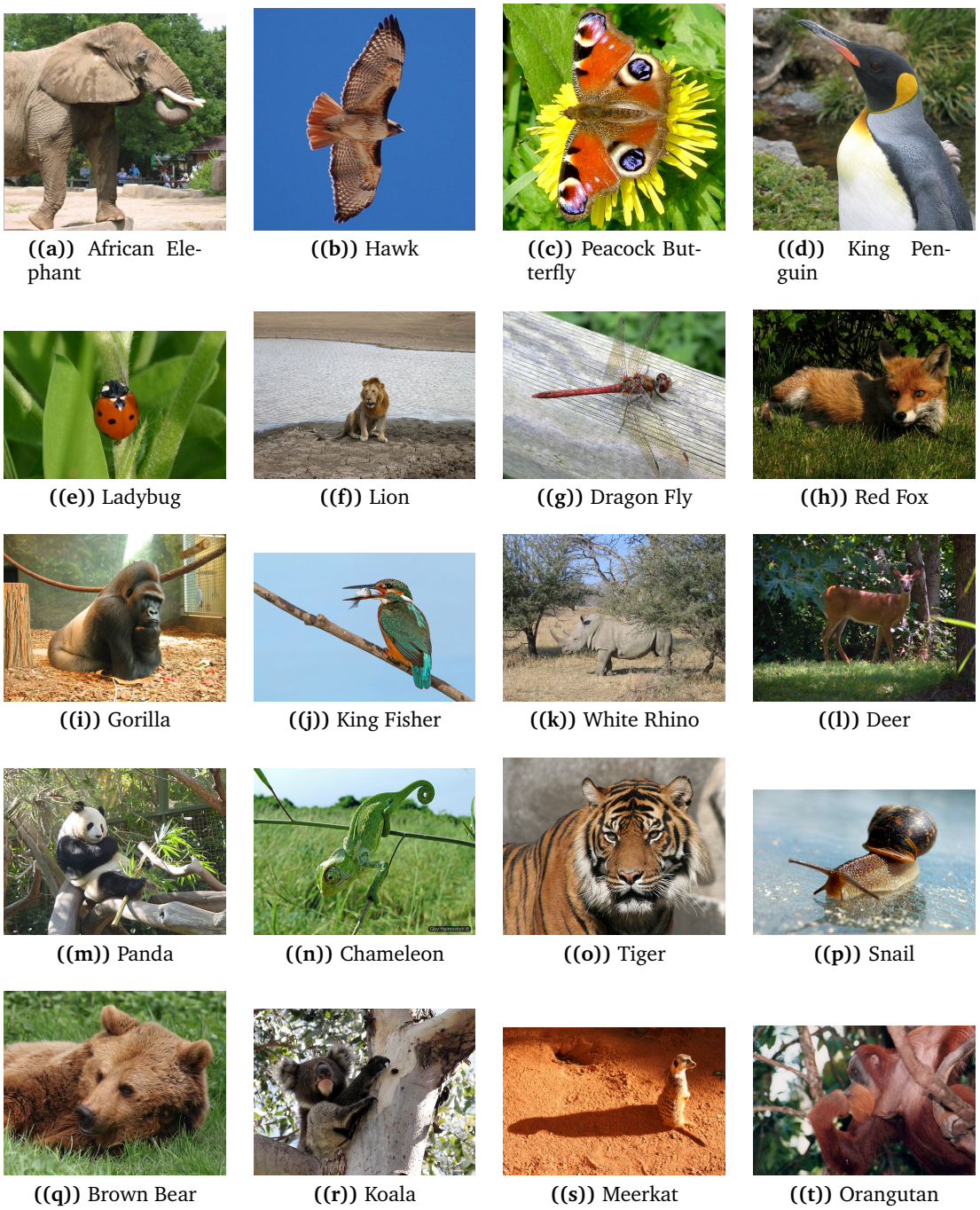
1. There are 14 million images in the ImageNet database, comprising 1.5TiB which would be extremely cumbersome for running quick analysis experiments and would put considerable strain on the compute resources available.
2. The dataset is made up of 20,000 classes, with some classes comprising fewer than 100 images. This yields a very hard classification problem and prevents the visual analysis, e.g. cluster analysis, of the learnt classifications.
3. Some of the classes are obscure, for example ‘commuter’, ‘great-nephew’ and ‘masterpiece’, resulting in a classification that humans would be unlikely to classify correctly.

The absence of an available dataset of high resolution images with an intermediate number of classes is an issue for the education of Machine Learning classification since this requires a more complex and rewarding image classification dataset without the complication of a large amount of compute power for practical classification and generative tasks. It was therefore deemed a useful allocation of time to create a new subset of ImageNet that would match the needs of Machine Learning classification training. Keeping with the theme of *Natural* images (to make the practice of classification more rewarding), 20 classes were manually selected, each with more than 2000 data instances (roughly the maximum number of data instances per class in ImageNet) along with a preference for *interesting* and *fun* classes. The final selected classes are detailed in Equation 3.1 and a sample from each class

is shown in Figure 3.5.

['African Elephant', 'Brown Bear', 'Chameleon', 'Dragonfly','Kingfisher', 'Giant Panda', 'Gorilla', 'Hawk', 'King Penguin', 'Koala', 'Ladybug', 'Lion', 'Meerkat', 'Orangutan', 'Peacock', 'Red Fox', 'Snail', 'Tiger', 'Deer', 'White Rhino']

(3.1)



((a)) African Elephant **((b))** Hawk **((c))** Peacock Butterfly **((d))** King Penguin

((e)) Ladybug **((f))** Lion **((g))** Dragon Fly **((h))** Red Fox

((i)) Gorilla **((j))** King Fisher **((k))** White Rhino **((l))** Deer

((m)) Panda **((n))** Chameleon **((o))** Tiger **((p))** Snail

((q)) Brown Bear **((r))** Koala **((s))** Meerkat **((t))** Orangutan

Figure 3.5: 4x5 grid of images representing the subset of ImageNet, NaturalImageNet. Each image is a sample from each of the 20 classes.

Chapter 4

Experimental Results and Discussion

4.1 Brittleness of latent space

When forming a mental model of the world it is important that the representations which form this model are robust. In that all the areas of the mental model have some interpretable meaning. This allows for easy interaction with the mental model, allowing for easy interpretation of the world.

We found that the latent space created by the vanilla VQ-VAE was extremely brittle to noise, in that a slight random perturbation of a data instance representation in latent space could completely scramble the image once decoded. This effect is illustrated in Figure 4.1. In addition to highlighting the fragility of the VQ-VAE latent space, this observation also gives us an insight into the condition of the data manifold within the latent space. We saw that any slight deviation from a data instance in the latent space resulted in a new point that would no longer lie on the manifold. This suggests that the manifold still occupies a very small volume in the latent space. This contrasts with the latent space of a β-VAE where the manifold densely occupies the prior. Instead the VQ-VAE manifold is thinly drawn out over the latent space, much like some high dimensional spaghetti. Next we repeated the experiment to the VQ-VAE2 latent space, but this time the VQ-VAE-2 was trained with quantisation noise (See section 3.1), this time, as shown in Figure 4.1 the perturbation had little effect on the quality of the decoded images. While this doesn't help us conclude anything about the structure of the data manifold, it does show that the representations are relatively more robust, and that the manifold is somewhat wider compared to the vanilla VQ-VAE.

((a)) **((b))** **((c))**

Figure 4.1: Investigation into the brittleness of the latent space of VQ-VAE. **a)** Unperturbed decoded image from the CelebA dataset **b)** decoded image after the addition of a small perturbation to the latent space of the Vanilla VQ-VAE **c)** A repeat of experiment b) but with the latent space created with a VQ-VAE with quantisation noise injected ($\sigma = 1.0$)

4.2 Learned codebook structure

As detailed in section 2.4 the VQ-VAE contains a codebook with 512 codebook vectors for each tiered latent space. Encoder vector outputs are then assigned to each of these codebook vectors according to K-means clustering and it is then a combination of codebook vectors which is passed to the decoder. Initially the codebook vectors are randomly initiated in space, but are then iteratively updated to equal the exponential moving average of the encoder output vectors which were assigned to them (Please see equation 2.13). As training progresses and the model learns to reconstruct input data to a higher likelihood, the codebook vector structure will change into a configuration which better represents the data. As it is the codebook vectors which are finally passed to the decoder, these codebook vectors represent the discrete data concepts which when combined in some pattern accurately capture the main features of the data instance. Investigating the learnt codebook structures is therefore an important practice to gain an insight into what type of representations are being formed.

Much like in natural language processing, the codebook vectors' position in high dimensional space encodes some form of a high level concept (Bengio et al., 2000; Rumelhart et al., 1986) that the autoencoder has learnt. Their position relative to one another can be inferred as some form of semantic similarity or difference. To investigate the structure of the high-dimensional codebook that the VQ-VAE has learnt, we used Principle Component Analysis, PCA, to perform dimensionality reduction and to project the codebook vectors onto a 2D plane. Figure 4.2 details the PCA projections of the bottom tier (left) and top tier (right) codebooks with an increasing amount of injected quantisation noise. Plots a) and b) represent the learned codebooks when no noise is injected. Looking at these plots we can see that for the standard VQ-VAE-2 codebook projections, a codebook's index has no correlation with its position in the embedding space since the different indices (represented by different coloured points in the PCA diagrams) are jumbled together in the latent space. Therefore we can infer that there is no semantic structure for the codebook vectors, in that a codebook's index has no correlation with its position in the embedding space, and so no correlation to high level meaning. In addition, for the top codebook, a significant proportion of codebook vectors have a value of 0, as a result of not being utilised. This is evidence of *codebook collapse* a phenomenon detailed in section 2.4.6. This happens because if a codebook vector is not currently being used, in this VQ-VAE formulation, its value will never be updated to a space where encoder outputs are likely to be mapped to it, and so no encoder output will be encouraged to *commit* to it. This is a problem as detailed in section 2.4.6 that limits the VQ-VAE's expressability resulting in a sub-optimal, local minimum.

Figures, 4.2 c) and d) demonstrate the effect of the injection of a small amount of noise ($\sigma = 0.2$) into the assignment stage in VQ, a process detailed in Section 3.1. It can be seen that two improvements to the codebooks occur. Firstly, although the global structure remains roughly the same (i.e. the distribution of codebook vectors remains relatively uniform), a gradient of colour and so gradient of index is observed. This means that now the codebook has learnt a structure where codebook vectors that are close to each other in ordering have a similar effect to each other once decoded (in other words: adjacent vectors, in codebook ordering, are more likely to have a similar meaning to each other). This simplifies the job for the prior in that if its output is slightly off (e.g. codebook index 203 is selected over 202) this will result in only a minor change to the output, instead of scrambling the decoded output as it would have done without the injected noise. As the prior operates in the discrete latent space of codebook indices (van den Oord et al., 2017; Razavi et al., 2019; Dhariwal et al., 2020) this process can be viewed as increasing the level of disentanglement of the manifold in the latent space. Within this latent space, with the addition of quantisation noise, concepts linearly change, rather than being completely random as they had been when the job of determining the structure or *language* of the codebook was left entirely to the prior without noise injection. The second benefit is that *codebook collapse* has been prevented. This can be seen in Figure 4.2 c) and d) where all the codebooks have been assigned to a value. The explanation behind this effect is that with the injected noise, there is a chance that an encoder vector's assignment will be nosily assigned to a non intented codebook vector. This noisy assignment may point towards a codebook vector that is not currently being used. This provides a foothold, or signal by which the codebook vector can change and adapt to the encoder outputs and as a result be used in future batches.(Please see

Section 2.4.6 for a full explanation).

If more noise is added ($\sigma = 1.5$), as seen in Figures 4.2 e) and d) a complex structure is formed in both Top and Bottom latent spaces. Some form of semantic ordering has been constructed, with the bottom codebook forming a type of tertiary structure. We hypothesise that the addition of the constraint of vector quantisation noise has forced the model to learn a complex codebook structure, one which might represent the higher level concepts of the data. The folding structure observed in 4.2e) is particularly interesting and it can be speculated that the reason for this structure is that as the representations start to differ with the codebook ordering, there is a re-surge of similarity, detailed in the fold, before then again becoming more different. This is something which can subjectively be seen in high level concepts. An example of this could be human hair, very loosely, female blonde hair is similar in many ways to male blonde hair but it is also similar to female brunette hair, female brunette hair is very different from male blonde hair. This then might result in a manifold where different colours of female hair are arranged adjacent to each other consecutively along the manifold which is also folded back on itself so that the points in space corresponding to male and female blonde hair can also be close to each other in some fashion. This can be likened to the tertiary structure of proteins which, given the sequence of amino acids, and the manner in which they repel and attract, form complex folds and twists which allow them to undertake their task.

4.3 LATENT INTERPOLATION

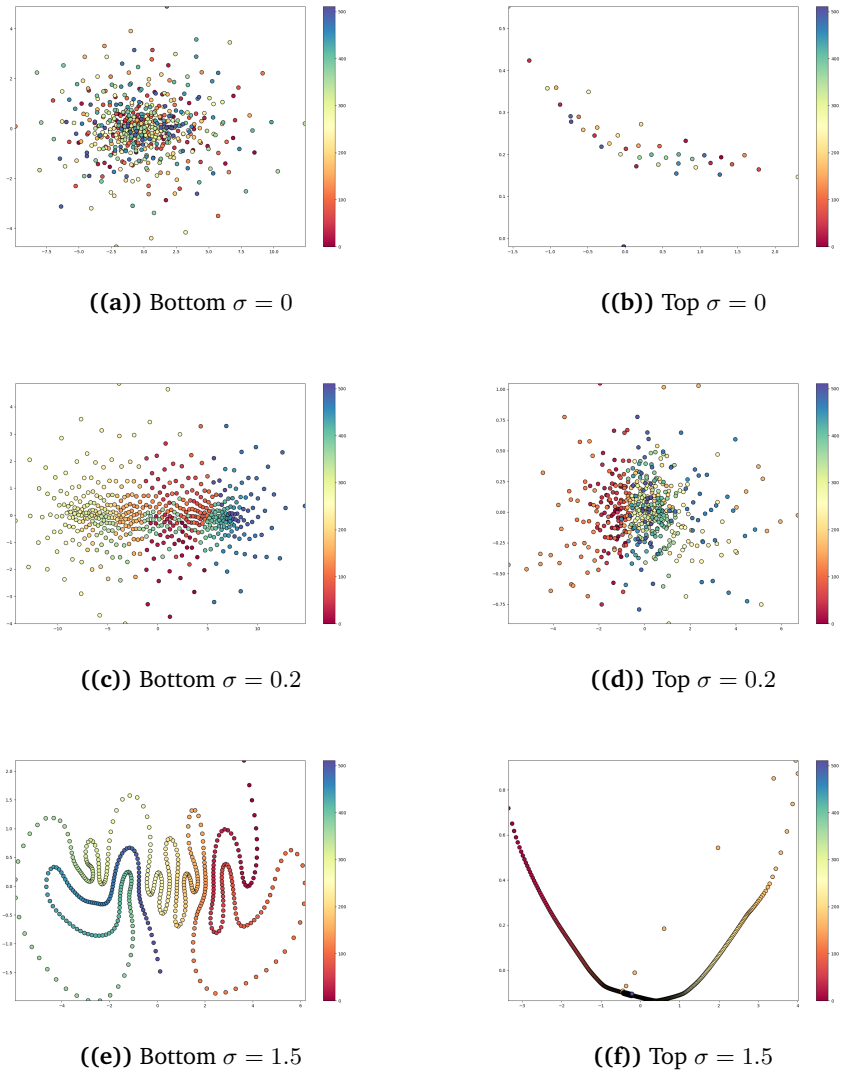
Latent Interpolation is defined as the ability of a model to mix the latent codes of two datapoints, resulting in a point in latent space lying between the two datapoints, which once decoded, mixes together the semantic features from the two datapoints in a meaningful and plausible manner (Berthelot et al., 2018). The ability for a latent model to interpolate well is important for a number of reasons:

1. From a geometric point of view it shows that the manifold has been disentangled, and represented in a linear manner (Bengio, 2019). That the data manifold is now a convex set.[1] This geometric interpretation is detailed schematically in Figure 4.3.
2. It shows that the model has learnt the key concepts of the dataset, or "structure" as it is referred to in Berthelot et al. (2018). Rather than learn a simple mapping from each datapoint to its unique position in the latent space and back, the model has learnt a reverse mapping that can generalise well.
3. When forming a meaningful model of the world it is important to allow easy interaction with the key concepts of the world. Latent interpolation is the easiest possible way to allow for the exploration of these complex concepts.

We found that running the vanilla VQ-VAE produced extremely poor interpolation results Figure 4.4 a) details an example interpolation between 2 datapoints. This is understandable, as the codebook has no semantic structure with respect to the codebook indices therefore, moving linearly through this space will not yield plausible results.[2] Adding a small amount of noise ($\sigma = 0.5$) to the VQ assignment yielded a marked improvement to the interpolation results, detailed in Figure 4.4 b) and c). This is thanks to the added semantic structure to the codebook, detailed in 4.2 and may be also due to the constraint on the latent space forcing the model to extract and represent higher level concepts to a greater extent. Analysing the interpolations closer however, suggests that these interpolations might not be gradually changing concepts and that instead they appear to be resembling what an interpolation in data space may appear like, i.e. the gradual superposition of the final image onto the other. This needs to be investigated further and is left for future work.

[1] Let S be an affine space, and $C \subseteq S$. C is a convex set if $\forall x \forall y \in C$ all points on the linear interpolation between x and y are in C (Carla C. Morris, 2015)

[2] However, interpolating in the continuous space (i.e. before VQ) also demonstrated VQ-VAE's poor interpolability, with Berthelot et al. (2018) liking the model to a vanilla autoencoder.



((a)) Bottom $\sigma = 0$ **((b))** Top $\sigma = 0$

((c)) Bottom $\sigma = 0.2$ **((d))** Top $\sigma = 0.2$

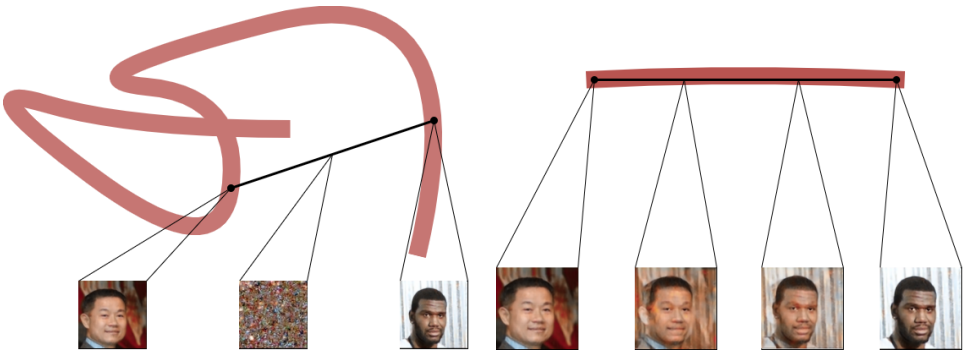
((e)) Bottom $\sigma = 1.5$ **((f))** Top $\sigma = 1.5$

Figure 4.2: PCA projections of the learned 512 codebook vectors with varying amounts of noise after 50 epochs trained on the CelebA dataset (Liu et al., 2015). The left column represents the bottom codebook and the right column represents the top codebook. Plots a) and b) are for 0 noise (the standard VQ-VAE), c) and d) are for a small amount of injected noise ($\sigma = 0.2$), and e) and f) represent a large amount of injected noise ($\sigma = 1.5$)

4.4 Analysis on downstream tasks

Analysing the performance of a simple classifier trained on the latent space created by an autoencoder is a common method to determine the quality of the extracted representations (Coates et al., 2011; Berthelot et al., 2018)[3]. The reason this is an effective method of classifier assessment is that a good representation disentangles concepts and presents them in an easier to interpret (and so easier to classify) manner. We trained a standard VQ-VAE-2 model on our custom subset of ImageNet, NaturalImageNet (detailed in Section 3.3), with a varying amount of quantisation noise (method detailed in Section 3.1). A simple classifier was trained on the latent space with 5-fold cross-validation. It is important to note that the classifier was not tuned and that the initial

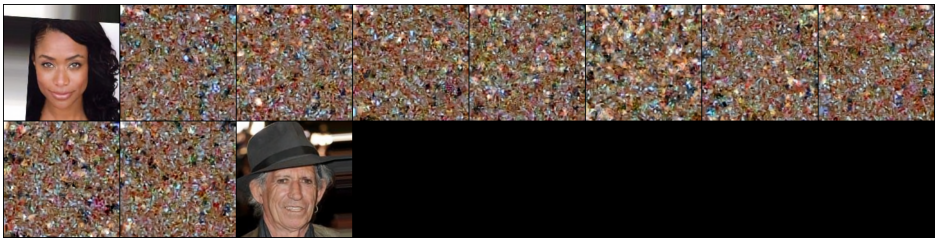
[3]A good representation was defined by Ian Goodfellow, Aaron Courville and Yoshua Bengio in their book Deep Learning to be "one that makes a subsequent learning task easier" (Goodfellow et al., 2016)



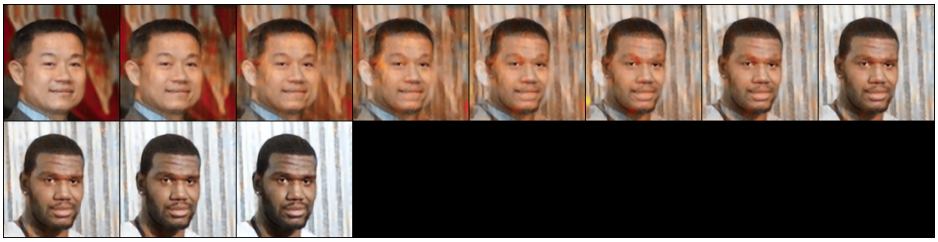
((a)) Interpolation with an entangled manifold

((b)) Interpolation when the manifold is laid out in a linear manner

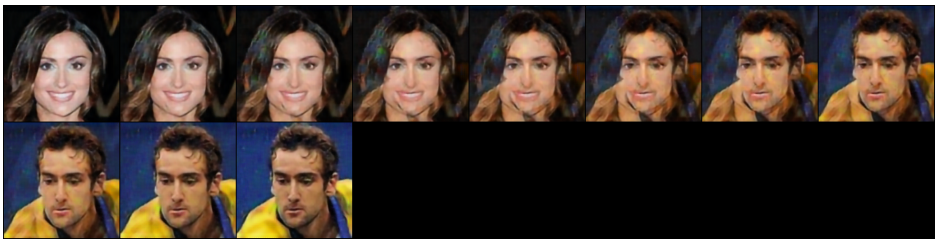
Figure 4.3: Schematic detailing the link between latent interpolation and the level of disentanglement in the latent space. Note that the maroon thick line represents the data manifold. **a)** details how when the manifold remains entangled, linearly interpolating between two datapoints, in this case faces from the CelebA dataset, yields meaningless decoded outputs due to the linear interpolation not lying on the manifold. **b)** details when the manifold has been disentangled and exists in a linear form, linear interpolation between datapoints yields semantically meaningful results. The interpolation results have been taken from a) a vanilla VQ-VAE and b) a VQ-VAE with $\sigma = 0.5$



((a))



((b))

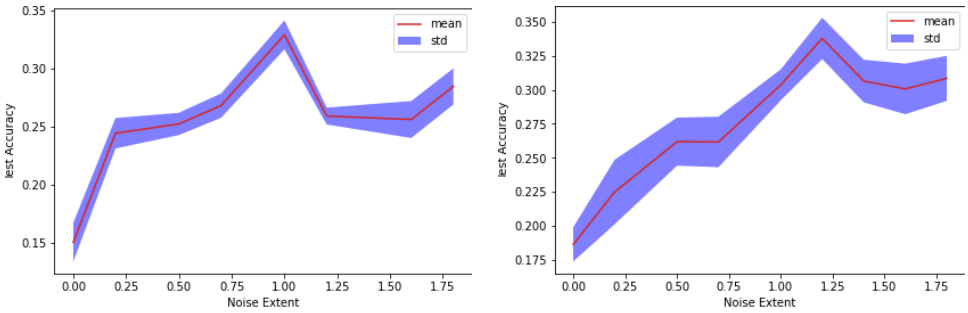


((c))

Figure 4.4: Interpolation results with the CelebA dataset. **a)** standard VQ-VAE-2 with no noise injected **b)** and **c)** with noise injected ($\sigma = 0.5$

)

hyper-parameters were used because this study was performed purely for the comparison between different VQ-VAE formulations and for nothing else.



((a)) Top Latent Space **((b))** Bottom Latent Space

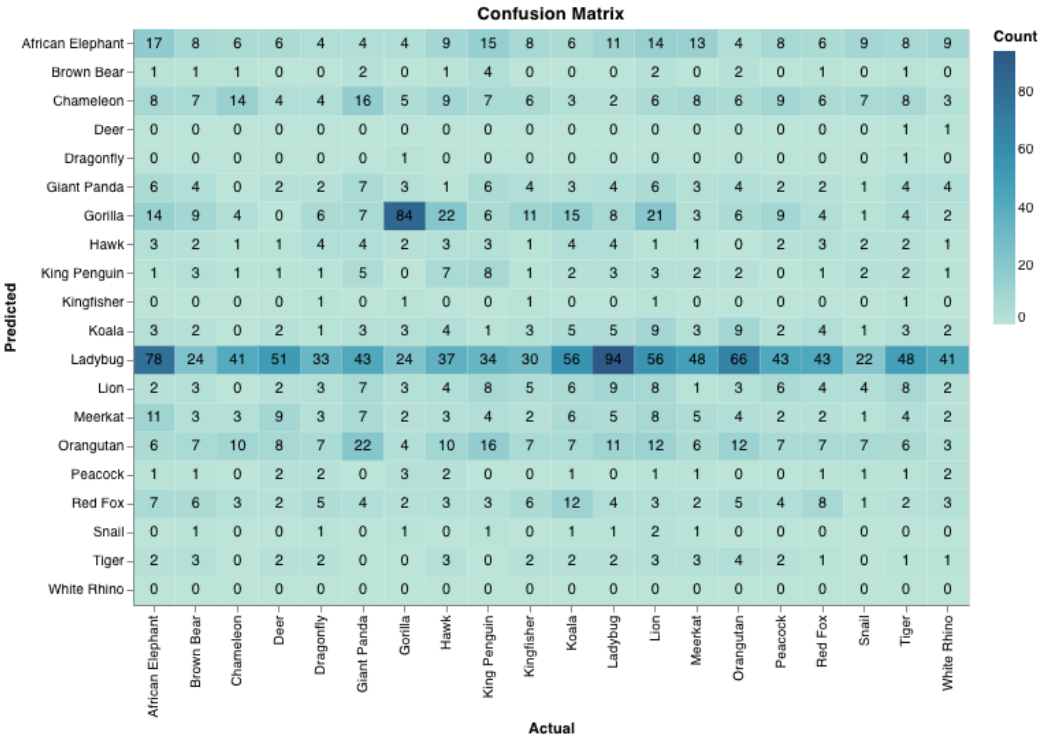
Figure 4.5: Plots of the test accuracy after a simple model was trained with 5 fold cross validation to classify the classes of a latent representation of NaturalImagNet generated by a VQ-VAE-2 model with increasing amounts on noise (please see Section 3.1 for implementation of noise.). **a)** is when the classifier was fitted to the top latent tier of the VQ-VAE-2 and **b** is for the bottom tier.

Figure 4.5 details the effect that adding noise to the quantisation step during training has on the quality of the latent space. There is a clear trend, which shows improvement of classification accuracy as the addition of noise during training is increased. Classification accuracy improved from $\sim$15% to $\sim$35% test accuracy as sigma transitioned from $\sigma = 0$ (vanilla VQ-VAE-2) to $\sigma = 1.0$. An important point is that when no noise was added, i.e. the standard VQ-VAE-2, the classification accuracy was extremely poor: the classifier simply predicted most of the data instances as a "Ladybug", the largest class in the dataset. This is illustrated by the confusion matrix in Figure 4.6a) and suggests that the quality of the representations extracted by the standard VQ-VAE-2 are poor with the resulting classifier's predictions being little or no better than random. The confusion matrix of 4.6 b), however, illustrates the beneficial effect on classifier prediction of adding noise ($\sigma = 1$). Figure 4.6 b) contrasts with a) in that the highest counts now clearly correlate to correct predictions made by the classifier. Interestingly, this beneficial effect of noise addition appears to plateau at $\sigma = 1.0$. The mechanisms behind this beneficial effect on the classifier and the shape of the test accuracy plots have not been elucidated. One possible explanation might be that the injection of small amounts of noise enforces some form of semantic structure, but increasing the amount of noise injected above the plateau fails to yield another tier of higher level structuring. However, this will need to be investigated further and is left for future work. The results of the current experiment, combined with the effects noted on the reconstructions detailed in Section 4.5, suggest that a $\sigma = 1.0$ can be considered as optimal for this model.

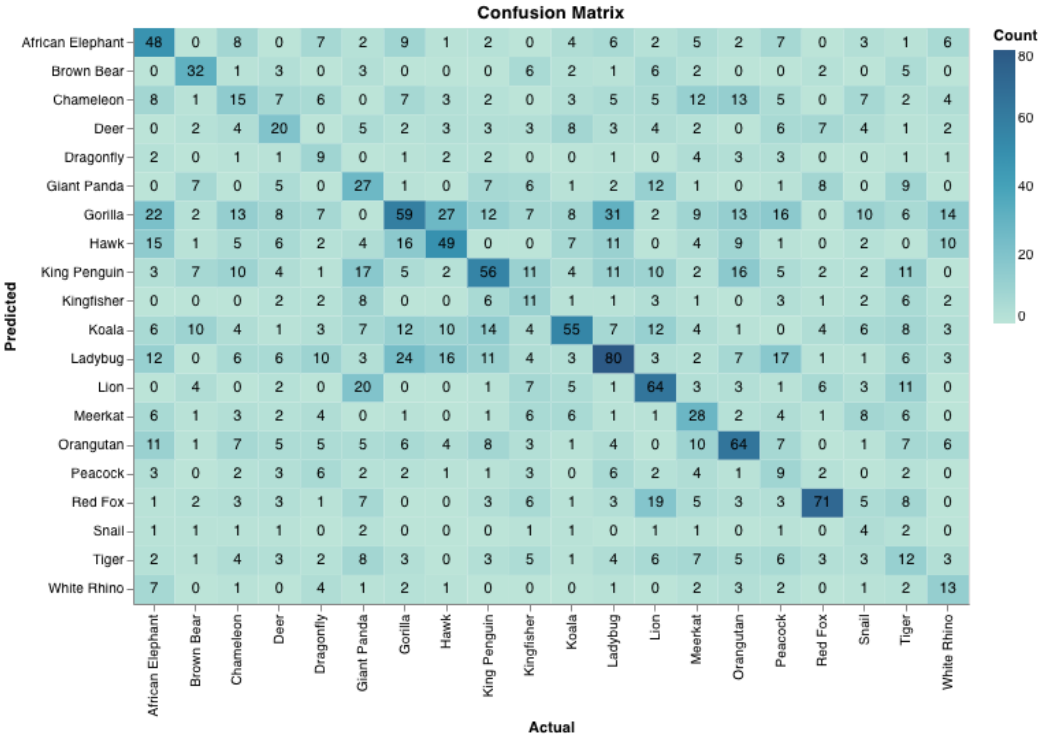
4.5 Effect of Injected noise on Reconstructed Image quality

As with VAEs, injecting noise reduces the quality of the reconstructed data[4]. It is important to analyse the extent to which the quantisation noise affects the quality of the output images. Figure 4.7 details the effect that an increasing amount of noise has on the reconstruction loss. Due to a constraint on compute resources, repeat experiments were not a feasible option. As a result, the trend is extremely sporadic, however, a slight increase in reconstruction loss can be seen for both test and train cases. It also may be inferred that with the addition of a small amount of noise (up to $\sigma = 1.0$) little increase

[4]With VAEs as detail in Section 2.3. The injected noise is a result of the reparamaterisation trick, simulating sampling from the approximated posterior, a simple Gaussian distribution. Sampling from the posterior, as opposed to a deterministic procedure, results in reduced quality reconstructions as the decoder has to infer the data modality of the posterior. The larger the variance of the posterior (or more added noise through the reparamaterisation trick) the more the posterior distributions overlap and the more likely the decoder can infer the wrong concept and so reconstruct something different from the input.



((a)) Standard VQ-VAE-2 (Classification on top latent)



((b)) VQ-VAE-2 with $\sigma = 1$(Classification on top latent)

Figure 4.6: Example confusion matrices for latent classification using simple classifier.

in reconstruction loss is observed, whereas increasing the noise beyond this point appears associated with increase in the reconstruction losses (Train and Test) to a greater extent. This inference is backed up by analysing the reconstructed images directly. These are detailed in Figure 4.8 where on close inspection, the reconstructed images from the model with $\sigma = 1.0$, appear to be of a similar quality as the input images whereas the reconstructed images from the model with $\sigma = 1.8$ are more blurry. This is more apparent when the images are viewed at their full resolution. It is important to note, that the quantisation noise is only applied during training, and at test time no quantisation re-assignment takes place.

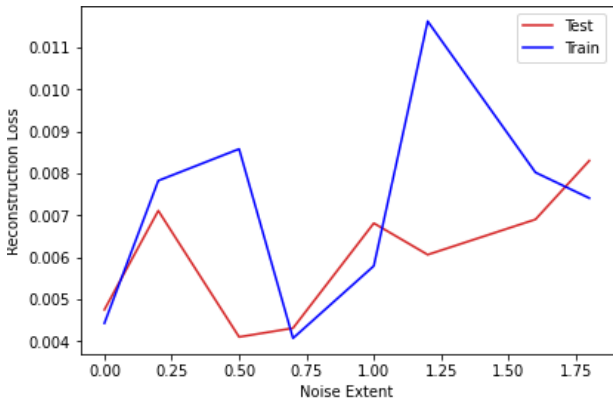


Figure 4.7: Plot detailing the reconstruction loss (test and train) for VQ-VAE-2 with a varying amounts of injected quantisation noise, σ. It is important to note that this noise is not injected at test time (quantisation noise only injected during training).

4.6 QUALITY OF GENERATED IMAGES

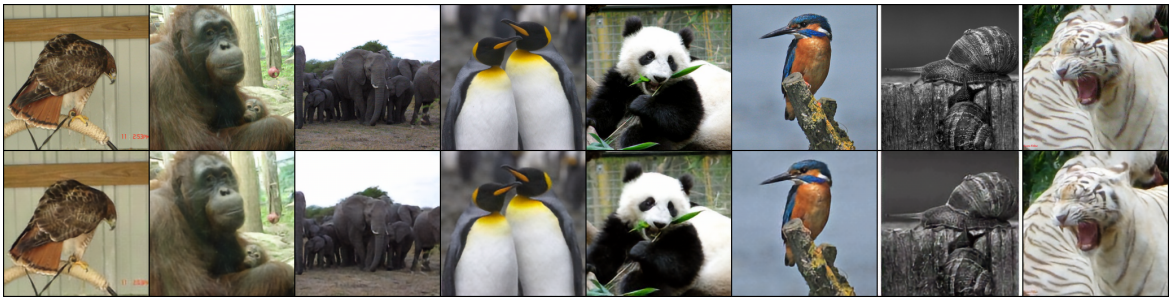
The quality of generated images of both the standard VQ-VAE-2 and a VQ-VAE-2 with injected noise of $\sigma = 0.5$ were analysed. 2 sets of top and bottom autoregressive PixelCNN priors were fitted to the standard and noise injected latent spaces for the FFHQ dataset and trained for a total of 32 training days[5]. Due to compute constraints this experiment was not repeated or extended to other datasets. Inspecting Figure 4.9 qualitatively it appears that injecting the noise to the VQ-VAE-2 has not yielded a significant increase in generated image quality, suggesting that unfortunately the quantisation noise did not make the autoregressive model's job of fitting the latent space any easier. This may be attributed to autoregressive models being very good at modelling unordered discrete distributions, as is the case for language models (Please see section 2.4.2).

4.7 GAN PRIOR RESULTS

This section details our attempts to fit a GAN as the prior to the latent space, i.e replace the PixelSNAIL. Please see Section 3.2 for a description of the implementation. Despite considerable efforts, no implemented formulation of the PGAN, trained on the latent space generated by the standard VQ-VAE-2, yielded even remotely convincing generated images. All generated latents once decoded resembled white noise, an example of such an image is shown in Figure. 4.10a). However, in contrast, when the same PGAN model was trained on the top and bottom latent space of the VQ-VAE-2 with quantisation noise ($\sigma = 0.5$), the images, while still of poor quality when compared to the state of the art or when an autoregressive PixelSnail is fitted to the latent space, are markedly improved. Figure 4.10 details a sample of these generated images. This suggests that adding quantisation noise during training yields a representation of the data which is easier for a GAN to model. Time restrictions on this project prevented further investigation, for example of different

[5]Each of the 4 PixelCNN models was trained for 8 days on a Nvidia TITAN XP 12GB

((a)) $\sigma = 0$ (vanilla VQ-VAE-2)



((b)) $\sigma = 1.0$

((c)) $\sigma = 1.8$

Figure 4.8: Three figures detailing the input image (top row) and reconstructed image (bottom row) for VQ-VAE-2 models which were trained with varying amounts of noise after a compression of 35x. All models were trained for 50 epochs. The reconstructions were from a test set (which was not trained from) from the NaturalImageNet (please see section 3.3).

levels of sigma and this is left for future work. Nevertheless, this work demonstrates proof of concept for the use of a GAN in this context. Using a GAN as the prior also yielded a reduction in training time of 75% (8 days per Pixel prior tier to train compared to 2 days per PGAN prior tier to train) and dramatically increased the speed at which new samples could be generated. This is because the GAN produces a sample with one forward pass through the network where as the autoregressive models have to sequentially generate each individual latent/pixel conditioned on all previous pixels, an extremely slow process.

4.8 LATENT SPACE VISUAL ANALYSIS

A side effect of forcing semantic continuity in the codebook is that visualisation of the latent codes conveys some description of the type of information which is being encoded. Therefore, if the codebook vectors are assigned a grey scale value, ranging from 0 to k (512 for all experiments in this project) they form an image which can be qualitatively analysed. Visual qualitative analysis of the

((a)) ((b))

Figure 4.9: Images sampled from a) VQ-VAE-2 with noise addition ($\sigma = 0.5$) b) Vanilla VQ-VAE-2 after identical PixelCNN models have been fitted.

generated latents enabled insights into the compression process. For example, if the latent images appeared very similar to the image that was being encoded, this clearly showed that the bottom latent was only encoding very local information. Figure 4.11 details the top and bottom latents for 3 data instances, for the vanilla VQ-VAE-2 (Figure 4.11a)) and a VQ-VAE-2 with $\sigma = 0.5$ of injected quantisation noise (Figure 4.11b)). Figure 4.11c) illustrates an additional issue with the VQ-VAE-2 set up that was highlighted for us when visualising the latent space in this manner. First detailed in (Dhariwal et al., 2020), namely, if the constraint on the top latent is too severe, the VQ-VAE-2 might completely bypass the top latent and would not be forced to learn the higher level representations required to utilise the more constrictive bottleneck. This effect can be seen in Figure 4.11c), where the top latent transfers little to no information, and all the information is passed through the bottom latent. This is a problem as the model no longer learns the high level representations. During this study, with limited time to re-run experiments, this effect occurred intermittently and no pattern or explanation for its occurrence was discovered apart from that the the tighter the constraint on the top latent the more likely it was to happen. Further investigation of this phenomenon is left for future work.

((a)) Vanilla VQ-VAE-2 ($\sigma = 0$)

((b)) VQ-VAE $\sigma = 0.5$

Figure 4.10: Generated images after fitting the experimental PGAN to the top and bottom latent space generated from a) the vanilla VQ-VAE-2 (please note that this image is representative of all the generated images) and b) a VQ-VAE-2 with the addition of $\sigma = 0.5$ to the quantisation assignment during training.

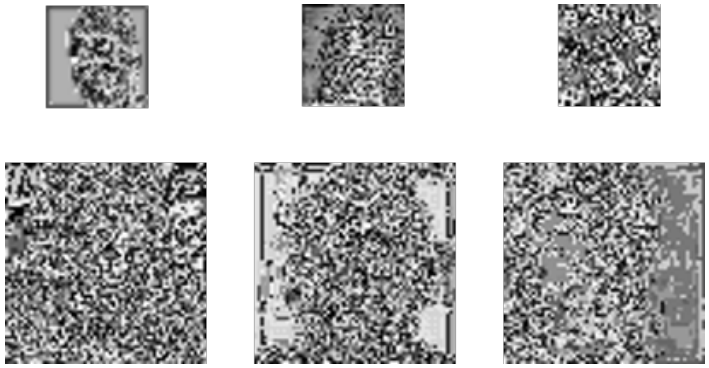
4.9 A REFLECTION ON VQ-VAES FOR REPRESENTATION LEARNING

4.9.1 SPACE CONSTRAINT

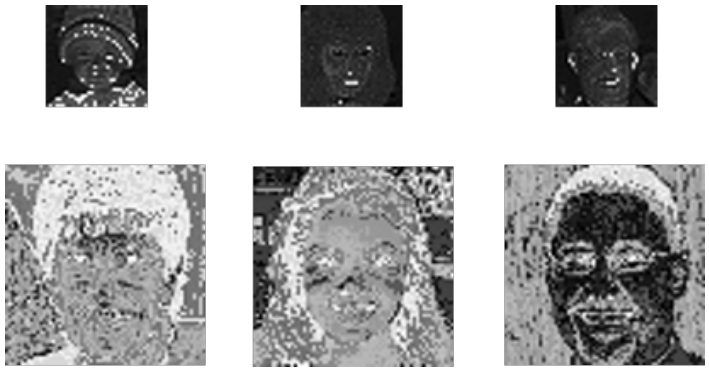
As detailed in section 2.3.3, imposing a bottleneck on the flow of data in the form of a constriction on the volume of space in which the information can be represented is an effective way of extracting meaningful features and also encourages *smooth* representations. To this end the constraint imposed by the VQ-VAE with and without quantisation is analysed.

Thanks to the $D_{KL}\left(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P(\mathbf{z})\right)$ term, VAE latent spaces are constrained to a given volume, dictated by the prior. This acts as a constraint yielding useful representations (see Section 2.3.3 for a detailed explanation). VQ-VAE latent spaces are however not so tightly constrained. Take a codebook, with vectors that occupy a finite volume V, thanks to the *commitment* loss, encoder outputs are encouraged to move towards the codebook vector to which they were assigned. The codebook can therefore be viewed as constraining the encoder output space. There is, however, an issue, that the codebook vectors are themselves updated towards the encoder outputs with every training iteration, through the EMA update, detailed in Section 2.4.4 and hence the codebook space is not static (in contrast to the prior in VAEs). A consequence of this is that as the VQ-VAE is trained with the objective to minimise the reconstruction loss, it favours a more expressive and so less regularised space. This corresponds to increasing the encoder magnitudes outside the volume V. If encoder vectors lie outside of V then at the next EMA update the codebook vectors will move out and will thereby expand V. As the VQ-VAE is trained via iterative stochastic gradient descent techniques, a step wise expansion of the latent space could occur, reducing the level of regularisation enforced. Now while this is good for the reconstruction capabilities of the model, it is bad for the model's ability to extract meaningful representations, which relies on a restrictive bottle neck.

It can be hypothesised that the addition of quantisation noise restricts this expansion. The effect of quantisation noise is that discrete representations can be interchanged with some given probability, this favours the discrete concepts occupying a small volume of space, so that if a swap occurs, the resultant change in representation is small and so results in only a small effect on the reconstruction quality. This can be viewed as confining the bottleneck to a smaller space and therefore yielding the superior representations which have been demonstrated in this work.



((a)) $\sigma = 0$ (vanilla VQ-VAE-2)



((b)) $\sigma = 0.5$

((c)) An example of when the top latent was ignored/bypassed by the VQ-VAE

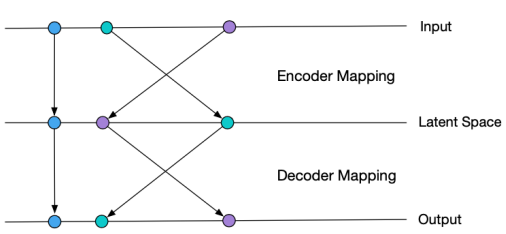
Figure 4.11: Visualisation of 3 data instance examples of the top (top row) and bottom (bottom row)latent representations generated from; **a)** vanilla VQ-VAE-2 **b)** VQ-VAE-2 with a small amount of quantisation noise and **c)** latents taken from a VQ-VAE-2 model which has completely avoided utilising the top latent.

4.9.2 Continuity Constraint

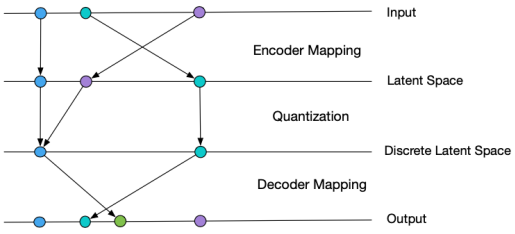
VAEs ensure similarity preservation in the latent space, i.e. points close in the data space are also close in the latent space, referred to in Section 2.3.3 as *continuity*, this is achieved by forcing the posterior to be squeezed into the prior. As depicted in 2.9 this forces posterior distributions for different data modalities to overlap, yielding an optimal latent space, where concepts near each other in the latent space are also near each other in the data space once decoded. Wu & Flierl (2020) argues that vector quantisation acts as a form of regularisation which favours *continuity*. This can be argued from a visual perspective, detailed in Figure 4.12. Assuming a very high capacity decoder, it

can be seen that a regular autoencoder with a standard bottleneck has no constraint to structure the latent space, and that an essentially random latent space can yield optimal reconstructions. This is because the decoder can infer the exact reconstruction from the latent (this point is argued in Section 2.2.2). With a VQ layer the encoder can only map to a set amount of latents. As a result, if the latent space is constructed in a random manner (as seen in Figure 4.12b) two inputs of very different modalities will be mapped to the same codebook vector. As the decoder is a deterministic function this means that the best the decoder can do in this circumstance is output a reconstruction which is an average in some form of the two very different data instances. Figure 4.12c) demonstrates a solution with a lower reconstruction loss where the latent space is constructed with *continuity* allowing the decoder to make better reconstruction predictions. Building on this point, in the same way with β-VAE the beta parameter controls the extent of posterior overlap and so quality of latent space/representations formed, it can be argued that increasing the number of codebook vectors, K, can be viewed as having the inverse effect of increasing β. The reason for this is that when K is low the data instances will have to share more discrete latent representations than when K is high. Therefore, an optimal solution would be one where *continuity* is enforced, and that the extent to which it is enforced is inversely proportional to the value of K. This can be seen from an information perspective where the value of K controls the number of bits transferred through the bottleneck, $\log_2(K)$ detailed further in Section 2.4.

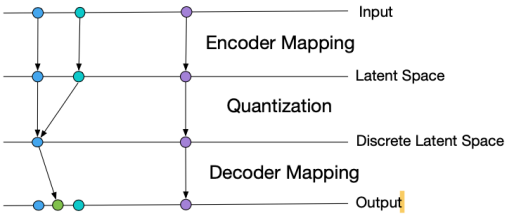
The addition of the quantisation noise enforces this constraint to a greater extent. Introducing quantisation noise during VQ results in there being a chance that a representation in the discrete quantised space being swapped to a neighbouring discrete representation. As detailed in section 4.8 an optimal latent space trained with quantisation noise will possess *continuity* (as swapping dissimilar concepts will have a more severe effect on reconstruction loss that swapping similar concepts). The quantisation noise, in a VAE setting, can be viewed as increasing the variance of the posteriors and so increasing the extent of which they overlap, therefore having a similar effect as increasing the value of β.



((a)) Standard Autoencoder



((b)) Non optimal VQ-VAE



((c)) Optimal VQ-VAE

Figure 4.12: Schematic detailing the similarity preservation mapping properties of vector quantisation. Assuming a very high capacity decoder, **a)** details how an autoencoder can be optimal, independent of the structure of the latent space **b)** demonstrates that if the latent space of a VQ-VAE is not constructed in a manner which preserves similarity, the decoder outputs will be sub-optimal **c)** depicts how constructing a latent space with similarity preservation results in optimal data likelihood. Figures taken from (Wu & Flierl, 2020)

Chapter 5

Conclusions and Future Work

We present two main novel contributions:

1. A new modification to the VQ-VAE training procedure which, through the addition of quantisation noise to the VQ-VAE, results in a soft vector assignment over the codebook vectors. This imposes a high-level structure on the codebook vectors of the VQ-VAE, with the consequences of yielding more interpretable and useful representations along with a latent space which possesses *continuity*. We have demonstrated these properties through a range of quantitative and qualitative experiments and our discussion covers potential causal factors for the observed improvements from a theoretical perspective which draws inspiration from VAE theory.

2. To the best of our knowledge we are the first to demonstrate that replacement of the powerful autoregressive PixelSNAIL prior with an alternative generative model, in this case a novel formulation of the Progressive Growing GAN, can successfully fit the VQ-VAE-2 hierachael latent space. This is a significant contribution to the field of VQ-VAEs as it demonstrates flexibility in the way these representations can be used. The PGAN prior, while currently producing data instances of reduced quality, compared to the PixelSNAIL, is markedly faster to train (75% reduction in computational time) and significantly faster to sample from.

This work makes progress towards extracting meaningful representations and forming mental models of high resolution and complex worlds. A disentangled mental model of the world will allow artificial agents to reason at a higher level and internally simulate and evaluate policies and theories through the process of imagination and dreaming. Providing AI with these capabilities is a necessary step if a higher level of cognition and generalisability of understanding is to be achieved.

There are numerous avenues for further work, in adapting VQ-VAEs to form mental models for artificial agents. The first is to replace the VAE in the world model architectures developed in (Ha & Schmidhuber, 2018; Hafner et al., 2019) with a VQ-VAE. This will directly highlight the problems which need to be overcome to enable VQ-VAEs to form useful world models. The following list enumerates aspects of VQ-VAEs where current inadequacies of function suggest potential future improvements and topics for future research:

1. **Problems extracting higher levels of representation**. As alluded to in section 2.4.5 and 4.8, to extract a very high level concept through autoencoder based methods, a very constrictive bottle neck is required. However, with the current VQ-VAE-2 formulation, increasing constriction of the top tier results in the model learning to ignore the top latent and instead to push all the information through the less constrictive lower tiers. By this process the model avoids having to learn the higher level representations. This was combated in (Dhariwal et al., 2020) through forming a separated VQ-VAE for each tier and then combining

the latent spaces. Preliminary studies conducted using this adapted approach have shown promising results with higher levels of compression being possible.

2. **Engineered Architecture**. A systematic study of neural network architecture tailored for the VQ-VAE needs to be conducted. Demonstrated in (Vahdat & Kautz, 2020), the benefits of fine tuning all aspects of the architecture, down to the type of activations and the order in which they are arranged, can result in significant improvements.
3. **Moving on from mean squared error?** Minimising the difference in pixel values of the input and reconstructed data instances has yielded promising results so far. However, it has limitations. The issue is that all aspects of the data are valued equally by this approach, regardless of which parts of the data are relevant/irrelevant to understanding the world/task at hand (please see section 2.1.3 and Figure 2.3 for more). The introduction of a new objective, perhaps one which contains a learning signal indicating the likely usefulness of particular representations (perhaps the reward from a reinforcement learning agent operating in the latent space), could be a potential solution.
4. **Further regularisation**. VQ-VAEs require powerful, state of the art and computationally burdensome autoregressive (or as shown in this study GAN) priors to be fitted to the latent space. This contrasts to VAE's static gaussian prior. A danger with swapping from VAEs to VQ-VAEs in world model architectures is that the added complexity of the latent space that is formed (requiring powerful priors) will cancel out the benefits that VQ-VAEs provide. Therefore, efforts to further regularise the VQ-VAE, with the result of a more simple prior being sufficient, are required.

Bibliography

2020. URL https://www.carbonfootprint.com/docs/2018_8_electricity_factors_august_2018_-_online_sources.pdf. pages 57

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017. pages 24

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017. pages 24

Yoshua Bengio. Deep representation learning, October 2019. pages 6, 37

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2000. pages 36

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2012. pages 1, 6, 7

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer, 2018. pages 37, 38

D. R. Bradley and H. M. Petry. Organizational determinants of subjective contour. *Amer. J. Psych*, 1977. pages 4

D. Brewster. A notice explaining the cause of an optical phenomenon observed by the rev. w. selwyn. *Report of the British Association for the Advancement of Science*, 1844. pages 4

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis, 2018. pages 19, 25, 26

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. pages 4

Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in -vae, 2018. pages 12, 13

Robert M. Stark Carla C. Morris. *Finite Mathematics: Models and Applications*. John Wiley Sons, 2015. pages 37

Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder, 2016. pages 17

Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. Pixelsnail: An improved autoregressive generative model, 2017. pages 19, 30

Aidan Clark, Jeff Donahue, and Karen Simonyan. Adversarial video generation on complex datasets, 2019. pages 26, 27

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings. URL http://proceedings.mlr.press/v15/coates11a.html. pages 38

W Caan D I Perrett, E T Rolls. Visual neurones responsive to faces in the monkey temporal cortex. *Experimental Brain Research*, 4, 1982. doi: 10.1007/BF00239352. URL https://pubmed.ncbi.nlm.nih.gov/7128705/. pages 3

Daniela Dentico, Bing Leung Cheung, Jui-Yang Chang, Jeffrey Guokas, Melanie Boly, Giulio Tononi, and Barry Van Veen. Reversal of cortical information flow during visual imagery as compared to visual perception. *NeuroImage*, 100:237–243, 2014. doi: 10.1016/j.neuroimage.2014.05.081. pages 4

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020. pages 19, 20, 21, 22, 28, 36, 44, 48

N. Dijkstra, P. Zeidman, S. Ondobaka, M. A. J. Van Gerven, and K. Friston. Distinct top-down and bottom-up brain connectivity during visual perception and imagery. *Scientific Reports*, 7(1), 2017. doi: 10.1038/s41598-017-05888-8. pages 4

Carl Doersch. Tutorial on variational autoencoders, 2016. pages 10, 11, 12, 30

Dileep George. *How the Brain Might Work: A Hierarchical and Temporal Model for Learning and Recognition*. PhD thesis, 01 2008. pages 3

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. pages 6, 38

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. pages 23, 24

Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 21*, pp. 545–552. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3449-offline-handwriting-recognition-with-multidimensional-recurrent-neural-networks.pdf. pages 17, 25

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans, 2017. pages 30

David Ha and Douglas Eck. A neural representation of sketch drawings, 2017. pages 5

David Ha and Jürgen Schmidhuber. World models, 2018. pages 1, 4, 5, 48

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2019. pages 4, 5, 48

Demis Hassabis, Dharshan Kumaran, and Eleanor A. Maguire. Using imagination to understand the neural basis of episodic memory. *Journal of Neuroscience*, 27(52):14365–14374, 2007. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.4549-07.2007. URL https://www.jneurosci.org/content/27/52/14365. pages 4

Maguire EA. Hassabis D. Deconstructing episodic memory with construction. *Pub.Med*, 2007. ISSN 0270-6474. doi: 10.1016/j.tics.2007.05.001. URL https://pubmed.ncbi.nlm.nih.gov/17548229/?dopt=Abstract. pages 4

J. Hawkins and D. George. Hierarchical temporal memory concepts , theory , and terminology. 2006. pages 3

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. pages 9

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735. pages 17, 25

Arka Pal Christopher Burgess Xavier Glorot Matthew Botvinick Shakir Mohamed Alexander Lerchner Irina Higgins, Loic Matthey. β-vae: Learning basic visual concepts with a constrained variational framework, 2017. pages 12

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax, 2016. pages 22

Emmanuel Kahembwe and Subramanian Ramamoorthy. Lower dimensional kernels for video discriminators, 2019. pages 25, 26

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017. pages 25, 30

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018. pages 30

W. Karush. Minima of functions of several variables with inequalities as side constraints. 1939. pages 12

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013. pages 1, 10, 11, 12, 14, 30

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving variational inference with inverse autoregressive flow, 2016. pages 13, 14

Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. doi: 10.1002/aic.690370209. pages 6

H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pp. 481–492, Berkeley, Calif., 1951. University of California Press. URL https://projecteuclid.org/euclid.bsmsp/1200500249. pages 12

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019. pages 57

Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area v2. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis (eds.), *Advances in Neural Information Processing Systems 20*, pp. 873–880. Curran Associates, Inc., 2008. URL http://papers.nips.cc/paper/3313-sparse-deep-belief-net-model-for-visual-area-v2.pdf. pages 9

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. pages 38, 57

N. K. Logothetis and J. Pauls. Psychophysical and Physiological Evidence for Viewer-centered Object Representations in the Primate. *Cerebral Cortex*, 5(3):270–288, 05 1995. ISSN 1047-3211. doi: 10.1093/cercor/5.3.270. URL https://doi.org/10.1093/cercor/5.3.270. pages 3

Alireza Makhzani and Brendan Frey. k-sparse autoencoders, 2013. pages 9

Gerrit W. Maus, Jason Fischer, and David Whitney. Motion-dependent representation of space in area mt+. *Neuron*, 78:554–562, 2013. doi: 10.1016/j.neuron.2013.03.010. pages 4

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. ISSN 14764687. doi: 10.1038/nature14236. URL https://doi.org/10.1038/nature14236. pages 5

John A. King John O'Keefe Neil Burgess, Suzanna Becker. Memory for events and their spatial context: models and experiments. *Phil. Trans*, 2001. URL https://royalsocietypublishing.org/doi/10.1098/rstb.2001.0948. pages 4

Nora Nortmann, Sascha Rekauzke, Selim Onat, Peter König, and Dirk Jancke. Primary visual cortex represents the difference between past and present. *Cerebral Cortex*, 25:1427–1440, 2013. doi: 10.1093/cercor/bht318. pages 4

Charles-Edouard Notredame, Delphine Pins, Sophie Deneve, and Renaud Jardri. What visual illusions teach us about schizophrenia. *Frontiers in Integrative Neuroscience*, 8, 2014. doi: 10.3389/fnint.2014.00063. pages 4

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans, 2016. pages 25

David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods, 2016. pages 23

R. Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435(7045):1102–1107, 2005. doi: 10.1038/nature03687. URL https://doi.org/10.1038/nature03687. pages 3

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015. pages 30

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017. pages 15

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. pages 17, 19, 30, 36

Asim Roy. The theory of localist representation and of a purely abstract cognitive system: The evidence from cortical columns, category cells, and multisensory neurons. *Frontiers in Psychology*, 8:186, 2017. ISSN 1664-1078. doi: 10.3389/fpsyg.2017.00186. URL https://www.frontiersin.org/article/10.3389/fpsyg.2017.00186. pages 3

Asim Roy, Leonid Perlovsky, Tarek R. Besold, Juyang Weng, and Jonathan C. W. Edwards. Editorial: Representation in the brain. *Frontiers in Psychology*, 9:1410, 2018. ISSN 1664-1078. doi: 10.3389/fpsyg.2018.01410. URL https://www.frontiersin.org/article/10.3389/fpsyg.2018.01410. pages 1, 3

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. pages 36

Masaki Saito, Eiichi Matsumoto, and Shunta Saito. Temporal generative adversarial nets with singular value clipping, 2016. pages 26

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016a. pages 24

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2234–2242. Curran Associates, Inc., 2016b. URL http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf. pages 23

Shagun Sodhani, Sarath Chandar, and Yoshua Bengio. Towards training recurrent neural networks for lifelong learning, 2018. pages 18

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998. URL http://www.cs.ualberta.ca/~sutton/book/the-book.html. pages 5

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method, 2000. pages 6

John K. Tsotsos. A 'complexity level' analysis of immediate vision. *International Journal of Computer Vision*, 1988. doi: https://doi.org/10.1007/BF00133569. URL https://link.springer.com/article/10.1007/BF00133569#citeas. pages 3

John K. Tsotsos. Complexity level analysis revisited: What can 30 years of hindsight tell us about how the brain might represent visual information? *Frontiers in Psychology*, 8:1216, 2017. ISSN 1664-1078. doi: 10.3389/fpsyg.2017.01216. URL https://www.frontiersin.org/article/10.3389/fpsyg.2017.01216. pages 3

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation, 2017. pages 26, 27

Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder, 2020. pages 1, 5, 13, 14, 15, 49

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016a. pages 17

Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks, 2016b. pages 17, 18, 30

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016c. pages 17, 18

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2017. pages 1, 16, 17, 19, 30, 36

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. pages 19, 30

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294. pages 7, 8

Lilian Weng. From autoencoder to beta-vae, 2018. URL https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html#references. pages 10, 11

Will Williams, Sam Ringer, Tom Ash, John Hughes, David MacLeod, and Jamie Dougherty. Hierarchical quantized autoencoders, 2020. pages 22, 23

Hanwei Wu and Markus Flierl. Vector quantization-based regularization for autoencoders. *arXiv:1905.11062v2*, 2020. pages 21, 46, 47

Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2018. pages 30, 32

Steven Zucker. Computer vision and human perception: An essay on the discovery of constraints. pp. 1102, 01 1981. pages 1, 3

Adrian Łańcucki, Jan Chorowski, Guillaume Sanchez, Ricard Marxer, Nanxin Chen, Hans J. G. A. Dolfing, Sameer Khurana, Tanel Alumäe, and Antoine Laurent. Robust training of vector quantized bottleneck models, 2020. pages 16, 21

Chapter 6

Appendix

6.1 Model Architectures

Table 6.1: Pixel autoregressive priors

	Top Prior (32x32)	Bottom Prior (64x64)
Input dimensions	32x32	64x64
Residual channels	256	256
Pixel blocks	4	4
Residual blocks per Pixel block	4	4
Conv layers per residual block	2	2
Conditional Residual blocks	0	3
Attention Layers	4	0
Convolutional Kernel size	5	5
Drop out	0.1	0.1
Learning Rate	0.0003	0.0003

Table 6.2: VQ-VAE-2

	VQ-VAE-2
Input dimensions	256x256
Latent tiers	64x64, 32x32
β	0.25
Residual channels	128
Residual blocks	2
Number of Conv layers per residual block	2
Decoder upsampling Convolutional Kernel size	4
Encoder Convolutional Kernel size	3
Total number of conv layers Encoder	25
Total number of conv layers Decoder	12
Codebook size (K)	512
Codebook dimension (D)	64
Learning Rate	0.0003

Table 6.3: Progressive GAN priors. Note the Generator and Discriminator mirror each other execept for the input and output layers

	Top GAN (32x32)	Bottom GAN (64x64)
Latent dimensions	32x32	64x64
Number of scales	4	5
Training steps per scale	[48000,96000 ,96000,200000]	[48000,48000,96000 ,96000,200000]
Number of channels at each scale	[512, 256, 64, 32]	[512, 256, 64, 32, 16]
Minibatch size at each scale	[256, 256, 128, 20]	[256, 256, 128, 15, 15]
Sampled noise latent vector	512	512
Loss mode	WGAN-GP	WGAN-GP
Residual layers per scale	4	4
Conditional Residual blocks	0	4
Self attention layers	1	1
Convolutional Kernel size	3	3
Encoder output activation	Linear	Linear
Learning Rate Encoder	0.0001	0.0001
Learning Rate Decoder	0.0004	0.0004

6.2 Ethical Consideration

6.2.1 Environmental Considerations

Experiments were conducted using a range of Imperial computing infrastructure, which was estimated to have a carbon efficiency of 0.432 $kgCO_2eq/kWh$, the European average (car, 2020). A cumulative of 6792 hours of computation was performed on a TITAN X Pascal. Total emissions are estimated to be 733.54 $kgCO_2eq$. Estimations were conducted using the MachineLearning Impact calculator presented in Lacoste et al. (2019). This is marginally more than the estimated CO2 emission of a single economy-class return flight from London to New York [1]

6.2.2 Datasets used

A range of datasets were used in this project.

1. FFHQ: all images are covered by either the Creative Commons BY 2.0, Creative Commons BY-NC 2.0, Public Domain Mark 1.0, Public Domain CC0 1.0, or U.S. Government Works license. This allows for the use, redistribution and adaption of all images for non-commercial purposes.
2. CelebA: although no evidence of any license was found, in the “agreement” section on the dataset website it states that this dataset can only be used for non-commercial research(Liu et al., 2015).
3. ImageNet: No explicit license can be found for the ImageNet dataset. This maybe due to the dataset simply providing the URLs to the images which can then be downloaded. Considering people do feature in the images, the legality into the dataset should be further investigated.

6.2.3 Ethical Checklist

[1] According to the International Civil Aviation Organization (ICAO)

	Yes	No	comments
Section 1: HUMAN EMBRYOS/FOETUSES			
Does your project involve Human Embryonic Stem Cells?		x	
Does your project involve the use of human embryos?		x	
Does your project involve the use of human foetal tissues / cells?		x	
Section 2: HUMANS			
Does your project involve human participants?		x	
Section 3: HUMAN CELLS / TISSUES			
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		x	
Section 4: PROTECTION OF PERSONAL DATA			
Does your project involve personal data collection and/or processing?	x		Two datasets involving human faces are processed/trained on
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	x		Machine Learning algorithms can easily infer the ethnicity of the face that it is being processed. Classifying race counts as processing
Does it involve processing of genetic information?		x	
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		x	
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?	x		The datasets CelebA and FFHQ are trained on. Along with a subset of ImageNet is created featuring non human animals
Section 5: ANIMALS			
Does your project involve animals?		x	
Section 6: DEVELOPING COUNTRIES			
Does your project involve developing countries?		x	
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		x	
Could the situation in the country put the individuals taking part in the project at risk?		x	
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY			
Does your project involve the use of elements that may cause harm to the environment, animals or plants?	x		Training large models on GPUs takes a considerable amount of energy, which in the UK, a portion of which is generated from an unsustainable source. See section
Does your project deal with endangered fauna and/or flora /protected areas?		x	
Does your project involve the use of elements that may cause harm to humans, including project staff?		x	
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		x	

Table 6.4: Ethics Checklist, part 1. Adapted from the Imperial College Department of Computing website `https://www.doc.ic.ac.uk/lab/msc-projects/ethics-checklist.xlsx`.

	Yes	No	comments
Section 8: DUAL USE			
Does your project have the potential for military applications?		x	As detailed in depth in this report high-level representations and explicit mental models are required for higher level reason. To this end this project contributes to a final product that could be used for military purposes, such as automated drones.
Does your project have an exclusive civilian application focus?		x	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		x	
Does your project affect current standards in military ethics ? e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		x	
Section 9: MIS USE			
Does your project have the potential for malevolent/criminal/terrorist abuse?		x	
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		x	
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?	x		This report endeavours to make progress towards a higher level of automated reasoning, therefore could end up helping the development of oppressive technologies
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		x	
SECTION 10: LEGAL ISSUES			
Will your project use or produce software for which there are copyright licensing implications?		x	
Will your project use or produce goods or information for which there are data protection, or other legal implications?		x	
SECTION 11: OTHER ETHICS ISSUES			
Are there any other ethics issues that should be taken into consideration?		x	

Table 6.5: Ethics Checklist, part 2. Adapted from the Imperial College Department of Computing website `https://www.doc.ic.ac.uk/lab/msc-projects/ethics-checklist.xlsx`.