UNIVERSITY OF AMSTERDAM

MSC ARTIFICIAL INTELLIGENCE
MASTER THESIS

SELF-SUPERVISED VARIATIONAL AUTO-ENCODERS

by

IOANNIS GATOPOULOS

12141666

September 6, 2021

48 ECTS
November - September 2020

Daily Supervisor:
Dr Jakub M. TOMCZAK

Supervisor:
Maarten C. STOL, MSc

Examiner:
Dr Efstratios GAVVES

BRAINCREATORS

VRIJE UNIVERSITEIT
AMSTERDAM

Abstract

In this thesis, we present a novel class of generative models, called *Self-Supervised Variational Auto-Encoder* (ssVAE), where we improve plain architecture (VAE) through self-supervised representations. The proposed method is able to perform both conditional and unconditional sampling, while demonstrating improved performance in sample synthesis and density estimation on standard image modelling benchmarks. Starting with the special case, where we use a single self-supervised transformation as a latent variable, we generalize to a multi-levelled architecture, which is agnostic to the provided representations. We show that the benefits of employing a multi-levelled self-supervised framework against the Variational Auto-Encoder (VAE) is more obvious when we move to higher-dimensional data, where the second fails to scale efficiently. Furthermore, we examine the advantages of choosing suitable representations for the data at hand when we want to focus on specific characteristics, which indicates a notion of introducing prior knowledge to the data generation process, but also allows for additional functionalities. The flexibility of ssVAE in data reconstruction through multiple ways find a particularly interesting use case in data compression tasks, where we can trade-off more memory for better data quality, and vice-versa. Additionally, we suggest replacing the fixed latent prior with a data-driven bijective network, as it allows the model to reach better likelihood estimations and impressive generations.

Acknowledgements

I would like to thank the following individuals and institutes who did not only help me to accomplish this thesis, but also helped to get as far as this point of my academic journey.

First, I would like to thank the University of Amsterdam, for giving me the opportunity to study and develop my skills further, in such a vibrant and diverse student community. BrainCreators, for taking me under their roof and allowing me to conduct my thesis while receiving their valuable support. Especially, I would like to thank Maarten Stol, general manager of the R&D department, for his trust and countless, fruitful discussions, in order to provide me with ideas, help me organize, and motivate me to bring this big project to this wonderful conclusion.

Efstratios Gavves, assistant professor of the University of Amsterdam, was the first person who ever believed in me and he is heavily responsible for the turn that my academic journey has gotten. Early on, he suggested me to read my first academic paper, namely the "Auto-Encoding Variational Bayes", which I loved and end up pursuing a thesis on it. He gave me the opportunity to collaborate with amazing researchers that made a tremendous impact on me, and even his very busy schedule, he was always there to help and advise me for my next steps. For his tremendous impact on my academic life these past two years, I will be forever grateful.

I met my supervisor, Jakub M. Tomczak, during my first internship, where we work together on a four-month project. Immediately, I witness his underline passion for research and innovation, which resembles my desire of pushing the boundaries of science further with novel ideas. I am extremely happy and honoured that he agreed to work with me on an idea that was conceptualized during a bus trip, and produce this very thesis. From our conversations during midnight to our brainstorming in front of a whiteboard, these past months were quite an exciting journey. As he is truly a role model of mine, I hope that one day I will become half of the researcher that he is, and affect the people around me in the way that he does.

My dear colleagues from all over the world and my friends from Greece, who always provided moral support and create such a fun environment, that always helped me to relax and reset my mind. Especially, I would like to thank Anna, for always been there for me, giving me countless laughs, beautiful moments and a reason for me to push myself to places that I could never imagine.

Lastly, this would not be possible without my family; my parents Vasilis and Ekaterini, and my brother Dimosthenis. Due to their unconditional love, support and sacrifices, I was able to get to the place where I am today, develop myself, and meet all these wonderful people. No words can describe their contributions and my love for them.

From the bottom of my heart,
Ioannis Gatopoulos.

[illegible]

Contents

1 Introduction **5**

2 Evolution of Artificial Neural Networks **8**

1 Convolutional Neural Networks 8

2 Modern Architectures 9

3 Variational Auto-Encoders **12**

1 Variational Inference 12

2 Scaling up the Variational Inference: Variational Amortised Inference 15

3 Optimization of VAEs 16

4 Improving VAEs 19

4 Flow-based Generative Networks **23**

1 Normalizing Flows 23

2 Optimization and Computational Complexity of FGNs 25

3 Categories of FGNs 26

4 Partitioned transformation flows 27

5 Method **29**

1 VAE with bijective prior 29

2 Self-Supervised Variational Auto-Encoders 31

2.1 Problem Formulation and Calculation of Lower Bound 31

2.2 Properties 33

2.3 Self-Supervised VAE 34

2.4 Hierarchical Self-Supervised VAE 35

2.5 Image Transformations 36

3 Neural Network Architecture 36

6 Experimental Setup **39**

1 Datasets 39

2 Optimisation and hyper-parameters 40
3 Evaluation Metrics 40
4 Hardware, Software and Experimental Code 40

7 Evaluation **41**
1 The impact of the bijective prior on VAEs 41
2 Self-Supervised Variational Auto-Encoders 43
3 Extending Self-Supervised VAE 47
3.1 Self-Supervised VAE with sketch representations 47
3.2 Multi-resolution Self-Supervised VAE 49
3.3 Comparison with VAE 50
4 Interpolation through latent spaces 52

8 Conclusion **53**

Bibliography **57**

Appendix **58**
A.1 Derivation of ELBO though Jensen's inequality 58
A.2 Derivation of Kullback-Leibler divergence between two isotopic Gaussians 58
A.3 Additional qualitative results 60
A.4 Thesis Statistics 60

List of Figures

1.1 Eye and vision development in infants 5
1.2 Bayesian directed graph of the proposed method 7

2.1 Image Processing with Neural Networks 9
2.2 Convolutional Neural Networks Architectures 10

3.1 Illustration of Variational Inference Task 14
3.2 Variational Posterior Approximations 15
3.3 Variational Auto-Encoder 16
3.4 Illustration of the reparameterization trick 18
3.5 Variational Posteriors 20
3.6 Illustration of the Prior 21
3.7 Discretized Logistic Distribution 21

4.1 Normalizing flow 25
4.2 Categories of Flow-based Generative Networks 26
4.3 Masking schemes for affine coupling layers 28

5.1 VAE with Bijective Prior 30
5.2 Direct graphical model graph of Self-Supervised VAE 31
5.3 Delta function 33
5.4 Self-Supervised Variational Auto-Encoder’s reconstruction methods. 35
5.5 Hierarchical Self-Supervised VAE architecture 36
5.6 Image Transformations 37
5.7 Architecture of the autoencoder 38

6.1 Samples from the CIFAR-10, ImageNet and, CelebA image datasets 40

7.1 VAE prior comparison 42
7.2 ssVAE CIFAR-10 Qualitative results 44

7.3 ssVAE CIFAR-10 Qualitative results 45
7.4 Reconstructions of ssVAE-sketch on CelebA 47
7.5 Sample generations of ssVAE-sketch on CelebA 48
7.6 Qualitative results on CelebA for 3-level ssVAE 49
7.7 Comparison on image reconstructions (CelebA) 50
7.8 Unconditional generation of CelebA. 51
7.9 ssVAE latent space interpolation. 52

8.1 VAE with RealNVP prior on CIFAR-10 reconstructions and interpolations 61
8.2 VAE with RealNVP prior CIFAR-10 unconditional generation samples 61
8.3 VAE with RealNVP prior on ImageNet32 reconstructions and interpolations 62
8.4 VAE with RealNVP prior ImageNet32 unconditional generation samples 62
8.5 Qualitative results on CIFAR-10 of the ssVAE 63
8.6 Unconditional generations of the ssVAE trained on CIFAR10 64
8.7 Qualitative results on ImageNet32 of the ssVAE 65
8.8 Unconditional generations of the ssVAE trained on ImageNet32 66
8.9 VAE with RealNVP prior on CelebA reconstructions and interpolations 67
8.10 VAE with RealNVP prior on CelebA reconstructions and interpolations 67
8.11 VAE with RealNVP prior on CelebA reconstructions and interpolations 68
8.12 Qualitative results on CelebA of the ssVAE-sketch 69
8.13 Image interpolations on CelebA of the ssVAE-sketch 69
8.14 Unconditional generations of the ssVAE-sketch trained on CelebA 70
8.15 Qualitative results on CelebA of the ssVAE-3lvl 71
8.16 Unconditional generations of the ssVAE-3lvl trained on CelebA 71
8.17 Comparison on CelebA unconditional generations 72
8.18 Comparison on CelebA reconstructions 72

List of Tables

1.1 Comparison of Deep Generative Neural Networks. 6

7.1 VAE with different priors for CIFAR-10 41
7.2 Density estimation with VAE and ssVAE for CIFAR-10 and ImageNet32 43
7.3 Generative modelling performance for natural images 46
7.4 FID scores for CIFAR-10 46
7.5 Generative modelling performance on CelebA 51
7.6 Density estimation with VAE and ssVAE for Imagenette64 52

Chapter 1

Introduction

> “*If we knew what we are doing, it would not be called research, would it?.*“
>
> — Albert Einstein

Unlike many other sensory systems, the human visual system (i.e. the components from the eye to neural circuits) develops largely after birth, especially in the first few years of life (Banks and Salapatek, 1978). In the beginning, even though the visual structures are fully present, they are still immature in their potentials. Newborns cannot detect objects that are further than a distance of twelve inches, and observe the world in black, white and grey colours. As these neural circuits adapt to natural light and reach their full potentials, they learn to enhance the already known signals with the new information that is becoming available (Figure 1.1). Furthermore, Ayzenberg V. (2019) suggested that human's visual system for object recognition tasks initially starts with the skeletal structure of the object and then maps other properties, such as textures and colours, onto it before it is classified. This allows determining an object within milliseconds, and with seemingly little effort. It seems that humans reinforce their capabilities by sequentially applying new content of information over time and some specific processes, like object recognition, are divided into two or simpler tasks.

Deep Neural Networks (NNs) employed with stochastic gradient descent, made possible for learning algorithms to model high-dimensional data, and in tasks like image classification and object recognition, to reach human-level performance. Interestingly, exploiting NNs capabilities for approximating the distribution of high-dimensional data, like images or text, while encoding them into low-dimensional representations (or *embeddings*) in an unsupervised way, is one of the current emerging research lines in machine learning. Its key goal is to identify and disentangle the underlying causal factors of the observations, so that it becomes easier

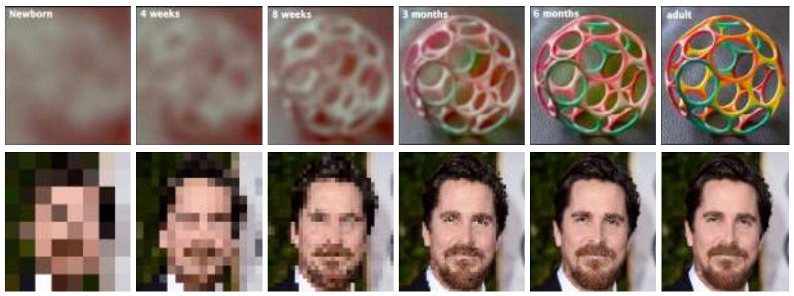


Figure 1.1: **Top**: Infant vision development during the first six months. The previous processed information is enriched with new signals in order for the sensory systems to be able to analyze high fidelity signals (Haskett, 2019). **Bottom**: A similar pattern of progressively decreasing the complexity of an image but preserving its global structure by sequential downscaling.

	Latent Variable Models	**Flow-based Models**	**Autoregressive Models**	**GANs**
Optimization Function	*Approximate likelihood (stable)*	*Exact likelihood (stable)*	*Exact likelihood (stable)*	*Minimax loss (unstable)*
Latent Space	*Meaningful (Low-Dimensional)*	*Meaningful (High-Dimensional)*	*Meaningless (High-Dimensional)*	*Meaningful (Low-Dimensional)*
Incorporate Encoder	*Yes*	*Yes*	*No*	*No*
Sampling	*Fast*	*Fast*	*Slow*	*Fast*

Table 1.1: Comparison of Deep Generative Neural Networks.

to understand them, to classify them, or to manipulate them in a meaningful way (Bengio et al., 2012). For image data, this often means that we are interested in uncovering the *global* structure that captures the content of an image (for example, the outline of objects present in the image) and its *local* structure (or *style*), but that we are typically less interested in the high-frequency sources of variation such as the specific textures or white noise patterns (Chen et al., 2016). Since these kinds of models, known as *(deep) generative models*, would build intuition around the observed data, the range of applications that come along are vast; from image and video compression (Kingma et al., 2019; Hoogeboom et al., 2019; Habibian et al., 2019), to audio synthesis and drug discovery (van den Oord et al., 2016a; Polykovskiy et al., 2018).

One can categorize the generative models into two groups according to their density modelling approach; the *explicit* and *implicit* density models. The latter refers mainly to the generative adversarial network (GAN) (Goodfellow et al., 2014) which objective function, known as *minimax loss*, theoretically approximates the minimization of the symmetric *Jensen–Shannon* divergence metric between the target and the predicted values. Despite their impressive performance in generating realistic high-dimensional images, the lack of encoder forbid them to infer the latent codes, suffer from training instabilities and usually, they do not have full support over the data distribution. In essence, they provide improved sample quality but at the cost of missing modes. The former category includes all the models that use the likelihood of the given data as an objective function. Three main classes of powerful and very successful explicit density models are the *autoregressive models* (ARM) (Larochelle and Murray, 2011; van den Oord et al., 2016b), the *probabilistic latent variable models* (LVM) (Kingma and Welling, 2013) and the *flow-based models* (FGN) (Dinh et al., 2016; Kingma and Dhariwal, 2018). Their features are grouped on Table 1.1.

Autoregressive models, such as the PixelRNN and PixelCNN (van den Oord et al., 2016b; Salimans et al., 2017), exploit the conditional factorization of the likelihood and have demonstrated superior likelihood performance. However, due to the costly run-time sampling performance and pixel-level "uninformative" latent variables, they scale poorly with the dimensionality of the input data. The focus of this research lies on a specific probabilistic latent variable model, called Variational Auto-Encoder (VAE) (Kingma and Welling, 2013; Rezende et al., 2014). The framework of Variational Auto-Encoders provides a principled method for jointly learning latent-variables and the corresponding inference models. However, their performance in terms of likelihood score and quality of generated samples are far from the desired one.

Contributions

In this thesis, inspired by the visual learning process of humans, we enhance the framework of the Variational Auto-Encoder by introducing a compressed representation of the data as a random variable, without any restrictions of its form. As an example, for images this can be the label of the presented object, a grey-scale or Fourier transformation, a sketch representation etc. By utilizing a conditional dependence relationship with the target data, the generation process is broken down into two simpler tasks; the discovery of the

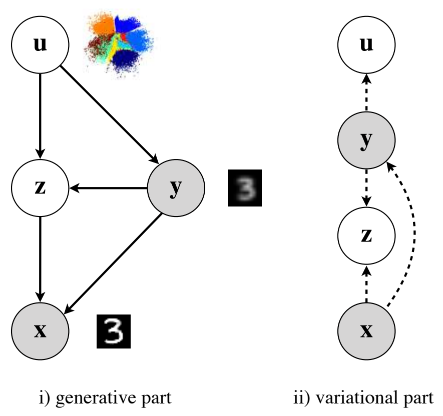


Figure 1.2: Stochastic dependencies of the proposed model. Our approach takes advantage of a compressed representation $\mathbf{y}$ of the data in the variational part. When the compressed representation is a downscaled version of the given image, it utilises a super-resolution functionality in the generative part. The white and grey nodes represent stochastic and deterministic variables respectively. We will go into detail in chapter 5.

underlying distribution of the compressed representation, which with the help of an additional latent variable, will guide the model to discover the complex distribution of the target observations. In this way, the proposed framework can thoroughly generate both novel content in an unconditional fashion but also produce conditional generations under the provided description of the compressed representation. As a result, we obtain a two-level VAE with three latent variables, where one is the compressed representation of the original data, essentially forming a *self-supervised* generative model. The proposed method is illustrated as a directed Bayesian graph in Figure 1.2.

The following statements shall be investigated:

- What are the advantages of this two-staged process over the plain architecture of VAEs, in terms of density estimation and data generation? Does the new framework introduce more operations apart from generation and reconstruction?
- Can the model learn the missing information between the compressed and target images successfully? If so, can we extend the framework by utilizing multiple compressed representations?
- How will the choice of the compressed representation affect the models' performance? Given that we choose a single downscaled transformation of the image, can we have a likelihood-based super-resolution model? Can the additional functionalities help us in data compression tasks?

There are also some key technical challenges and questions that arise:

- What would be the training objective of this model?
- How can we model the prior of the model $p(\mathbf{u})$ effectively and what would be its influence on the overall framework?
- How can we make the model more efficient in terms of training and inference? Are there any mathematical properties that we can take advantage of and design powerful and scalable neural networks?

The structure of this thesis is as follows. In chapter 2 we briefly present the advantages in deep learning, with a focus on the progression over convolutional neural network architectures. The chapters 3 and 4 provide an extensive introduction and analysis of Variational Auto-Encoders and flow-based generative models. These will establish the theoretical framework that our contributions build upon. We continue with chapter 5, where we present our approach and our novel contributions in detail. Our experimental setup and results are presented in chapters 6 and 7, along with an throughout analysis of the presented experiments. Finally, we discuss our final conclusions in chapter 8, with a brief outlook on future areas of research regarding this topic.

Chapter 2

Evolution of Artificial Neural Networks

"The measure of intelligence is the ability to change"

— Albert Einstein

In this chapter, we will start by briefly outlining the origins of artificial neural networks. In particular, we will focus on a special type of networks, namely convolutional neural networks, where we will discuss their functionality and their development over the years. Then, we will present modern architectural approaches, which are easier to optimize and efficiently allow for deep representations, which are proved to be of critical importance.

1 Convolutional Neural Networks

In 1943, Warren McCulloch and Walter Pitts published a scientific study in theoretical neurophysiology, entitled *A Logical Calculus of Ideas Immanent in Nervous Activity*, which would become the landmark of modern artificial intelligence (AI) technology (McCulloch and Pitts, 1988). In their effort to understand the functionality of the brain, they presented a highly simplified model of a single biological brain cell, called *artificial neuron*, or simply *neuron*. The assumption was that the neurons would act as modules of a complicated network, called a *neural network* (NN), and employed with logical operations, they could *learn* from the data and discover highly-complex patterns. The first attempt to put this idea into practice was from Frank Rosenblatt, who introduced the *Perceptron*; a single layer of neurons that were connected through a linear equation of learnable weights and generated an output (Rosenblatt, 1960). Even though the model could distinguish linearly separable data, it failed on more complicated tasks, like the *Exclusive OR* (XOR) classification problem. It was estimated that a network would require more than one layer to capture a nonlinear behaviour, which was not possible with the proposed methodology. The XOR and the training of *deep* architectures were solved with the introduction of *Back-Propagation* (Werbos, 1994), an algorithm which later proved that it could learn interesting internal representations of the data (Rumelhart et al., 1986). Although neural nets did solve a few toy problems by extracting features of the observations, the demand for enormous computational power and a large amount of data prevented them for being a general problem-solving tool. With the availability of Graphics Processing Units (GPUs) that allowed fast matrix multiplications and the opportunity to collect large chunks of data though the world-wide-web, the breakthrough that demonstrated the power of neural networks was in 2012, where AlexNet won the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Krizhevsky et al., 2012).

Since then, many researchers focused on the design of different types of neural networks, in order to make them more efficient and scalable. In general, the choice of the building blocks is closely related to the nature of the processed data. Deep NNs are mainly used to process and extract features from *unstructured data*, like image, speech and text. Specifically, for sequential data, the most popular architectures are the so called *recurrent neural nets* (RNNs) and *long short-term memory recurrent neural nets* (LSTMs), as they exhibit temporal

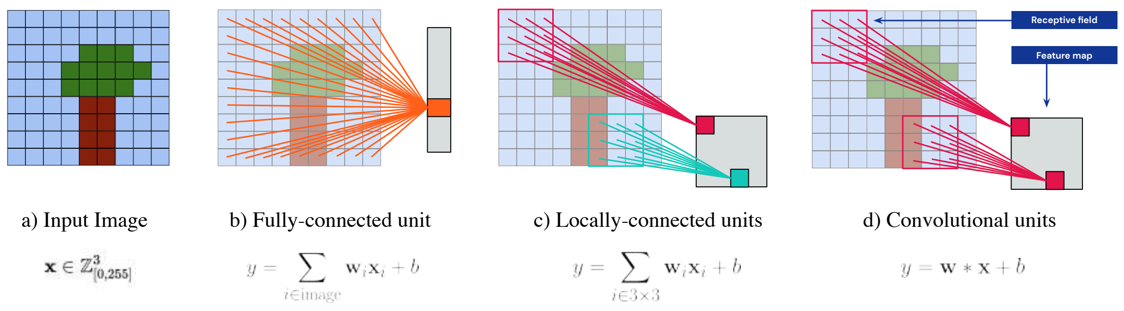

Receptive field
Feature map
a) Input Image
b) Fully-connected unit
c) Locally-connected units
d) Convolutional units
$\mathbf{x} \in \mathbb{Z}^3_{[0,255]}$
$y = \sum_{i \in \text{image}} \mathbf{w}_i \mathbf{x}_i + b$
$y = \sum_{i \in 3\times3} \mathbf{w}_i \mathbf{x}_i + b$
$y = \mathbf{w} * \mathbf{x} + b$

Figure 2.1: From fully-connected layers to locally-connected and convolution. The symbols + and ⊔ represent element-wise addition and channel-wise concatenation respectively.

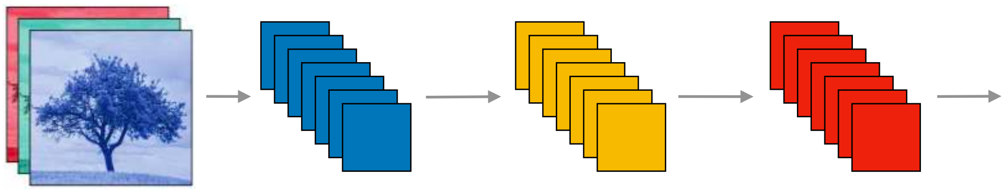
dynamic behaviour, which allows them to discover chronological dependent patterns. The focus of this thesis lies in *identical and independently distributed* data, and specifically image data, where *Convolutional Neural Networks* (CNNs or ConvNets) are considered to be the most dominant approach.

CNNs exploit two key properties of images; *locality*, as nearby pixel values are more strongly correlated than those who are further away, and *translation invariance*, because meaningful patterns can occur anywhere in the image. To take advantage of this topological structure, CNNs incorporate weight sharing and model hierarchy principles. The former uses the same network parameters to detect local patterns at many locations in the image, while in the latter, local low-level features are composed into larger, more abstract ones. The building blocks of CNNs are the locally-connected units which are represented by a two-dimensional grid. In contrast with fully connected layers, the locally-connected units sum over the contributions for the local region, called *receptive field*, of the image. Particularly, the weights that correspond to every receptive field of the image are shared, resulting in equivariance to translation outcome, called *feature map* or *channels* and the weights which are associated with each feature map are called the *kernel* or *filter*. In practice, we use multiple kernels that we convolve to obtain multiple channels. That means that the inputs and outputs between convolutions will be three-dimensional objects (height, width and channels), which are known as *tensors*, and in essence, each filter will consist of multiple sub-filters, one for every channel of the input tensor. Figure 2.1 illustrates the difference between fully-connected, locally-connected and convolutional units on image processing.

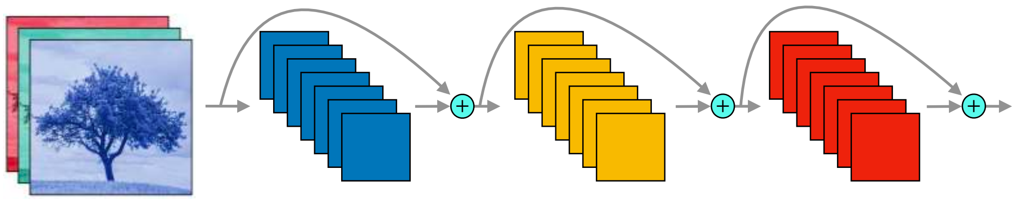
Importantly, the architectural choice of how these building blocks fit together in a deep neural network is crucial for the models' efficiency and scalability. The number of processing units and the non-linearity activation function that is going to be used are some core selections that defines the architecture. One of the earliest CNN models was LeNet-5 (Lecun et al., 1998), which was applied to recognise handwritten digits. Its design to sequentially reduce the input receptive field while expanding the number of feature maps is an architectural choice that is still used in modern networks. With the invention of Alexnet (Krizhevsky et al., 2012), the focus was then mainly shifted to use neural networks to process real-world images and perform classification tasks. Apart from proving that NNs have the potential to form a general problem-solving tool, AlexNet imported some important features. It replaced the $\tanh(\cdot)$ activation functions with $\mathrm{ReLU}(\cdot)$, and introduced two additional regularisation strategies; the *weight decay* and *dropout*. The former penalised the parameters with high magnitude while the latter randomly removed units during training, as this will make all the other units robust to potentially absent or distorted features. The next notable architecture, named VGGnet (Simonyan and Zisserman, 2014), introduced the concept of the *same* convolutional layer, where the input size is preserved after the operations. This will allow to use multiple sequential units and result in deep, more powerful models. While the convolution operation allows for numerous of different kernel sizes, the authors suggested that the 3×3 filter size should be preferred, as a convolution with a larger receptive field can be replaced with stacks of smaller ones, that additionally will permit the use of extra non-linearities.

2 Modern Architectures

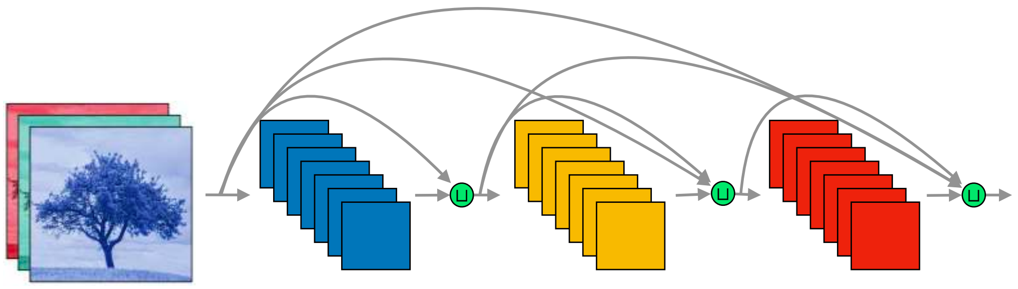
It was clear that deeper architectures indeed led to better results. The intuition is that the use of more layers will introduce more non-linearities and allow the model to discover more complex patterns. However, apart



a) Standard CNN connectivity



b) Residual (ResNet) connectivity



c) Dense (DenseNet) connectivity

Figure 2.2: Convolutional Neural Networks Architectures. The symbols + and $\sqcup$ represent element-wise addition and Channel-wise concatenation respectively.

from the additional computational complexity, a key limitation that prohibits the architectures to have arbitrary many layers is an optimization one. While the first intuition when designing a deep network may be to simply stack as many building blocks as possible, after a a certain point the performance quickly diminishes. The issue arises from how neural networks are trained through back-propagation. As during the training process the gradient signal must be propagated backwards through the entire network, for deeper architectures this signal essentially disappears by the time it reaches the top-level (first) layer. This issue is known as a *vanishing gradient*. The solution is to design a network in which the gradient can easier reach all the layers, without having any restrictions from its depth. This is the goal behind the following popular and back-bones of state-of-the-art architectures: *Residual Neural Networks* (ResNets) and *Densely Connected Networks* (DenseNets).

ResNets (He et al., 2015) revolutionized the CNN architecture design by introducing the concept of *residual learning* and devised an efficient methodology for the training of deep networks. Their building blocks are enhanced by utilizing a skip (or residual) connection, where the input is reintroduced and added to the output. Formally, for $\mathbf{h}_t$ being the t-th hidden layer parametrised by θ_t, the characteristic equation of the residual block would be:

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f\left(\mathbf{h}_t, \theta_t\right).$$

In contrast with traditional neural nets which learn the $\mathbf{h}_t$ directly, ResNet instead models the layers to learn the residual of input and output of subnetworks. This will give the network an option to just skip subnetworks by learning the identity mapping $\mathbf{h}_{t+1} = \mathbf{h}_t$. Thus, the identity function will preserve the gradient during back-propagation and, in theory, its performance will not decline by making the model deeper. Because of its compelling results on various computer vision tasks, ResNet quickly became one of the most popular architectures and showed that representational depth is of central importance.

Another variant of the idea of skip connections is DenseNet (Huang et al., 2016), where instead of utilizing residual connections by addition, it concatenates all feature maps directly with each other. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. In mathematical terms, a densely connected layer is expressed though the following equation:

$$\mathbf{h}_{t+1} = f\left([\mathbf{h}_i, ..., \mathbf{h}_t], \theta_{i:t}\right),$$

where $0 \leq i \leq t$ and $[\mathbf{h}_i, ..., \mathbf{h}_t]$ refers to the concatenation of the feature-maps produced in layers $[i, ..., t]$. In this way, the architecture ensures maximum information flow between layers in the network. Therefore, instead of drawing representational power from extremely deep or wide architectures, DenseNets exploit the potential of the network through feature reuse, yielding condensed models that are easy to train and highly parameter efficient. Concatenating feature-maps learned by different layers increases variation in the input of subsequent layers and improves efficiency. The connectivity architecture of plain CNNs, along with ResNet and DenseNet are presented in Figure 2.2.

Furthermore, various other improvements have been proposed from an optimization perspective. Techniques like elegant weight initialisation (Glorot and Bengio, 2010) and the introduction of normalisation layers (Ioffe and Szegedy, 2015; Salimans and Kingma, 2016) can help to deal with exploiting/vanishing gradients issues, but also boost the training performance of the model. Importantly, a big focus has been given to the optimization methods, which in essence, are the engines of the underlying neural networks that enable them to learn from data by adapting their parameters. Given the problem of the minimization of an objective function that measures the mistakes made by the model (such as prediction error or log-likelihood estimation in classification and density estimation tasks respectively), they work by making a sequence of small incremental changes to its parameters in order to reduce it (by, similarly, a small margin). In NNs, most commonly, this is done with stochastic (mini-batch) gradient descent (SGD). Apart from its efficiency, vanilla mini-batch gradient descent offers a few challenges. Choosing a proper learning rate, which determines the size of the steps we take to reach a (local) minimum, can be difficult, as too small values will lead to slow convergence, while too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge. Furthermore, SGD tends to oscillate the gradients across the slopes of the loss surface and does not award each parameter with the same update magnitude. However, these problems can be smoothed out by taking advantage of the *momentum* of the gradient and compute adaptive learning rates for individual parameters (Nesterov, 1983; Duchi et al., 2011; Kingma and Ba, 2014).

Chapter 3

Variational Auto-Encoders

“*What I cannot create, I do not understand.*“
— Richard Feynman

The framework of Variational Auto-Encoders is member of the latent variable models (LVMs) that provides a principled method for jointly learning latent-variable models and corresponding inference models. Due to its capability of estimating densities of complex distributions, while automatically learning meaningful (low-dimensional) representations from raw data, it is an attractive solution for generative modelling tasks. In this chapter, we explain the basic and advanced concepts of VAEs.

1 Variational Inference

Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^{\mathrm{D}}$ be independent and identically distributed observable variables that we wish to model with learnable parameters ϑ. Given that the dimensionality D of real-world data is typically large, thus resulting in a complex, high-dimensional distributions, approximating the marginal likelihood $p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n)$ becomes a difficult task. An interesting direction is to introduce latent (unobserved) variables $\mathbf{z}$, as they can dramatically increase the capacity of our model. The underlying idea is that each observation $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$ has a corresponding causal latent variable $\mathbf{z} \in \mathbb{R}^{\mathrm{M}}$, and by marginalizing it out, we result into a quite flexible distribution over $\mathbf{x}$. Consequently, we can model the data likelihood as follows:

$$p_\vartheta(\mathbf{x}) = \int p_\vartheta(\mathbf{x}, \mathbf{z})\, \mathrm{d}\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z})\, p_\lambda(\mathbf{z})\, \mathrm{d}\mathbf{z} \tag{3.1}$$

We refer to the first term as *likelihood* function and the second as *prior* over the latent variables. One can notice that in this way if $p(\mathbf{x}|\mathbf{z})$ is chosen to be a Gaussian distribution for continuous (or discrete) $\mathbf{z}$, the marginal likelihood can be interpreted as a mixture of infinite (or finite) Gaussian (MoG) distributions, which theoretically can approximate any smooth density with any specific nonzero amount of error. As a result, this often gives us the flexibility to model (3.1) using pre-defined distributions such as:

$$p_\theta(\mathbf{x}|\mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})), \qquad p_\lambda(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where $\mu_\theta(\cdot)$, $\Sigma_\theta(\cdot)$ describe the parametric way of how the latent variables influence the observations. A straightforward way to model these functions is to restrict them to a family of linear mappings, so that the parameters would be a result of a scale and translate transformation of the latent codes:

$$\mu(\mathbf{z}) = W \cdot \mathbf{z} + b, \quad \Sigma(\mathbf{z}) = \Sigma_0.$$

In this way, we obtain a highly scalable approach with a simple iterative training scheme, but with the trade-off that linear assumptions are often too restrictive for modelling high-dimensional data like natural images. This framework is known as *Probabilistic Principal Components Analysis* (PPCA) (Tipping and Bishop, 1999) and it is mainly used for dimensionality reduction purposes. Alternatively, to increase the models' capabilities, we can employ powerful deep neural networks instead. However, as the integral in (3.1) involves multiple non-linear stacked functions, the marginal likelihood is now typically intractable. From the *Bayes* law identity,

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})},$$

it is clear that this intractability relates to the intractability of the posterior distribution $p(\mathbf{z}|\mathbf{x})$, and vice versa. Thus, this computation often requires the use of approximate inference techniques of the posterior, which then lead to the computation of the marginal likelihood.

In general, there are two dominant approaches for approximating the posterior in a latent variable model setting; the Markov Chain Monte Carlo (MCMC) sampling methods and Variational Inference (VI). The former constructs an *ergodic* Markov Chain (an aperiodic chain where all possible states are reachable) on $\mathbf{z}$ whose stationary distribution is the posterior $p(\mathbf{z}|\mathbf{x})$. After enough samples, it is guaranteed that the Markov chain will eventually converge to the desired distribution, allowing us to sample from it. Even though developments on efficient ways to build a Markov chain, including the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970) and the Gibbs sampler (Geman and Geman, 1984), managed to form an unbiased and robust approach, on high-dimensional datasets the convergence is rather slow, due to the *curse of dimensionality*; as the number of dimensions grows, the amount of sample we need to approximate accurately grows exponentially. However, the latter's main idea is to turn the inference problem into an optimization one. By choosing a family of approximate densities $\mathcal{Q}$, the key task is to identify the member that minimizes a divergence measure to the exact posterior, so that $q(\mathbf{z}) \approx p(\mathbf{z}|\mathbf{x})$. Due to its simplicity and its inherent connection with the data likelihood divergence through Shannon's source coding theorem (Shannon, 1948), the Kullback-Leibler ($\mathcal{KL}$) is often preferred as a distance metric. Hence, we select the approximate posterior q^* so that:

$$q^*(\mathbf{z}) = \underset{q(\mathbf{z}) \in \mathcal{Q}}{\arg\min}\, \mathcal{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})).$$

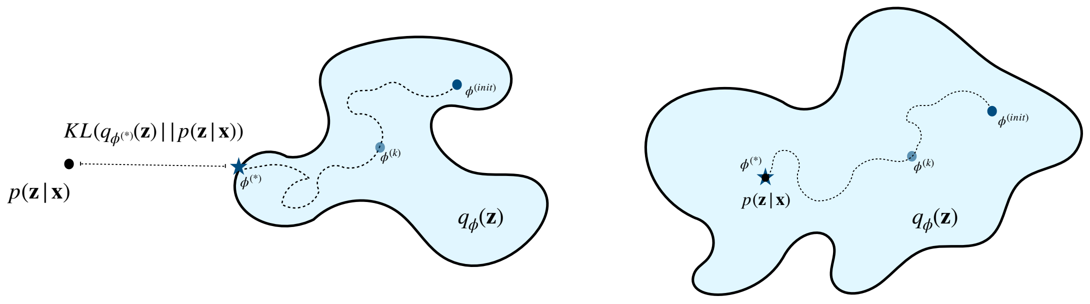
It is important to note that as the Kullback-Leibler divergence is not symmetric, and it often results in alternative projections by switching the order of the distributions, given that the number of the observations is limited. Specifically, while in $\mathcal{KL}(q||p)$, known as *information-projection* (reverse $\mathcal{KL}$), the projection focuses mainly on the most prominent mode showcasing small variance, in $\mathcal{KL}(p||q)$, known as *moment-projection* (forward $\mathcal{KL}$), it tends to extend over regions with an intermediate probability according to p demonstrating a relatively large variance. In variational inference, we use the reverse $\mathcal{KL}$, as it can result in a computationally tractable objective.

Using the definition of $\mathcal{KL}$ for continuous random variables, we can decompose the above objective as follows:

$$\begin{aligned} \mathcal{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q(\mathbf{z})} \log q(\mathbf{z}) - \mathbb{E}_{q(\mathbf{z})} \log p(\mathbf{z}|\mathbf{x}) \\ &= \mathbb{E}_{q(\mathbf{z})} \log q(\mathbf{z}) - \mathbb{E}_{q(\mathbf{z})} \log p(\mathbf{x}, \mathbf{z}) + \log p(\mathbf{x}). \end{aligned} \tag{3.2}$$

The last equality reveals that our objective depends on the marginal likelihood, making it computationally infeasible. Nonetheless, a delicate solution is to optimize an alternative objective that is equivalent to $\mathcal{KL}(q||p)$ up to an added constant value. With the observation that the $\log p(\mathbf{x})$ is constant with respect to $q(\mathbf{z})$, maximizing the summation of the negative $\mathcal{KL}(q||p)$ divergence with the $\log p(\mathbf{x})$, will be equivalent to minimizing (3.2). In consequence, we can derive a new tractable objective function, namely the *evidence lower bound* (ELBO) as:

$$\mathcal{L}(q) \equiv \mathcal{KL}(q(\mathbf{z}) \| p(\mathbf{z}|\mathbf{x})) - \log p(\mathbf{x}) = \mathbb{E}_{q(\mathbf{z})} \log p(\mathbf{x}, \mathbf{z}) - \mathbb{E}_{q(\mathbf{z})} \log q(\mathbf{z}).$$



a) Variational family does **not** include the exact posterior b) Variational family does include the exact posterior

Figure 3.1: Illustration of Variational Inference Task. The blue coloured area represents a set of variational distributions over $\mathbf{z}$ indexed by $\phi^{(i)}$. Altering the index i will result into a different setting of the variational parameters. The goal is to fit the variational parameters $\phi^{(i)}$ to be as close as possible to the exact posterior $p(z|x)$ under the $\mathcal{KL}$ divergence metric. The dashed line indicates the optimization process of the algorithm and the $\star$ indicates the optimal value given the variational family $q_\phi(\mathbf{z})$. The left figure (a) represents the case where the variational family does not include the ground truth posterior and creates a gap in the approximation with the marginal equal to $\mathcal{KL}(q_{\phi^\star(z)}||p((z)|(x)))$. In contrast, the figure on the right (b) shows the case where the variational family includes the true posterior and we are able to retrieve it, resulting into the exact approximation of the marginal likelihood $p(\mathbf{x})$.

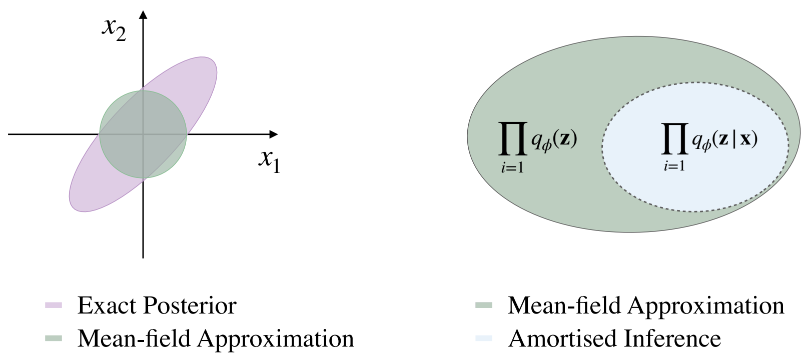
Another interpretation of the ELBO, which also explains its name, is that it is a natural lower bound to the (logarithm) evidence $\log p(\mathbf{x}) \geq \mathcal{L}(q)$ for any $q(\mathbf{z})$. We retrieve this bound from the inequality $\mathcal{KL} \geq 0$, which is equal to zero *if and only if* the two distributions match almost everywhere, meaning that we managed to discover the true posterior with the variational one, and thus, the ELBO would be equal to the marginal likelihood. Therefore, we write the above as follows:

$$\log p(\mathbf{x}) \geq \mathbb{E}_{\,q(\mathbf{z})}\log p(\mathbf{x},\mathbf{z}) - \mathbb{E}_{\,q(\mathbf{z})}\log q(\mathbf{z}) = \mathcal{L}(q).$$

It is worth it to note that a second possible approach to derive the above relationship is through Jensen's inequality (Appendix A.1). This relationship between the ELBO and the marginal log-likelihood has led to the use of the variational bound as a model selection criterion. Interestingly, if we examine the ELBO by re-writing its components, we will have

$$\begin{aligned}\mathcal{L}(q) &= \mathbb{E}_{q(\mathbf{z})}\log p(\mathbf{x}|\mathbf{z}) + \mathbb{E}_{q(\mathbf{z})}\log p(\mathbf{z}) - \mathbb{E}_{q(\mathbf{z})}\log q(\mathbf{z}) \\ &= \mathbb{E}_{\,q(\mathbf{z})}\log p(\mathbf{x}|\mathbf{z}) - \mathcal{KL}(q(\mathbf{z})||p(\mathbf{z})). \end{aligned} \tag{3.3}$$

The first term is the expected likelihood (*reconstruction loss*) and it encourages densities that place their mass on configurations of the latent variables which explain the observed data. The $\mathcal{KL}(\cdot)$ that forms the second term of the equation, which is known as the *regularisation loss*, pushes the variational posterior to be close to the prior distribution. Intuitively, the variational objective mirrors the usual balance in Bayesian learning between likelihood and prior, where the latter acts as a natural regulariser during the learning process. Looking closely into the ELBO, one can notice that the regularisation loss forces the variational posterior to be stochastic; even though reducing the variance to 0 would result into an easier optimization process, the $\mathcal{KL}$ will drive the model to always introduce an amount of noise to the samples, because otherwise, the regularisation loss would be infinite.



a) Mean-field Approximation b) Amortised Inference Approximation

Figure 3.2: Left (a): Illustration of Mean-field Approximation on true posterior. VI fits the approximation "inside" the posterior, to avoid putting probability mass for $p(D) \approx 0$, and hence always underestimates variance. The use of Mean-field approximation makes the issue worse for sparse data points, as it will focus only on a small area of the ground-truth distribution. Right (b): The amortised family shrinks the class of variational approximations even further, in favour of computational speed. The amount of decrease depends on the complexity of the function that defines the parametric way that the observable variable $\mathbf{x}$ and parameters ϕ turned into the latent code z.

2 Scaling up the Variational Inference: Variational Amortised Inference

In the Variational Inference framework, the main challenge is to choose the variational family $\mathcal{Q}$ to be flexible enough to approximate the true posterior, but also simple enough for efficient optimisation, even for high-dimensional data (Figure 3.1). One step towards a simpler family of distributions is to focus on the *mean-field variational* family, where it assumes that the distribution over the latent variables factorises into a product of lower-dimensional terms. This will ensure that we can compute various expectations over $q(\mathbf{z})$. A generic member of the mean-field variational family will be:

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j\left(z_j\right).$$

Notice that we have not specified the parametric form of the individual variational factors, where in principle, each one can take any desired form. For example, for a continuous variable, we may use a Gaussian factor and for a discrete a categorical factor. Nonetheless, even though each observation $\mathbf{x}$ has its variational distribution $q(\mathbf{z})$, which does not connect with the latent density of any other observation in any way, this approach is still intractable. Another difficulty of performing maximum likelihood estimation in deep latent variable models is that because the variational posterior $q(\mathbf{z})$ does not depend directly on the observations $\mathbf{X}$, it must approximate the whole dataset jointly. This is problematic because since it will require different training parameters for each $\mathbf{x}$, it will introduce a rather large model and impose a challenge to obtain them for unseen observations.

Alternatively, by assuming that all variational densities follow the same family of distributions, but they are dependant on $\mathbf{x}$ with parameters ϕ, we can impose a connection between the variational distributions and the individual observations $\mathbf{x}$. In other words, each variational distribution q of each $\mathbf{x}$ will be different but share the same reparametrization. Sharing the variational parameters across data points will result in a more restricted, but more flexible variational family, called *amortised variational family*. Formally, by introducing a parametric *inference model* $q_\phi(\mathbf{z}_n|\mathbf{x}_n)$ which is optimized such that $q_\phi(\mathbf{z}_n|\mathbf{x}_n) \approx p_\theta(\mathbf{z}_n|\mathbf{x}_n)$, the objective function (3.3) can be written as follows:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \mathcal{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]. \quad (3.4)$$

The use of amortized variational inference makes the deep latent variable models an attractive solution for generative modelling tasks, as it incorporates both encoder and decoder, scalable to high-dimensional data, and a parallelizable architecture for both inference and synthesis. However, we have to emphasize that the amortized approach is a statistical and computational trade-off; the amortised family admits a smaller class of approximations but provides a computationally faster framework. The size of this decrease depends on the flexibility of the parameters θ (Figure 3.2).

Given that the described framework incorporates a variational inference optimization scheme while using both an encoder and inference (decoder) model, it is known as *Variational Auto-Encoder* (VAE). The VAE can be interpreted as a direct graphical model, as shown in Figure 3.3. However, despite their intuitively simple form, their optimization process is not straightforward.

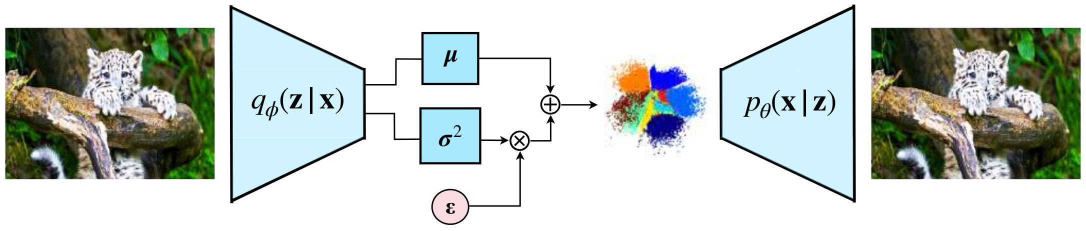

$q_\phi(\mathbf{z}|\mathbf{x})$
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}^2$
$\boldsymbol{\varepsilon}$
$p_\theta(\mathbf{x}|\mathbf{z})$

Figure 3.3: Illustration of Variational Auto-Encoder as a visual representation.

3 Optimization of VAEs

Let $p^*_X(\mathbf{x})$ be from the empirical distribution of our observations $\{\mathbf{x}_n\}_{n=1}^N$. In VAEs setting, we aim at minimizing the objective function (3.4) concerning the reconstruction loss and the regularization loss. While in the latter case the $\mathcal{KL}$ divergence can often be integrated analytically as the probability densities of the variational posterior and prior are tractable (the derivation of the closed-form solution is available in Appendix A.2), the former case is challenging. Similar to the Expectation-Maximization (EM) algorithm, we have to optimize the likelihood function with respect to both θ and ϕ with stochastic gradient descent, as we are employing neural networks as parametric models.

Deriving (3.4) with respect to θ, we will have

$$\begin{aligned}
\nabla_\theta \sum_{n=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)} \left[\log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\right] &= \nabla_\theta \sum_{n=1}^N \int q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right) \ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^N \nabla_\theta \int q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right) \ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^N \int \nabla_\theta \, q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right) \ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^N \int q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \nabla_\theta \ \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right) \ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^N \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)} \nabla_\theta \ \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right) \ \mathrm{d}\mathbf{z}_n \\
&\approx \sum_{n=1}^N \nabla_\theta \ \log p_\theta\left(\mathbf{x}_n|\hat{\mathbf{z}}_n\right) \ , \quad \hat{\mathbf{z}}_n \sim q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right)
\end{aligned}$$

and thus approximate the expected value of the variational posterior with sampling, where on average the stochastic approximation of this gradient will be correct. In essence, we showed that the gradient for θ of the expectation of the objective is equal to the expectation of the gradient.

In the same manner, for the variational parameters ϕ we will have:

$$\begin{aligned}
\nabla_\phi \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)} \left[\log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\right] &= \nabla_\phi \sum_{n=1}^{N} \int q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^{N} \nabla_\phi \int q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^{N} \int \nabla_\phi\, q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\ \mathrm{d}\mathbf{z}_n
\end{aligned} \tag{3.5}$$

However, in contrast with the previous case, the integral does not have any form of an expectation, meaning that the Monte-Carlo approximation can not be used for estimation. Additionally, as the integral is typically unknown, computing the gradient is not straightforward. Nonetheless, we can tackle this issue by taking advantage of simple mathematical techniques. Two popular methods are the *REINFORCE gradients* (also known as Monte-Carlo policy gradient and score function estimator) (Glynn, 1990) and the *reparameterization trick* (Kingma and Welling, 2013).

REINFORCE

By leveraging the *log-derivative trick*, we can compose a distribution outside of the gradient. Given that $\nabla \log g(\phi) = \frac{\nabla g(\phi)}{g(\phi)}$, we can re-write the above as follows:

$$\begin{aligned}
(3.5) &= \sum_{n=1}^{N} \int \frac{q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right)}{q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right)} \nabla_\phi\, q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^{N} \int q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \nabla_\phi\ \log q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\ \mathrm{d}\mathbf{z}_n \\
&= \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)} \nabla_\phi\ \log q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right) \log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)
\end{aligned}$$

The above method for computing the gradient of an expectation of a smooth function is known as *REINFORCE*. Now we can sample from the variational posterior and approximate the gradient of ϕ with Monte Carlo estimation. However, even though this approximation is unbiased, the estimations, and especially the initial ones, will suffer from high variance. To understand the reason behind it, we should focus on the $\log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)$ term. At the start of the training process, as the model will be randomly initialized, it will allocate low probability mass to every observation and as a result, calculating the logarithm of this value will lead to some very small proportions. Importantly, because the gradient $\nabla_\phi\ \log q_\phi\left(\mathbf{z}_n|\mathbf{x}_n\right)$ can take positive and negative values, with the Monte Carlo approximation on a batch of samples we will get big oscillations between the values (high absolute values but of different signs). Thus, even though that on average the estimation would be correct, it will require lots of samples to approximate it accurately, making it impractical for our purposes. A simple approach to obtain unbiased gradients for continuous $\mathbf{z}$ while reducing the variance of the estimations dramatically is through a change of variables, called the *reparameterization trick*.

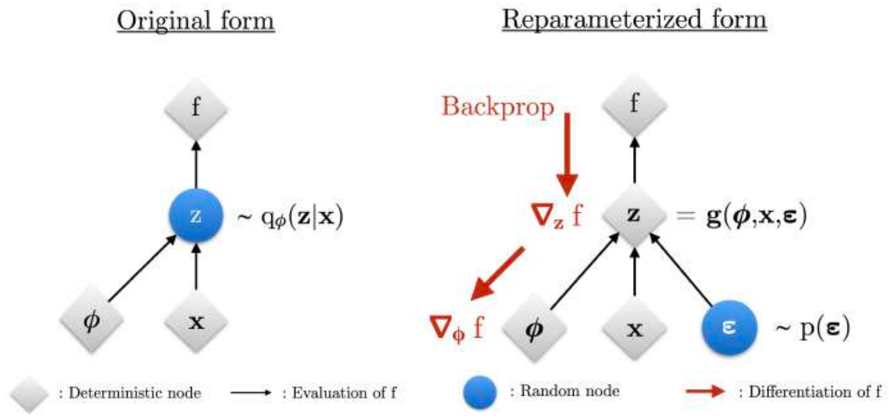


Figure 3.4: Illustration of the reparameterization trick (Kingma and Welling, 2019).

The Reparameterization Trick

Lets recall that the variational parameters ϕ affect the objective function through the random variable $\mathbf{z}_n \sim q_\phi(\mathbf{z}_n|\mathbf{x}_n)$. However, in this configuration we are not able to compute the gradients with respect to ϕ, as we cannot directly back-propagate the gradients through the random variable $\mathbf{z}$. The core idea is to turn $\mathbf{z}$ into a deterministic differentiable function of $\{\phi, \mathbf{x}\}$ and express its randomness through an independent of $\mathbf{z}, \phi$ random variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$. This allows us to back-propagate the gradients through $\mathbf{z}$, compute gradients $\nabla_\phi \mathcal{L}(\theta, \phi)$ but also express it as a random variable by injecting noise from $\boldsymbol{\epsilon}$. In order to achieve this, we will impose an invertible and differentiable transformation $g(\boldsymbol{\epsilon}, \mathbf{z}, \phi)$ from the distribution $p(\boldsymbol{\epsilon})$ to $p(\mathbf{z})$ so that the mass of the distribution is invariant under the change of variable. Thus, defining $\mathbf{z} = g(\boldsymbol{\epsilon}, \mathbf{z}, \phi)$, the derivative with respect to ϕ can now be written as follows:

$$\begin{aligned}\nabla_\phi \sum_{n=1}^{N} \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}\left[\log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\right] &= \sum_{n=1}^{N} \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z}_n|\mathbf{x}_n)}\left[\log p_\theta\left(\mathbf{x}_n|\mathbf{z}_n\right)\right] \\ &= \sum_{n=1}^{N} \nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon}_n)}\left[\log p_\theta\left(\mathbf{x}_n|g(\boldsymbol{\epsilon}_n, \mathbf{z}_n, \phi)\right)\right] \\ &= \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{\epsilon}_n)} \nabla_\phi \left[\log p_\theta\left(\mathbf{x}_n|g(\boldsymbol{\epsilon}_n, \mathbf{z}_n, \phi)\right)\right].\end{aligned}$$

As the gradient is now unrelated to the distribution with which we take the expectation, it can easily pass through the integral and we can obtain an unbiased estimator of the gradient by computing the expectation by Monte Carlo integration. Using reparameterization has given us a new approach for optimisation under the following simple assumptions: (i) the latent variable $\mathbf{z}$ is continuous (ii) $p(\boldsymbol{\epsilon})$ can generate samples relatively easy, and finally (iii) the $g(\cdot)$ map is differentiable. An illustration of the reparameterization is presented in Figure 3.4.

For the usual case that the amortized variational posterior is modelled with factorized Gaussian distributions, we can carry out the reparameterization trick by defining the random variable $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) = \mathcal{N}(0, 1)$ and the transformation from $\boldsymbol{\epsilon}$ to $\mathbf{z}$ as:

$$\mathbf{z}_n = g(\boldsymbol{\epsilon}_i, \mathbf{z}_i, \phi) = \mu_n + \sigma_n^2 \odot \epsilon_n.$$

4 Improving VAEs

As we saw, the training objective (3.4) consists of two parts, namely, the reconstruction and the regularization loss. However, we can expand it even further, by separating the second into two terms:

$$\mathcal{L}\left(\theta, \phi, \lambda\right) = \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) \;+\; \log p_\lambda(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]\right]. \tag{3.6}$$

where $q_\phi(\mathbf{z}|\mathbf{x})$ is the variational posterior (or the *encoder*), $p_\theta(\mathbf{x}|\mathbf{z})$ is the likelihood function (or the *decoder*) and $p_\lambda(\mathbf{z})$ is the *prior* over the latent variables, parameterized by ϕ, θ and λ respectively. We can improve the Variational Auto-Encoder and obtain a tighter lower bound, by advancing over these three elements.

Improving the Encoder

The variational inference framework requires that the intractable posterior has to be approximated by a class of known probability distributions. Thus, the plain framework of VAEs leverages the amortized mean-field approximation with factorized Gaussians, restricting the flexibility of the variational posterior for scalability and fast inference. However, this approximation is usually not sufficient for real-world data, as they often possess complex dependencies. Since the true posterior can only be recovered exactly if it happens to be in the chosen family, designing more expressive but tractable variational families is an important difficulty in variational inference. A popular strategy for obtaining richer variational posteriors efficiently is by introducing new multiple latent variables in the following schemes; using *auxiliary variables* or through a *change-of-variable* scheme.

In the change-of-variable approach, Rezende and Mohamed (2015) proposed the extension of the encoder with a sequence of invertible mappings, in order to transform a fully-factorised Gaussian into an arbitrarily complex distribution efficiently. Specifically, the encoder maps the input data into a a simple base posterior distribution, such as a diagonal Gaussian $\mathcal{N}(\mathbf{z}_0|\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$. The sample is then transformed through K flows, and the final latent stochastic variables are given by $\mathbf{z} = \mathbf{z}_K = f_K \circ ... \circ f_0(\mathbf{z}_0)$. Given a variational posterior $q_\phi(\mathbf{z}|\mathbf{x}) = q_K(\mathbf{z}|\mathbf{x})$, the variational objective can be rewritten as:

$$\begin{aligned}&\mathcal{L}\left(\theta, \phi, \lambda\right) = \\ &= \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\left[\mathbb{E}_{q_0(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\lambda(\mathbf{z}) - \log q_0(\mathbf{z}|\mathbf{x}) + \sum_{k=1}^{K}\log\left|\det\left(\frac{\partial f_k\left(\mathbf{z}_{k-1}; \lambda_k(\mathbf{x})\right)}{\partial \mathbf{z}_{k-1}}\right)\right|\right]\right],\end{aligned}$$

where λ_k are the parameters of the k-th transformation. While still exploiting the use of amortised inference (the parameters λ are dependant on $\mathbf{x}$), the encoder employed with *planar flows* resulted in better likelihood performances on high-dimensional data. However, the requirement for the invertible parametric transformations to have tractable Jacobians demands many transformations to reach acceptable performance. van den Berg et al. (2018) presented a more generalized and improved framework, where the one-neuron bottleneck of a flow that was proposed, named *planar-flow*, can be expanded to a vector using Sylvester's identity for matrices. On the contrary, Kingma et al. (2016) proposed autoregressive flows with parameters that are independent of the datapoint, and thus not taking advantage of the amortised inference framework. Variational inference with normalizing flows showcased clear improvements in performance and applicability.

However, the framework of variational inference can be further extended with auxiliary variables (Salimans et al., 2014; Maaløe et al., 2016) additional latent variables are used in a hierarchical order to make the posterior more flexible. These models are known as *hierarchical models*, and in contrast with the *change-of-variables* approach, the latent variables are stochastic instead of deterministic. Even though the introduction of auxiliary variables, defined as ω, leave the plain model unchanged, they have the ability to capture the structure of correlated variables, as they turn the variational posterior into a mixture of distributions $q(\mathbf{z}|\mathbf{x}, \omega)$. Thus, by varying ω, we are able to vary the mixture and adapt to the kind of dependency structure that is available in our posterior. Formally, we have:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \int q_\phi(\omega, \mathbf{z}|\mathbf{x})d\omega = \int q_\phi(\omega|\mathbf{x})q_\phi(\mathbf{z}|\omega, \mathbf{x})d\omega \qquad \text{and} \qquad p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}, \omega) = p_{\boldsymbol{\theta}}(\omega|\mathbf{x}, \mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}).$$

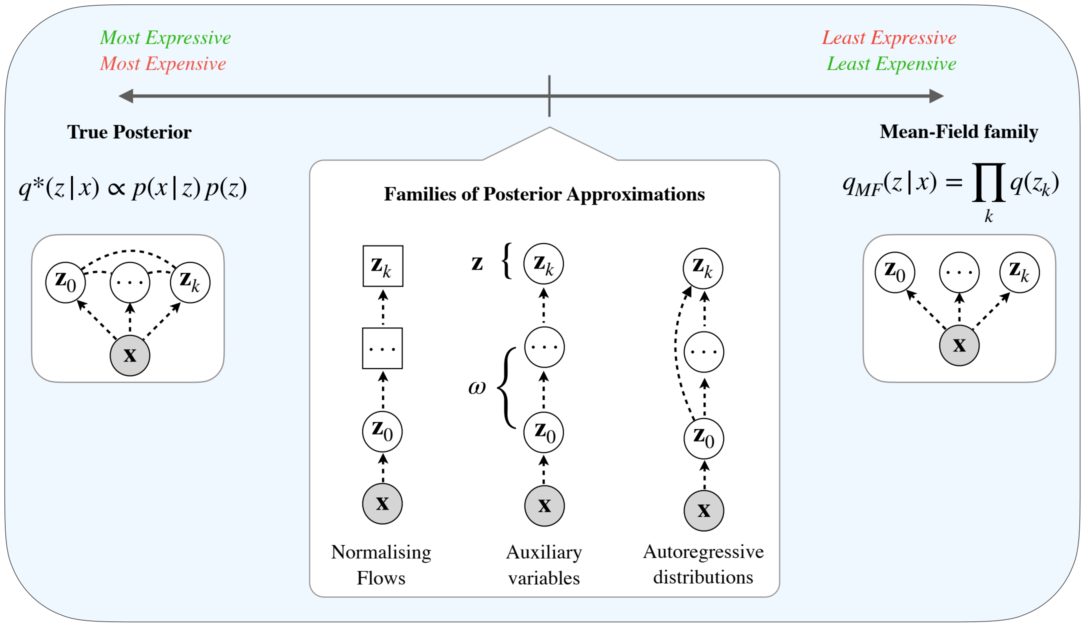


Figure 3.5: Families of Posterior Approximations. The choice of the posterior introduces a trade-off between efficiency and flexibility of the approximation task.

For tractability, hierarchical models optimise an alternative lower bound, namely the *auxiliary variational bound*:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{q_\phi(\omega,\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x},\mathbf{z}) + \log r(\omega|\mathbf{z},\mathbf{x})] - \mathbb{E}_{q_\phi(\omega,z|\mathbf{x})}[\log q(\mathbf{z}|\mathbf{x})]$$

which is an even lower bound to the marginal likelihood than without the auxiliary variable:

$$\mathcal{KL}\left(q_\phi(\mathbf{x},\mathbf{z},\omega) \| p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z},\mathbf{u})\right) \geq \mathcal{KL}\left(q_\phi(\mathbf{x},\mathbf{z}) \| p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\right).$$

Counterintuitively, because now the inference model can fit more complex latent distributions, it can improve the variational lower bound, potentially outweighing the additional cost on the objective function. Thus we receive a flexible and powerful model with easy sampling, evaluation of bound and gradients. In the same manner, one can introduce structural dependencies on the auxiliary variable that are related to the task at hand. A noteworthy example would be an autoregressive structured distribution for sequential data, like video frames. Maaløe et al. (2019) have shown that indeed that deep hierarchy of stochastic variables can lead to significant improvements resulting in competitive performance in image density estimation. The trade-off between efficiency and flexibility of the posteriors is depicted in Figure (3.5).

Improving the Prior

Even though the lower bound (Eq. 3.6) suggests that the prior plays a crucial role in improving the variational bounds, usually a not data-dependant prior is chosen in advance (e.g. follows a standard normal distribution). While being relatively simple and cheap, a fixed prior is known to result in over-regularized models that tend to ignore more of the latent dimensions (Burda et al., 2015). Moreover, as the objective function through the regularization loss is optimised to match the variational posterior with the prior, Rosca et al. (2018) argued that even powerful posteriors may still fail to match the optimal prior, namely the *aggregated posterior*, to a unit Gaussian distribution. The first evidence that a learnable prior can help to minimize the objective function was demonstrated by Hoffman and Johnson (2016). Tomczak and Welling (2017), while presenting an overview

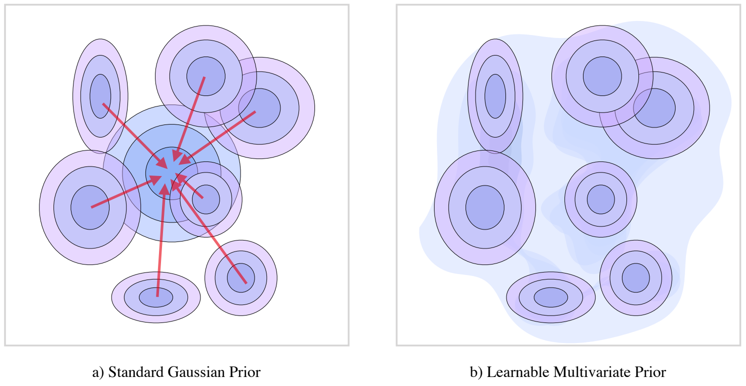


Figure 3.6: Left: The standard prior is too strong and over-regularizes the encoder and create "holes" in the latent space. Right: Learnable prior adjusts to the posterior distribution, surrounding the variable posterior with smooth transitions.

of the advantages over the use of fixed distributions, showed that the optimal prior, called the *aggregated posterior prior*, is the expectation of the encoder over the data distribution. Moreover, they suggest a finite set of learnable pseudo-inputs to approximate the optimal prior by assembling a mixture of *variational* posteriors. Thus, richer, multinomial and data-dependant priors will result in better performance both in terms of higher likelihood but also better generations.

Improving the Decoder

The generative part of the model (the decoder), aims to relate the latent variables to the observations through the likelihood function $p(\mathbf{x}|\mathbf{z})$. Specifically, it is optimised to reconstruct the input data while given their latent codes as precisely as possible. Due to the limited expressivity of a unimodal normal distribution, especially using mean square error for highly complex data like natural images, various extensions have been proposed. In general, as VAEs can effectively model the global structure of the observations, but have difficulty in capturing high-quality local representations (such as textures and sharp edges in images) due to the conditional independence of the output pixels, Gulrajani et al. (2016) proposed to use an autoregressive model to model the likelihood. However, due to the powerful nature of PixelCNN and because in order for the model to be trained, the input data had to be reintroduced to the decoder with a skip connection, the model chooses to bypass the latent codes and thus does not obtain meaningful low-dimensional representations. This issue is known as *posterior collapse*.

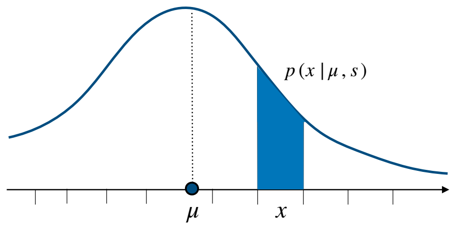


Figure 3.7: The discretized logistic distribution. The shaded area shows the probability density.

Another way to increase the flexibility of the decoder is to replace the normal distribution that describes the likelihood with a more powerful family. As images are usually encoded in 8-bit RGB colour (each channel is a uint8 integer that ranges in value from 0 to 255, inclusive), van den Oord et al. (2016b) proposed to model the likelihood of the PixelCNN with a categorical distribution in discrete space. Consequently, the distribution would be able to capture each intensity value as a discrete outcome while being factorized across all three channels. Nevertheless, this method has two important drawbacks. First, as each individual pixel is modelled with 256 categories (in uint8 case), the decoder has to project $256 \cdot 3 \cdot 32 \cdot 32 = 786,432$ total dimensions for a $3 \times 32 \times 32$ image in the last layer alone. This approach quickly becomes memory intensive with a small increase at the output image resolution or its colour depth (for example HDR images). Second, the categorical cross-entropy gradient is *sparse*, meaning that it does not provide information on how close (in pixel intensity) the prediction is to the target distribution. For example, for a target value $y = 128$, categorical cross-entropy loss does not incorporate the information that the pixel value of 127 is closer to y than it is to 0.

The issue of the sparse gradients can be solved by introducing a probability distribution for modelling RGB pixels as ordinal data. Salimans et al. (2017) proposed the use of a discrete mixture of logistics to model the pixel probability densities, as it captures the inductive bias that values close together are related. However, instead of maximizing the probability distribution function at hand under the maximum likelihood estimation, one can maximize the derivative of the cumulative distribution function (CDF). With $x \in \mathbb{Z}$, the latter can be retrieved by simply calculating the difference $\mathrm{CDF}(x + 0.5) - \mathrm{CDF}(x - 0.5)$. As the logistic's CDF distribution is the sigmoid function, the probability mass around x would be:

$$\mathrm{DLogistic}(x|\mu, s) = \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} \mathrm{Logistic}\left(x'|\mu, s\right) \mathrm{d}x' = \sigma\left(\frac{x + \frac{1}{2} - \mu}{s}\right) - \sigma\left(\frac{x - \frac{1}{2} - \mu}{s}\right).$$

Because the discretized logistic distribution is unimodal and therefore limited in complexity, with a minor increase in computational cost, we can increase the flexibility of the distribution by extending it to a mixture of logistic distributions. Moreover, as the colour channels of an image are highly correlated, coefficients were introduced to capture their linear dependencies. Thus, now for each pixel, the number of parameters would be 10 times the number of mixture components; for 32×32 images, the network outputs 102.400 dimensions, a sevenfold reduction in training parameters while enabling denser gradients and lower memory.

Chapter 4

Flow-based Generative Networks

"Those who know, do. Those who understand, teach."

— Aristotle

In contrast with the Variational Auto-Encoders, where the integral of the marginal likelihood is approximated by a lower bound, *flow-based networks* manage to compute the exact likelihood of high-dimensional densities by exploiting the change-of-variables formula. Its framework allows for efficient inference and synthesis, but also provide a useful latent space for downstream tasks. This chapter presents the theoretical background of flow-based generative networks, and specifically, focuses to a method known as Real-valued Non-Volume Preserving (RealNVP) (Dinh et al., 2016), which will be proven to be an essential building block of our proposed method.

1 Normalizing Flows

As explained in the previous chapter, in density estimation and generative modelling tasks we wish to estimate an unknown data distribution $p(\mathbf{x})$ from which we have drawn N samples $\mathbf{x} \in \mathbb{R}^{\mathrm{D}}$. A relatively simple yet powerful approach is to use flow-based networks. In flow-based modelling, the marginal can be expressed as a transformation $f : \mathrm{Z} \to \mathrm{X}$ of a simple distribution $p_Z(\mathbf{z})$, called *base distribution*, which is characterised by its straightforward process to sample from and evaluate densities.

Importantly, for flow models to be tractable, f must be restricted to a *diffeomorphism*; an invertible (bijective) and differentiable (smooth) function. As a corollary, f must operate between spaces of the same dimensionality, meaning that x and z must have an equal number of dimensions. Given these constraints, the marginal $p(\mathbf{x})$ is well-defined and computable from $p(\mathbf{z})$ by leveraging the change-of-variables theorem. Analytically, it states that one can evaluate the density of an observation $x \sim p(x)$ by computing the product of (i) the density of the transformed observation via f under the base distribution and (ii) the associated change in volume induced by the inverse transformation. Formally, in order to reach this conclusion, we will start from the very definition of probability distributions which states that the area under any pdf curve has to be equal to one. In mathematical terms, we have that:

$$\int_{\mathbb{R}^{\mathrm{D}}} p_x(\mathbf{x})\mathrm{d}\mathbf{x} = 1 = \int_{\mathbb{R}^{\mathrm{D}}} p_z(\mathbf{z})\mathrm{d}z.$$

However, for an infinitesimal neighbourhood $\mathrm{d}\mathbf{z}$ around $\mathbf{z}$ that f maps to an infinitesimal neighbourhood $\mathrm{d}\mathbf{x}$ around $\mathbf{x}$, the probability mass must be preserved in order for the above to hold locally. Thus, it would be

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) \left|\det \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\mathbf{x}}\right| = p_Z(f^{-1}(\mathbf{x})) \left|\det(\frac{\mathrm{d}f^{-1}(\mathbf{x})}{\mathrm{d}\mathbf{x}})\right|$$

We can convert the above equation to be a function of $\mathbf{z}$ so that we can perform inference with the base distribution. We have:

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) \left|\det \frac{\mathrm{d}\mathbf{z}}{\mathrm{d}\mathbf{x}}\right| \overset{1}{=} p_Z(\mathbf{z}) \left|\det(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}\mathbf{z}})^{-1}\right| = p_Z(\mathbf{z}) \left|\det(\frac{\mathrm{d}f(\mathbf{z})}{\mathrm{d}\mathbf{z}})^{-1}\right| \overset{2}{=} p_Z(\mathbf{z}) \left|\det(\frac{\mathrm{d}f(\mathbf{z})}{\mathrm{d}\mathbf{z}})\right|^{-1}$$

and hence, overall we can express the marginal likelihood as follows

$$p_X(\mathbf{x}) = p_Z(\mathbf{z}) \left|\det J_f(\mathbf{z})\right|^{-1}$$

where $J_f(\mathbf{z})$ denotes the Jacobian of f at z, that is the $D \times D$-matrix of first-order partial derivatives

$$J_f(z) = \frac{\partial f(z)}{\partial z} = \begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial z_1} & \cdots & \frac{\partial f_D}{\partial z_D} \end{pmatrix}.$$

This transformation of the density p_Z to p_X in an invertible manner is known as a *normalizing flow*. The word *flow* refers to the trajectories that a collection of samples from p_Z move along as the transformation f is applied to it. The modifier *normalizing* emerges from the fact that the density p_X is re-normalized after the transformation f by its Jacobian determinant $J_f(\mathbf{z})$, in order to output a well-defined probability distribution.

An attractive property of diffeomorphisms is that they form a group under composition, meaning that the successive application of multiple diffeomorphisms always results in a new transformation that is itself a diffeomorphism. For instance, given the transformations f_1 and f_2, the inverse and the Jacobian determinant of their (diffeomorphism) composition $f_2 \circ f_1$ are

$$\begin{gathered}(f_2 \circ f_1)^{-1} = f_1^{-1} \circ f_2^{-1} \\ \det J_{f_2 \circ f_1}(\boldsymbol{z}) = \det J_{f_2}\left(f_1(\boldsymbol{z})\right) \cdot \det J_{f_1}(\boldsymbol{z}).\end{gathered}$$

Thus, given $f_1, ..., f_L$ a total of L sequential transformations to be diffeomorphisms with $f_i(\mathbf{z}_{i-1}) = \mathbf{z}_i$, $\mathbf{z}_0 = \mathbf{z}$ and $\mathbf{z}_L = \mathbf{x}$, by retaining a normalized probability distribution at every step, we are able to flow through a chain of transformations and ultimately obtain a probability distribution of the final target variable. By leveraging the change of variable between two intermediate steps with $z_i \sim p_i(z_i)$ and $z_{i-1} \sim p_{i-1}(z_{i-1})$ that are connected from the transformation $f_i(\mathbf{z}_{i-1}) = \mathbf{z}_i \iff \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i)$, we have

$$\begin{aligned} p_i(\mathbf{z}_i) &= p_{i-1}(f^{-1}(\mathbf{z}_i)) \left|\det J_{f_i}(\mathbf{z}_{i-1})\right|^{-1} \\ \iff \log p_i(\mathbf{z}_i) &= \log p_{i-1}(\mathbf{z}_{i-1}) - \log\left|\det \frac{\partial f_i(\mathbf{z}_{i-1})}{\partial z_{i-1}}\right|. \end{aligned}$$

It is trivial now to generalise the above for a sequence of transformations and generate a normalizing flow as follows:

$$\begin{aligned} \log p_X(\mathbf{x}) = \log p_L(\mathbf{z}_L) &= \log p_{L-1}(\mathbf{z}_{L-1}) - \log\left|\det \frac{\partial f_L(\mathbf{z}_{L-1})}{\partial z_{L-1}}\right| \\ &= \log p_{L-2}(\mathbf{z}_{L-2}) - \log\left|\det \frac{\partial f_{L-1}(\mathbf{z}_{L-2})}{\partial z_{L-2}}\right| - \log\left|\det \frac{\partial f_L(\mathbf{z}_{L-1})}{\partial z_{L-1}}\right| \\ &= ... \\ &= \log p_0(\mathbf{z}_0) - \sum_{i=1}^{L} \log\left|\det \frac{\partial f_i(\mathbf{z}_{i-1})}{\partial z_{i-1}}\right|. \end{aligned}$$

[1]From the *inverse function theorem* for a continuously differentiable f

[2]For an invertible matrix M: $\det(M)\det\left(M^{-1}\right) = \det\left(M \cdot M^{-1}\right) = \det(I) = 1$, thus $\det(M)^{-1} = \det\left(M^{-1}\right)$

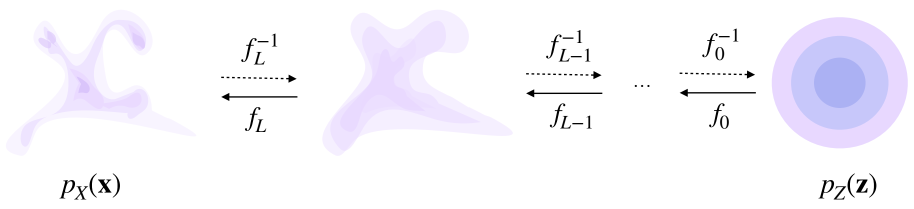


Figure 4.1: Example of an two-dimensional unimodal distribution $p_Z(\mathbf{z})$ to be transformed into a multinomial $p_X(\mathbf{x})$ by applying the change of variable formula though a sequence of functions f_i. Note that the bijective nature of f_i allows also the reverse process.

Consequently, considering the restricted form of the transformation functions, and since the distributions of interest usually lie in high-dimensional manifolds, a sequence of transformations is preferred in order to increase the flexibility of the mapping.

2 Optimization and Computational Complexity of FGNs

Density estimation & Optimization

Our goal in any probabilistic modelling task is to approximate the ground-truth distribution $p_X^*(\mathbf{x})$ with a parametric one $p_X(\mathbf{x};\theta)$, where θ are the learnable parameters, by minimizing a divergence between them. As explained in Section 3, among the many different measures of discrepancy that we can use, the Kullback–Leibler ($\mathcal{KL}$) divergence is usually the most preferred and popular one. Most importantly, minimizing the $\mathcal{KL}$ divergence results into the same optima as minimizing the negative log-likelihood (nll). Formally, we have:

$$\begin{aligned}\mathcal{KL}(p_X^*(\mathbf{x})||p_X(\mathbf{x};\boldsymbol{\theta})) &= \mathbb{E}_{p_X^*}\left[\log p_X^*(\mathbf{x}) - \log p_X(\mathbf{x};\boldsymbol{\theta})\right] \\ &= -\mathbb{E}_{p_X^*}\log p_X(\mathbf{x};\boldsymbol{\theta}) + \mathbb{E}_{p_X^*}\log p_X^*(\mathbf{x}) \\ &= \mathcal{H}(p_X^*, p_X) - \mathcal{H}(p_X^*)\end{aligned}$$

Importantly, the entropy of the observed data is constant when we minimise the above proportion with respect to θ. Thus, the exact log-likelihood $\log p_X(\mathbf{x};\boldsymbol{\theta})$ is tractable through the base function, minimizing the negative cross-entropy between true distribution and model distribution is equal with minimizing the negative log-likelihood over the training dataset. In order to calculate the density estimation of an observation $\mathbf{x}$, we will project it to $p(\mathbf{z})$ via the inverse function f^{-1} while keeping track of Jacobian determinants of the trajectories.

Thus, it would be:

$$\begin{aligned}\log p_X(\mathbf{x}) &= \log p_Z((f_L \circ \cdots \circ f_1)^{-1}(\mathbf{x})) + \log\left|\det J_{(f_L\circ\cdots\circ f_1)^{-1}}(\boldsymbol{x})\right| \\ &= \log p_Z(f_1^{-1} \circ \cdots \circ f_L^{-1}(\mathbf{x})) + \sum_{i=1}^{L}\log\left|\det J_{f_i^{-1}}(\boldsymbol{z}_i)\right| \\ &= \log p_Z(\mathbf{z}_0) + \sum_{i=1}^{L}\log\left|\det J_{f_i^{-1}}(\boldsymbol{z}_i)\right|.\end{aligned}$$

In essence, to optimize a FGN and perform density estimation at a given point, we just simply optimize an unbiased Monte Carlo estimate of $\mathcal{H}(p_X^*, p_X)$ as follows:

$$\mathcal{L}(D;\theta) = \frac{1}{|D|}\sum_{\mathbf{x}\in\mathcal{D}}\log p_X(\mathbf{x};\boldsymbol{\theta}) = \frac{1}{|D|}\sum_{\mathbf{x}\in\mathcal{D}}\left[\log p_Z\left(\mathbf{z}_0\right) + \sum_{i=1}^{L}\log\left|\det J_{f_i^{-1}}\left(\boldsymbol{z}_i\right)\right|\right].$$

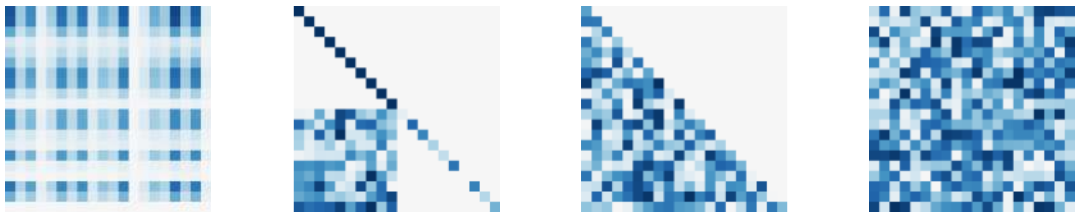


Figure 4.2: The enforced Jacobian structure of the presented categories of Flow-based Generative Networks. The square space indicates the 2-dimensional Jacobian matrix while the blue intensities the values that we keep track and manipulate. The use of more restricted transformations results into a sparse or structured Jacobian in order to omit the computation of certain values (white intensities) so that the FGN can scale to high-dimensional data (Chen et al., 2019).

Generation Process

As any other generative model, FGNs can be used also for generation of novel samples. Because the transformation function f is initialized so that it is an invertible mapping, we can use the f^{-1} to convert a sample drawn from the base distribution $p_Z(\mathbf{z})$ to the $p_X(\mathbf{x})$. Hence, sampling a point $\mathbf{z} \sim p_Z(\mathbf{z})$, we generate a point $\mathbf{x}$ by simply applying the defined series of transformations as follows:

$$\boldsymbol{x} = \boldsymbol{z}_k = f_L \circ \cdots \circ f_1\left(\boldsymbol{z}_0\right), \quad \text{where } \boldsymbol{z}_0 = \boldsymbol{z}$$

Computational Complexity

The restriction from the change of variable theorem that limits the transformations to bijective functions demands the use of a sequence of numerous transformations for modelling arbitrary distributions. However, a naive application of the change-of-variable formula will require a time cost of $\mathcal{O}\left(D^3\right)$ for computing the log determinant in every step, which quickly appears to be impractical for high-dimensional data. Consequently, it is of vital importance to develop restricted neural network architectures, which will make the computation of the Jacobian's determinant tractable. Moreover, to perform both inference and generation, the inverse f^{-1} must be also tractable and computationally feasible to evaluate. These two conditions impose a challenge in designing flexible, computationally tractable but powerful FGNs.

3 Categories of FGNs

The design choice of the neural network that parametrizes each flow form a distinguishing factor for FGNs (Figure 4.2). These approaches broadly fall into the following four categories:

- **Normalizing flows**. Normalizing flows restrict the functional form of f to exploit various determinant identities. However, as these methods do not have tractable f^{-1}, they can not be applied directly for density estimation tasks, but they find use in variational inference (Rezende and Mohamed, 2015).
- **Partitioned transformations**. By manipulating a part of the input dimensions while leaving the rest unchanged, partitioned transformations flows reduces the Jacobian of f to form a lower triangular matrix. Since the determinant of such a matrix is just the product of its diagonal elements, this results into a cheap to compute Jacobian determinant and an easy retrieval of the inverse f (Dinh et al., 2016).
- **Autoregressive transformations**. Autoregressive transformations also exploit the determinant property of a lower triangular matrix. However, the Jacobian of the transformation is enforced to be lower triangular by specifying the ordering of the dimensions by using an autoregressive model (Kingma et al., 2016).

- **Free-Form transformations**. While taking advantage of the recent developments in neural ordinary differential equations, free-form continuous dynamic transformations allow completely unrestricted architectures (Chen et al., 2019).

In this thesis, we will focus on the partitioned transformations flows, as they exhibit the best likelihood estimations on high dimensional data while their training, inference and sampling speed remains relatively high among the others. Specifically, we will focus on the method known as RealNVP.

4 Partitioned transformation flows

RealNVP Partitioned transformations were first introduced with the *Real-valued Non-Volume Preserving* transformations (RealNVP; Dinh et al. (2016)). RealNVP make use of the *affine coupling layer* (ACL), a bijection $f_{\text{ACL}} : Z \to X$ which splits the input dimensions into two parts:

- The first d dimensions of the input variable $\mathbf{z}$ remain unaltered.
- The remaining $D - d$ dimensions are scaled and translated under an affine transformation from the first d dimensions.

In a mathematical context, the affine coupling layer is described as follows:

$$f_{\text{ACL}} = \begin{cases} \boldsymbol{x}_{1:d} = \boldsymbol{z}_{1:d} \\ \boldsymbol{x}_{d+1:D} = \boldsymbol{z}_{d+1:D} \odot \exp\left(s\left(\boldsymbol{z}_{1:d}\right)\right) + t\left(\boldsymbol{z}_{1:d}\right) \end{cases}$$

where the functions $s, t : \mathrm{I\!R}^{\mathrm{d}} \to \mathrm{I\!R}^{\mathrm{D-d}}$ describe the *scale* and *translate* mappings respectively and $\odot$ represents the element-wise Hadamard product. Importantly, this partitioned transformation of the input satisfies both the desired computational properties of a flow transformation. In particular, the inverse mapping f_{ACL}^{-1} has the same complexity cost as the forward propagation f_{ACL}

$$f_{\text{ACL}}^{-1} = \begin{cases} \boldsymbol{z}_{1:d} = \boldsymbol{x}_{1:d} \\ \boldsymbol{z}_{d+1:D} = \left(\boldsymbol{x}_{d+1:D} - t\left(\boldsymbol{x}_{1:d}\right)\right) \oslash \exp\left(s\left(\boldsymbol{x}_{1:d}\right)\right) \end{cases}$$

with $\oslash$ denoting Hadamard division.

Moreover, due to the careful design of f_{ACL}, the computation of the Jacobian determinant is extremely simple. The Jacobian matrix of f_{ACL} will be:

$$\boldsymbol{J} = J_f(\mathbf{z}) = \frac{\partial f_{\text{ACL}}(\mathbf{z})}{\partial \mathbf{z}} = \begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial z_1} & \cdots & \frac{\partial f_D}{\partial z_D} \end{pmatrix} = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial f_D}{\partial z_1} & \cdots & \exp\left(s\left(\boldsymbol{x}_{1:d}\right)\right)_D \end{pmatrix}$$

which is a lower triangular matrix. By taking advantage that the determinant of a triangular matrix can be efficiently computed as the product of its diagonal terms, we write that

$$\det\left(\boldsymbol{J}_{\text{ACL}}\right) = \prod_{j=1}^{D-d} \exp\left(S\left(\boldsymbol{z}_{1:d}\right)\right)_j = \exp\left(\sum_{j=1}^{D-d} S\left(\boldsymbol{z}_{1:d}\right)_j\right)$$

The design choice of ACL to leave the first d dimensions unchanged leads to a tractable Jacobian and easy computation of f^{-1} with the trade-off that it limits the flexibility of the model remarkably. To this end, a

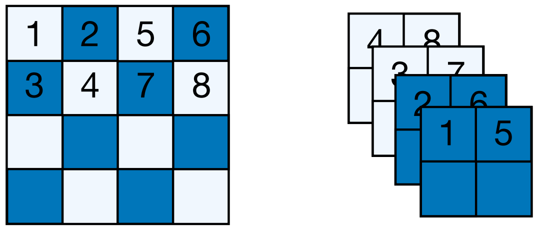


Figure 4.3: Masking schemes for affine coupling layers. Left: the spatial checkerboard pattern. Right: the channel-wise masking.

trivial solution is the composition of coupling layers in an alternating pattern, such that the components that are left unchanged in one coupling layer are updated in the next. Moreover, to exploit the local correlation structure of the data like images, instead of splitting the dimensions of the input data in half by their natural order (pixel order: from top left to bottom right), we can employ a *spatial checkerboard* selection pattern or *channel-wise masking* (Figure 4.3). Importantly, the affine coupling layer allows for arbitrarily complex scale and translate functions, since f_{ACL}^{-1} does not require computing the inverse functions of $s(\cdot)$ and $t(\cdot)$, and the Jacobian determinant of the input transformation does not involve computing their respective Jacobian determinants J_s and J_t. Thus, we can implement them as powerful deep non-invertible neural networks, as they will increase the flexibility of the model.

Overall, RealNVP forms a well-defined generative model that is able to perform fast sample estimation and sample generation, with just a forward or reverse pass of the network respectively. Moreover, since they can estimate the exact likelihood of the data, the cost function does not rely on a handcrafted per-pixel reconstruction loss (such as a mean square error), which most likely results into more accurate reconstructions. However, since the projected space needs to have the same dimensionality as the input, and in combination with the large number of flows that are needed for a satisfactory approximation of the target density, they tend to be extremely computationally intensive. It is worth to mention that other partitioned transformation flows have been explored and have demonstrated better density estimation scores (Kingma and Dhariwal, 2018; Ho et al., 2019) by using different flow architectures and dequantization formulas.

Chapter 5

Method

"*Research is to see what everybody else has seen, and to think what nobody else has thought.*"

— Albert Szent-Gyorgyi

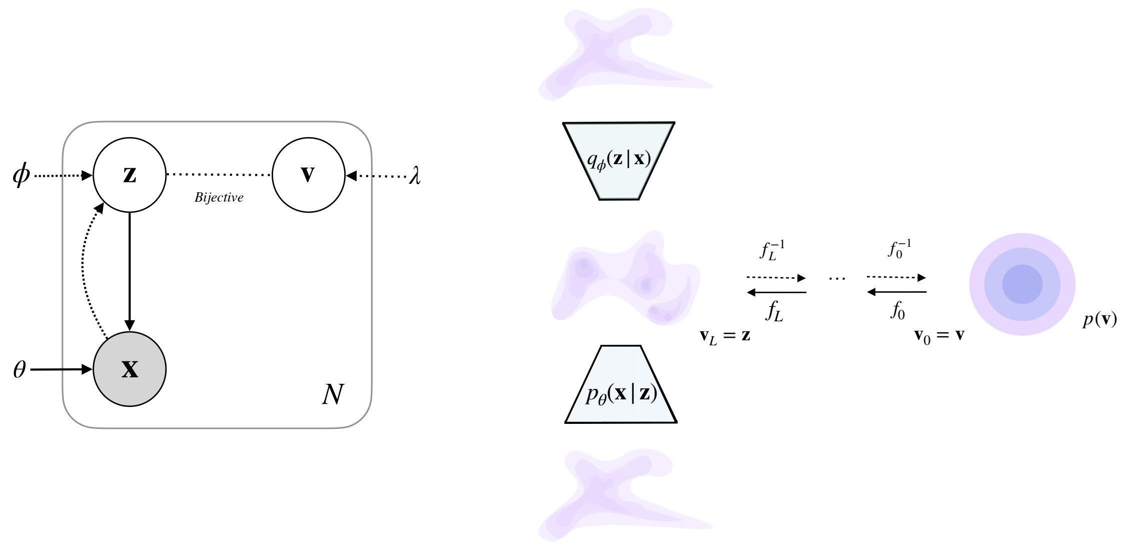
In this chapter, we present our proposed models and original contributions. First, we advance over the Variational Auto-Encoder framework by employing a bijective network as a data-driven prior. Then, we introduce the *Self-Supervised Variational Auto-Encoder*, a generative model which takes advantage of a single or multiple compressed (self-supervised) representation(s) of the input data, in order to discover gradually the underlying complex distribution of the observations. We will derive the objective function, present the additional functionalities that the proposed model offers, and illustrate various self-supervised transformations for image data. Finally, we present our novel neural network architecture for an autoencoder setting, while reasoning for our choices, in order to make an efficient and overall scalable framework.

1 VAE with bijective prior

Bijective neural networks were first introduced to increase the expressibility of the encoder, due to its forced limitations to follow a mean-field variational family of Gaussian distributions. As they demand the base function to be known and simple (i.e. a distribution that is easy to estimate and sample from), the mapping of the input space to the latent follows a two step process; first the input sample is mapped to the base distribution though the encoder, and then transformed to match the unimodal Gaussian prior. However, despite the richer variational posterior, Rosca et al. (2018) proved that the optimal prior may still not much a fixed unit Gaussian distribution, which in addition, is known to result in over-regularized models that tend to ignore more of the latent codes (Burda et al., 2015; Tomczak and Welling, 2017).

However, little attention has been paid to adapt bijective neural networks as a learnable, data-driven prior of Variational Auto-Encoders. Since the prior of the latent dimensions plays a crucial part in the objective function, it will contribute into a better likelihood estimation of the data, while providing more informative latent codes. There are a couple of advantages using a bijective network to estimate the prior distribution:

- **Arbitrary complex, data-driven prior** Instead of forcing the encoder to match a fixed distribution, we follow the intuitive process, that now, the prior distribution adapts to the posterior during the training progress. Thus, the posterior is not being dragged to the centre of a standard Gaussian distribution and it is allowed to move freely in the latent space. As the normalizing flows framework allows the transformation of a simple function to an arbitrary complex one, the result would be a richer, multi-modal distribution that incorporates causal knowledge of the data at hand.
- **Simple implementation and integration** Instead of breaking the encoder into a two step process, we fit a flow-based generative model to the latent codes produced by the encoder. Additionally,



ϕ
θ
λ
z
v
x
Bijective
N
$q_\phi(\mathbf{z}|\mathbf{x})$
$p_\theta(\mathbf{x}|\mathbf{z})$
f_L^{-1}
f_L
f_0^{-1}
f_0
$\mathbf{v}_L = \mathbf{z}$
$\mathbf{v}_0 = \mathbf{v}$
$p(\mathbf{v})$

Figure 5.1: VAE with Bijective Prior, as the name suggests, extends plain the VAE by incorporating a flow-based prior. Left: Direct graphical model of a VAE with a bijective prior. The random variable $\mathbf{u}$ indicates the base distribution of the flow model while $\mathbf{z}$ the complex prior of the VAE. Right: Visual representation of the inference and generative process. The encoder outputs a factorised gaussian distribution which is transformed into the base distribution, a standard multivariate normal density.

getting advantage of the invertible nature of the network, we can generate a datapoint by simply sampling a point from its base distribution, and transforming it (through the inverse process) into a latent code of the variational posterior.

- **Fast sampling and inference**. An alternative way of retrieving a powerful latent prior is through autoregressive modelling of the latent codes. Formally, $p(\mathbf{z})$ is factorized as

$$p(\mathbf{z}) = \prod_{i=1}^{d} p\left(z_i|\mathbf{z}_{<i}\right)$$

so that estimating $p(\mathbf{z})$ reduces to the estimation of each single conditional probability density (CPD) expressed as $p(z|\mathbf{z}_{<i})$, where the symbol $<$ implies an order over random variables. However, because high-dimensional data often requires a relatively high-dimensional latent space, in order to extract the highly informative causal components, the sampling process can be an important barrier for fast generation of novel content. On the contrary, the bijective neural networks require only a forward pass of a sampled point, reducing significantly the inference time.

It is straightforward to obtain a rich, multi-modal prior $p(\mathbf{z})$ from a simple one (i.e., standard multivariate Gaussian) by utilizing a bijective network. Formally, given a vector of latent codes $\mathbf{z} \sim q_Z(\mathbf{z}|\mathbf{x})$, a simple base distribution $p_V(\mathbf{v})$ on a latent variable $\mathbf{v} \in V$, and $f_1, ..., f_L$ a total of L sequential transformations to be diffeomorphisms with $f_i(\mathbf{v}_{i-1}) = \mathbf{v}_i$, $\mathbf{v}_0 = \mathbf{v}$ and $\mathbf{v}_L = \mathbf{z}$, the sequential use of the change of variable formula defines a model distribution on Z by:

$$\log p_Z(\mathbf{z}) = \log p_V(\mathbf{v}) - \sum_{i=1}^{L} \log \left| \det \frac{\partial f_i(\mathbf{v}_{i-1})}{\partial \mathbf{v}_{i-1}} \right| .$$

Substitute the above into (3.6), we result into the following training objective:

$$\mathcal{L}\left(\theta, \phi, \lambda\right) = \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})} \left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left(\ln p_\theta(\mathbf{x}|\mathbf{z}) + \log p_V(\mathbf{v}_0) + \sum_{i=1}^{L} \log \left| \det J_{f_i^{-1}}(\mathbf{v}_i) \right| - \log q_\phi(\mathbf{z}|\mathbf{x}) \right) \right] .$$

2 Self-Supervised Variational Auto-Encoders

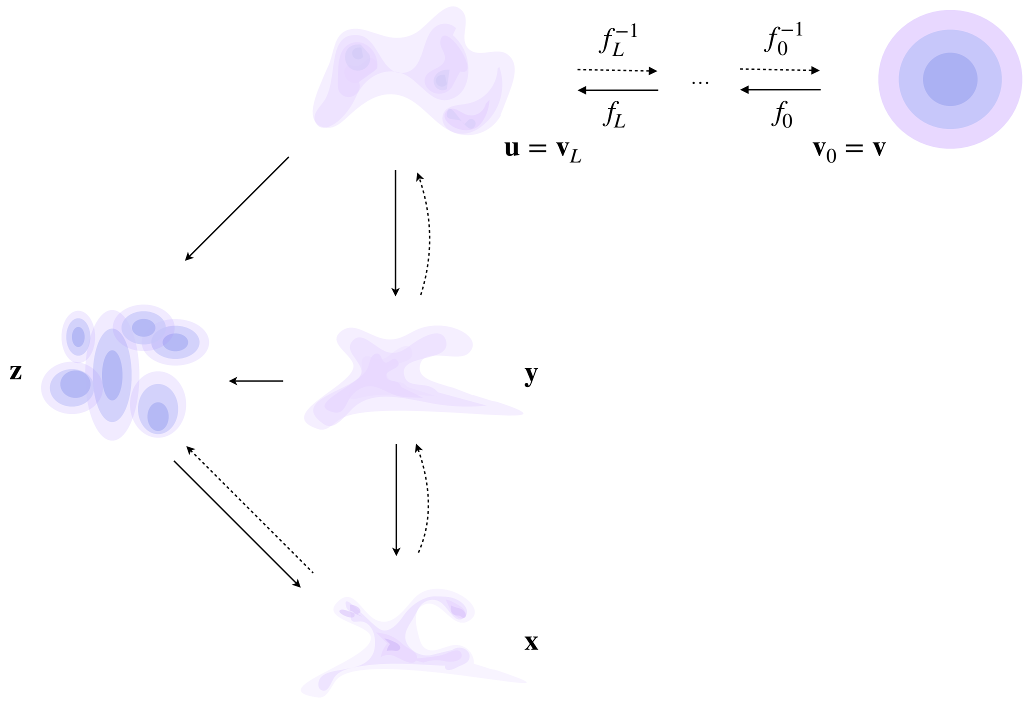


Figure 5.2: Stochastic dependencies of the proposed model. The Self-Supervised Variational Auto-Encoder takes advantage of a compressed, deterministic representation of the data $\mathbf{y}$, splitting the model's task to discover the underline complex distribution of the data into two simpler tasks. The framework models $\mathbf{y}$ with $\mathbf{u}$ through variational inference, and enhance $\mathbf{y}$ with $\mathbf{z}$ to produce $\mathbf{x}$. The generative part is denoted by solid lines and the variational part by the dashed ones.

Drawing inspiration from humans' visual learning system, we propose to expand the framework of Variational Auto-Encoders into a two-step procedure. By incorporating a compressed representation of the data, with the premise that it would be easier to approximate, the process of modelling a high-dimensional complex density breaks down into two simpler tasks. In this way, the overall expressivity of the model will grow and gradually result into richer, more concrete generations. A positive effect of the proposed framework, is that the model allows us to integrate prior knowledge through the compressed representations, without losing its unconditional generative functionality. Overall, we obtain a two-level VAE with three latent variables, where one is a data transformation that can be retrieved in a self-surprised (or supervised) fashion. This model will required an alternative optimization function from VAEs, which we will derive in the following section.

2.1 Problem Formulation and Calculation of Lower Bound

Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^{\mathrm{D}}$ be the observable data that we wish to model. Additionally, let us introduce an additional variable $\mathbf{y} \in \mathbb{R}^{\mathrm{C}}$ that is a compressed representation of $\mathbf{x}$, where $C \leq D$. Further, let $\mathbf{u} \in \mathbb{R}^{K}$ and $\mathbf{z} \in \mathbb{R}^{M}$ be two stochastic latent variables that interact with the above observed ones in a way that is presented in Figure 5.2. One can notice that even though we can impose a discriminate relationship between the compressed $(\mathbf{y})$ and the ground-truth representation $(\mathbf{x})$ of the data, we choose to employ an additional latent variable $\mathbf{z}$, in order to operate into a fully probabilistic framework and being able to modify high-level attributes of the final result.

From the dependencies of the considered probabilistic graphical model, we can write the joint probability as follows:

$$p(\mathbf{x},\mathbf{y},\mathbf{z},\mathbf{u}) = p(\mathbf{x}|\mathbf{y},\mathbf{z})\,p(\mathbf{z}|\mathbf{y},\mathbf{u})\,p(\mathbf{y}|\mathbf{u})\,p(\mathbf{u}).$$

Then, we define the amortized variational posterior of $p(\mathbf{y},\mathbf{z},\mathbf{u}|\mathbf{x})$ as follows:

$$q(\mathbf{y},\mathbf{z},\mathbf{u}|\mathbf{x}) = q(\mathbf{z}|\mathbf{y},\mathbf{x})\,q(\mathbf{u}|\mathbf{y})\,q(\mathbf{y}|\mathbf{x}) \equiv q(\mathbf{w}|\mathbf{x}),$$

where $\mathbf{w} = \{\mathbf{y},\mathbf{z},\mathbf{u}\}$, and derive the corresponding lower bound of the likelihood function in the following manner:

$$\begin{aligned}\log p(\mathbf{x}) &\geq \mathbb{E}_{q(\mathbf{w})} \log \frac{p(\mathbf{x},\mathbf{w})}{q(\mathbf{w})}\\ &= \mathbb{E}_{q(\mathbf{w})}\Big[\log p(\mathbf{x},\mathbf{w})\Big] - \mathbb{E}_{q(\mathbf{w})}\Big[\log q(\mathbf{w})\Big]\\ &\equiv \mathcal{L}(\mathbf{x}).\end{aligned} \tag{5.1}$$

Expanding the lower bound from (5.1), from the first part we will have

$$\begin{aligned}\mathbb{E}_{q(\mathbf{w})}\Big[\log p(\mathbf{x},\mathbf{w})\Big] = \mathbb{E}_{q(\mathbf{w})}\Big[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})\,p(\mathbf{z}|\mathbf{y},\mathbf{u})\,p(\mathbf{y}|\mathbf{u})\,p(\mathbf{u})\Big] = \mathbb{E}_{q(\mathbf{z}|\mathbf{y},\mathbf{x})q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})\Big]+\\ + \mathbb{E}_{q(\mathbf{z}|\mathbf{y},\mathbf{x})\,q(\mathbf{u}|\mathbf{y})\,q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{z}|\mathbf{y},\mathbf{u})\Big] + \mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{y}|\mathbf{u})\Big] + \mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{u})\Big],\end{aligned}$$

and for the second

$$\mathbb{E}_{q(\mathbf{w})}\Big[\log q(\mathbf{w})\Big] = \mathbb{E}_{q(\mathbf{z}|\mathbf{y},\mathbf{x})}\Big[\log q(\mathbf{z}|\mathbf{y},\mathbf{x})\Big] + \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\Big[\log q(\mathbf{y}|\mathbf{x})\Big] + \mathbb{E}_{q(\mathbf{u}|\mathbf{y})\,q(\mathbf{y}|\mathbf{x})}\Big[\log q(\mathbf{u}|\mathbf{y})\Big].$$

Interestingly, plugging the above terms back to (5.1) and rearranging them, we will have

$$\begin{aligned}\mathcal{L}(\mathbf{x}) \overset{(5.1)}{=} \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{y},\mathbf{x})q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{x}|\mathbf{y},\mathbf{z})\Big] + \mathbb{E}_{q(\mathbf{z}|\mathbf{y},\mathbf{x})\,q(\mathbf{u}|\mathbf{y})\,q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{z}|\mathbf{y},\mathbf{u})\Big] - \mathbb{E}_{q(\mathbf{z}|\mathbf{y},\mathbf{x})}\Big[\log q(\mathbf{z}|\mathbf{y},\mathbf{x})\Big]}_{\boldsymbol{A}} +\\ + \underbrace{\mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{y}|\mathbf{u})\Big] + \mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\Big[\log p(\mathbf{u})\Big] - \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\Big[\log q(\mathbf{y}|\mathbf{x})\Big] - \mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\Big[\log q(\mathbf{u}|\mathbf{y})\Big].}_{\boldsymbol{B}}\end{aligned}$$

Working with term $\boldsymbol{B}$, one can see that

$$\boldsymbol{B} = \mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\Big[\log \frac{p(\mathbf{y}|\mathbf{u})p(\mathbf{u})}{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\Big],$$

which denotes a (hidden) lower bound on of the marginal $\log p(\mathbf{y})$ with variational posterior $q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})$. Thus, the resulted lower bound of the marginal likelihood of $\mathbf{x}$ would be

$$\begin{aligned}\mathcal{L}(\mathbf{x}) =& \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})\,q(\mathbf{y}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) - \mathbb{E}_{q(\mathbf{u}|\mathbf{y})}\,\mathcal{KL}(q(\mathbf{y}|\mathbf{x})||p_\theta(\mathbf{y}|\mathbf{u}))+\\ &- \mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})}\,\mathcal{KL}(q(\mathbf{z}|\mathbf{x},\mathbf{y})||p(\mathbf{z}|\mathbf{y},\mathbf{u})) - \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\,\mathcal{KL}(q(\mathbf{u}|\mathbf{y})||p(\mathbf{u})),\end{aligned} \tag{5.2}$$

where $\mathcal{KL}(\cdot||\cdot)$ denotes the Kullback-Leibler divergence.

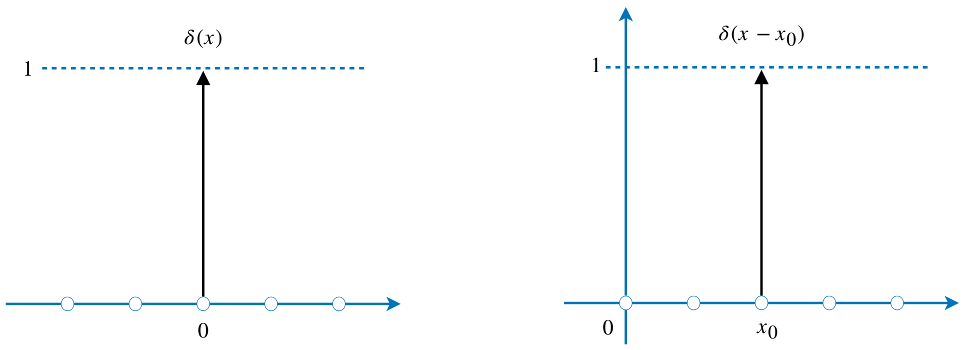


Figure 5.3: Graphical representation of delta function. The delta function, $\delta(x)$, is shown by an arrow at $x = 0$, which height is equal to 1. In the figure, we also show the function $\delta(x - x_0)$, which is the shifted version of $\delta(x)$.

2.2 Properties

There are two main properties that we are going to take advantage of.

A. If $q(\mathbf{y}|\mathbf{x})$ is both deterministic and discrete, then $\mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\Big[\log q(\mathbf{y}|\mathbf{x})\Big] = 0$.

Dependence between two random variables can take a variety of forms, of which *stochastic independence* and *functional dependence* can be argued to be most opposite in character. In the former case, neither of the variables provide any information about each other, whereas in the latter, there is a full determination. Even though the proposed framework allows to model $q(\mathbf{y}|\mathbf{x})$ as stochastic, the choice of a deterministic relationship is more attractive. This is because as the transformations to a compressed representation are usually available (e.g., a downscaled image), the optimization process would be faster, easier and the model overall will require less trainable parameters. Furthermore, this will allow us to exploit prior knowledge and control of the data generation process, giving emphasis on specific desired characteristics of the processed data. For example, if we want to focus on the outlines and shapes of the objects in an image (e.g. characteristics of person face), we can define $\mathbf{y}$ as a sketch representation of the image.

In order to deal with constant values in a probabilistic framework, we are going to make use of the *Dirac's delta* function. The Dirac's delta function represents a mapping with an infinitely sharp peak bounding unit area; a function $\delta(x)$ that has zero value everywhere except at $x = 0$, where it is infinitely large, in such a way that the total integral is sums up to 1. This property allow it to be defined as a probability distribution that is also a measure (Figure 5.3). A limiting case of a distribution whose variance goes to 0, causing the probability density function to be *delta* at x_0 is the *degenerate* distribution. The degenerate probability distribution can be defined on a discrete or a continuous space, with support only on a space of lower dimension. Formally:

Definition *Let X be a discrete random variable on a probability space. Then X has a degenerate distribution with parameter r if*

$$\Omega_X = r, \quad P(X = x_0) = \begin{cases} 1, & \text{if } x_0 = r \\ 0, & \text{if } x_0 \neq r \end{cases}$$

That is, there is only value that X can take, namely r, which it takes with certainty.

It trivially gives rise to a probability mass function satisfying $P(\Omega) = 1$ and has an expectation of a constant value $c \in \mathbb{R}$, a variance of 0 and most importantly, its entropy is also equal to 0.

Thus, modelling the deterministic transformation $q(\mathbf{y}|\mathbf{x})$ as a discrete degenerate distribution yields:

$$\mathbb{H}\big[q(\mathbf{y}|\mathbf{x})\big] = 0 \Leftrightarrow \mathbb{E}_{q(\mathbf{y}|\mathbf{x})}\big[-\log q(\mathbf{y}|\mathbf{x})\big] = 0$$

which simplifies the derived objective function in (5.2).

B. The distribution $q(\mathbf{z}|\mathbf{y},\mathbf{x})$ can be simplified to $q(\mathbf{z}|\mathbf{x})$.

One of the core motivation behind the architecture of the two staged approach, is that the latent variable $\mathbf{z}$ will be able to capture the missing information between $\mathbf{x}$ and $\mathbf{y}$. Thus, since $\mathbf{y}$ is a compressed representation of $\mathbf{x}$, it does not introduce any additional information about $\mathbf{z}$ that is not already in $\mathbf{x}$. This intuitively allows to model $\mathbf{z}$ only using $\mathbf{x}$, and, in essence, to replace $q(\mathbf{z}|\mathbf{y},\mathbf{x})$ with $q(\mathbf{z}|\mathbf{x})$, in order to further simplify the objective function.

Final ELBO

With these two properties in mind, the final lower bound of the marginal likelihood of $\mathbf{x}$ is the following:

$$\begin{aligned}\mathcal{L}(\mathbf{x}) =& \underbrace{\mathbb{E}_{q(\mathbf{z}|\mathbf{x})\,q(\mathbf{y}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})}_{\mathrm{RE}_x} + \\ &+ \underbrace{\mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})} \log p_\theta(\mathbf{y}|\mathbf{u})}_{\mathrm{RE}_y} + \\ &- \underbrace{\mathbb{E}_{q(\mathbf{u}|\mathbf{y})q(\mathbf{y}|\mathbf{x})} \mathcal{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{y},\mathbf{u}))}_{\mathrm{KL}_z} + \\ &- \underbrace{\mathbb{E}_{q(\mathbf{y}|\mathbf{x})} \mathcal{KL}(q(\mathbf{u}|\mathbf{y})||p(\mathbf{u}))}_{\mathrm{KL}_u}. \end{aligned} \tag{5.3}$$

2.3 Self-Supervised VAE

We choose the following distributions in our model:

$$\begin{aligned} q_{\phi_1}(\mathbf{u}|\mathbf{y}) &= \mathcal{N}\left(\mathbf{u}|\boldsymbol{\mu}_{\phi_1}(\mathbf{y}), \mathrm{diag}\left(\boldsymbol{\sigma}_{\phi_1}(\mathbf{y})\right)\right) \\ q(\mathbf{y}|\mathbf{x}) &= \delta(\mathbf{y} = d(\mathbf{x})) \\ q_{\phi_2}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}_{\phi_2}(\mathbf{x}), \mathrm{diag}\left(\boldsymbol{\sigma}_{\phi_2}(\mathbf{x})\right)\right) \\ p(\mathbf{v}) &= \mathcal{N}(\mathbf{v}|\mathbf{0},\mathbf{1}) \\ p_\lambda(\mathbf{u}) &= p(\mathbf{v}) \prod_{i=1}^{L} \left| \det \frac{\partial f_i(\mathbf{v}_{i-1})}{\partial \mathbf{v}_{i-1}} \right|^{-1} \\ p_{\theta_1}(\mathbf{y}|\mathbf{u}) &= \sum_{i=1}^{K} \pi_i^{(\mathbf{u})} \mathrm{Dlogistic}\left(\mu_i^{(\mathbf{u})}, s_i^{(\mathbf{u})}\right) \\ p_{\theta_2}(\mathbf{z}|\mathbf{y},\mathbf{u}) &= \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}_{\theta_2}(\mathbf{y},\mathbf{u}), \mathrm{diag}\left(\boldsymbol{\sigma}_{\theta_2}(\mathbf{y},\mathbf{u})\right)\right) \\ p_{\theta_3}(\mathbf{x}|\mathbf{z},\mathbf{y}) &= \sum_{i=1}^{K} \pi_i^{(\mathbf{z},\mathbf{y})} \mathrm{Dlogistic}\left(\mu_i^{(\mathbf{z},\mathbf{y})}, s_i^{(\mathbf{z},\mathbf{y})}\right). \end{aligned}$$

where Dlogistic is defined as the discretized logistic distribution (Salimans et al., 2017), $\delta(\cdot)$ is the Dirac's delta, and $d(\mathbf{x})$ denotes the compressed transformation that returns a discrete values.

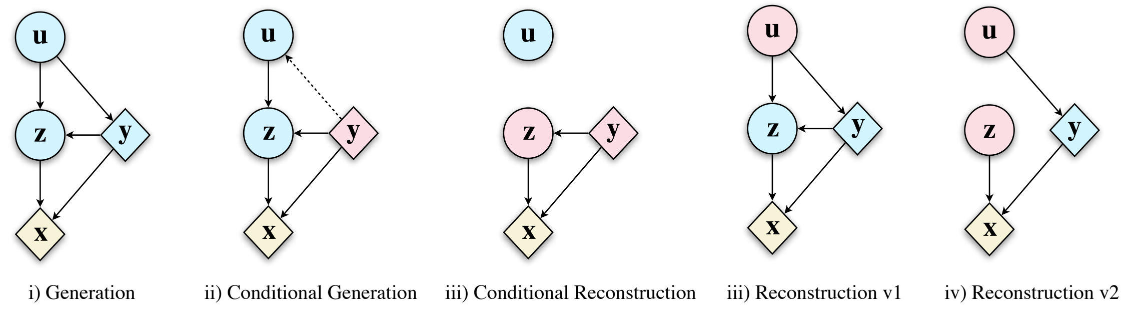


Figure 5.4: Two-leveled Self-Supervised Variational Auto-Encoder's reconstruction methods. The blue and pink nodes represent the sampled and inferred latent codes, while the circle and the rhombus stochastic and deterministic variables respectively.

In VAEs, it is possible to use the following functionalities:

- **Generation:** The model is able to generate new images through the following process:

$$\mathbf{z} \sim p(\mathbf{z}) \rightarrow \mathbf{x} \sim p(\mathbf{x}|\mathbf{z}).$$

- **Reconstruction:** The model allows to reconstruct $\mathbf{x}$ by using the following scheme:

$$\mathbf{x} \rightarrow \mathbf{z} \sim q(\mathbf{z}|\mathbf{x}) \rightarrow \mathbf{x} \sim p(\mathbf{x}|\mathbf{z}).$$

Interestingly, our approach allows more operations:

- **Generation:** The model allows to generate novel content by applying the following hierarchical sampling process:

$$\mathbf{u} \sim p(\mathbf{u}) \rightarrow \mathbf{y} \sim p(\mathbf{y}|\mathbf{u}) \rightarrow \mathbf{z} \sim p(\mathbf{z}|\mathbf{u}, \mathbf{y}) \rightarrow \mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \mathbf{y}).$$

- **Conditional Generation:** Given $\mathbf{y}$, we can sample the latent codes:

$$\mathbf{u} \sim q(\mathbf{u}|\mathbf{y}) \rightarrow \mathbf{z} \sim p(\mathbf{z}|\mathbf{y}, \mathbf{u}), \rightarrow \mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \mathbf{y}).$$

- **Conditional Reconstruction:** Similarly to standard VAE, we can reconstruct $\mathbf{x}$:

$$\mathbf{y} \sim q(\mathbf{y}|\mathbf{x}) \rightarrow \mathbf{z} \sim q(\mathbf{z}|\mathbf{x}) \rightarrow \mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \mathbf{y}).$$

- **Reconstruction:** Additionally, we can reconstruct $\mathbf{x}$ by combining the generation and the reconstruction:

$$\mathbf{y}^* \sim q(\mathbf{y}^*|\mathbf{x}) \rightarrow \mathbf{u} \sim q(\mathbf{u}|\mathbf{y}^*) \rightarrow \mathbf{y} \sim p(\mathbf{y}|\mathbf{u}) \rightarrow \mathbf{z} \sim p(\mathbf{z}|\mathbf{y}, \mathbf{u}), \rightarrow \mathbf{x} \sim p(\mathbf{x}|\mathbf{z}, \mathbf{y}).$$

The sampling processes of the above operations are illustrated in Figure 5.4. In order to highlight the enrichment of the generation process in our model through a self-supervised fashion, we refer to it as the *Self-Supervised Variational Auto-Encoder* (ssVAE). In the special case for image data where the compressed representation is a down-sampled transformation, the framework is also known as *Super-Resolution Variational Auto-Encoder* (srVAE) (Gatopoulos et al., 2020).

2.4 Hierarchical Self-Supervised VAE

The proposed architecture can be easily extended by introducing more representations, in the way that it is illustrated in Figure 5.5. This can be extremely helpful when dealing with high-dimensional images, where we downscale the input sample multiple times in order to effectively upscale the resolution sequentially, and thus introduce successively higher quality to the end result. Formally, the general case of 5.3 for K self-supervised

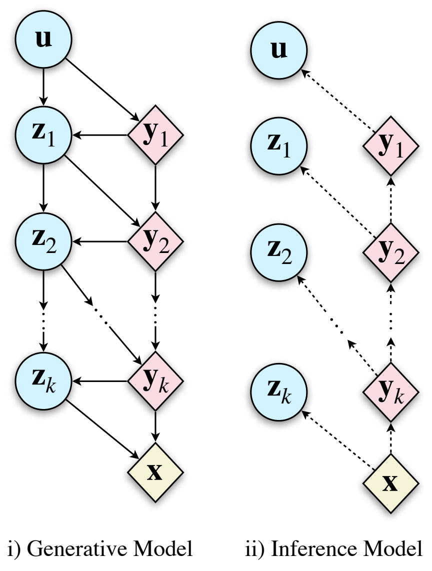


Figure 5.5: Hierarchical (multi-level) Self-Supervised VAE architecture. The plain ssVAE architecture introduced on section 2.3, can be easily extended by using k (hence k-leveled ssVAE) discrete and deterministic representations of the given data.

data transformations, where we define $\mathbf{z}_0 = \mathbf{u}$ and $\mathbf{y}_{K+1} = \mathbf{x}$, can described with the following lower-bound:

$$\mathcal{L}(\mathbf{x}) = \sum_{k=1}^{K} \mathbb{E}_{q(\mathbf{z}_k|\mathbf{y}_{k+1})q(\mathbf{y}_k|\mathbf{y}_{k+1})} \log p_\theta(\mathbf{y}_{k+1}|\mathbf{y}_k, \mathbf{z}_k) + \mathbb{E}_{q(\mathbf{u}|\mathbf{y}_1)q(\mathbf{y}_1|\mathbf{y}_2)} \log p_\theta(\mathbf{y}_1|\mathbf{u}) +$$
$$- \mathbb{E}_{q(\mathbf{y}_1|\mathbf{y}_2)} \mathcal{KL}(q(\mathbf{u}|\mathbf{y}_1)||p(\mathbf{u})) - \sum_{k=2}^{K} \mathbb{E}_{q(\mathbf{z}_{k-1}|\mathbf{y}_k)q(\mathbf{y}_k|\mathbf{y}_{k+1})} \mathcal{KL}(q(\mathbf{z}_k|\mathbf{y}_{k+1})||p(\mathbf{z}_k|\mathbf{y}_k, \mathbf{z}_{k-1})). \quad (5.4)$$

2.5 Image Transformations

With property (A) described in section (2.2), we have decided that the self-supervised (or compressed) representation would be a discrete and deterministic transformation of the data. For image data, this can be done in various ways, and some of them are represented in Figure 5.6. We can see that with these transformations, even though we discard a lot of information, the global structure is preserved. However, what sets them apart is which high-level details they neglect. As a result, in practice the model should have the ability to extract a general concept of the data in the first stage, and at the same time, the details that characterize and make them unique, given the chosen transformation, afterwards. Furthermore, different compressed representations enable different functionalities of the model. For example, by using a downscaled, grey-scale or a hazed representation, the model automatically utilizes super-resolution, coloring and dehazing capabilities respectively. An interesting approach is to retrieve a *sketch* representation of the image (Geirhos et al., 2018). Even though humans recognize objects primarily from their shapes, learning algorithms are known to be biased on textures (Geirhos et al., 2018). In this way, we can debias our framework, by indicating the importance of the shape, but also providing an simpler initial task for the initial part of the algorithm (that is, the process of the compressed representation).

3 Neural Network Architecture

The choice of the NN architecture is crucial for the performance and the scalability of the overall framework, and usually architectures that showcased great performance in discriminate tasks (i.e. classification) are used in

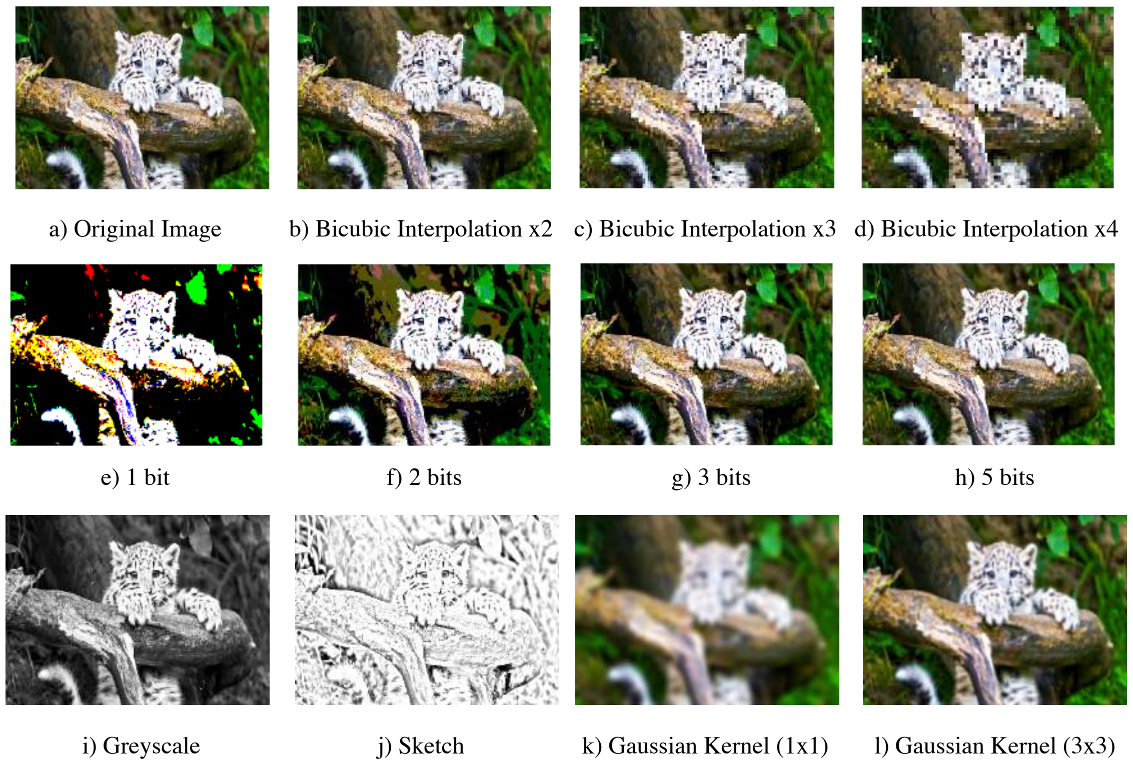


Figure 5.6: Image Transformations. All of these transformations still preserve the global structure of the samples but they disregard the high resolution details in different ways.

generative modelling tasks as well. However, the internal representations that the networks have to discover are fundamentally different, and little attention has been given into designing a NN specifically for an auto-encoder setting. For example, in classification tasks the network extracts specific representation of a particular object, in contrast with the generative models, where we aim for discovering the semantic structure of the data. Thus, as we argue that we can benefit from a carefully designed architecture, in this section we present our approach.

For building blocks of the network, we employed densely connected convolutional networks instead of residual ones. The motivation for this choice is that since DenseNets encourage feature reuse, it will help preserve visual information from the very first layer effectively, while requiring less trainable parameters. Thus, the network could discover easier generic graphical features and local pixel correlations. The concatenation of the filters will also alleviate the vanishing-gradient problem and allow us to build deep architectures. Additionally, exponential linear units (ELUs) are used everywhere as activation functions. In contrast to ReLUs, ELUs have negative values which allows them to push mean unit activations closer to zero, which speeds up the learning process. This is due to a reduced *bias shift effect*; bias that is introduced to the units from those of the previous layer which have a non-zero mean activation.

Typically, every convolution operation precedes a batch normalization layer, as they empirically exhibit a boost in performance in discriminate tasks. However, their performance is known to degrade for small batch sizes, as the variance of the activation noise that they contribute is inversely proportional to the number of data that is processed. This noise injection, in combination with their intensive memory demands, can be critical drawbacks when we process image data, especially high-dimensional ones. We instead use weight normalization, where even though it separates the weight vector from its direction just like batch normalization, they do not make use of the variance. This allows them to get the desired output even in small mini-batches, while allocating a small proportion of memory. We empirically find out that indeed using weight normalization reduces the overfitting of the model. In addition, we used a data-dependant initialization of the model parameters, by sampling the first batch of the training set. This will allow the parameters to be adjusted by the output of the previous layers, taking into account and thus resulting into a faster learning process.

An important element of the auto-encoding scheme is the process of feature downscaling and upscaling. Despite its success in classification tasks, pooling is a fixed operation that replacing it with a stride-convolution layer can also be seen as generalization, as the scaling process is now learned. This will increase the models' expressibility with the cost of adding a negligible amount of learning parameters. For the upscaling operation, even though various methods have been proposed (Shi et al., 2016), we found out that the plain transposed convolution generalised better than the others, while requiring far less trainable parameters. Finally, inspired from the recent advantages on super-resolution neural network architectures, we used *channel-wise attention* blocks (CA) at the end of every DenseNet block (Zhang et al., 2018). The CA blocks will help the network to focus on more informative features, by exploiting the inter-dependencies among feature channels. Thus, it performs feature recalibration in a global way, where the per-channel summary statistics are calculated and then used to selectively emphasise informative feature-maps as well as suppress useless ones (e.g. redundant feature-maps). This is done through a global average pooling, that squeezes global spatial information into a channel statistical descriptor, followed by a gating mechanism, where it learns nonlinear interactions between the input channels.

The core building blocks and the network of an auto-encoding network are illustrated in Figure 5.7.

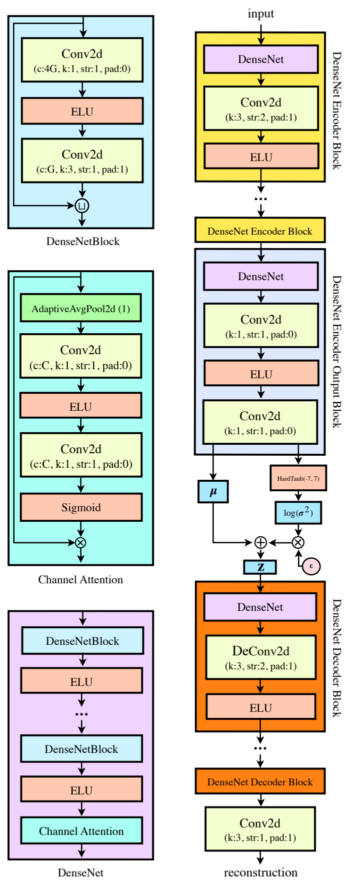


Figure 5.7: Architecture of our autoencoder. On the right, there are some basic buildings block of the network. The notation as 'G' on the Conv2D channels indicate the growth rate of the densely connected network. The ϵ indicates a random variable drawn from a standard Gaussian, which help us to make use of the *reparametrization* trick. Until $\mathbf{z}$, we refer to this architecture as *Encoder NN* and after as *Decoder NN*. The former and the later where used as building blocks to every model that we train and evaluate.

Chapter 6

Experimental Setup

"I don't know anything, but I do know that everything is interesting if you go into it deeply enough."

— Richard Feynman

This chapter will lay out our experimental setup that we conducted in detail. We will present the image datasets that we use to evaluate our models for density estimation tasks, our optimisation scheme, the used evaluation metrics, hardware and software.

1 Datasets

CIFAR-10 The CIFAR-10 dataset is a well-known standard benchmark for machine learning. Specifically, it is used mainly for classification and generative modelling tasks. The training set contains 60000 examples and the test set 10000 examples, with 10 different classes. Images are 32 by 32 pixels.

ImageNet32, ImageNet64 & Imagenette The ImageNet32 and ImageNet64 datasets are two downsampled versions of the ImageNet dataset, which were specifically design for density estimation and generative modeling experiments. The two datasets consist of images of resolution 32×32 and 64×64 respectively, and both were introduced in van den Oord et al. (2016b). In contrast with CIFAR-10, ImageNet models are in general less compressible. This is because ImageNet has greater variety of images, and the CIFAR-10 images were most likely resized with a different algorithm than the one used for the ImageNet images. The ImageNet images are less blurry, which means neighboring pixels are less correlated to each other and thus less predictable. The datasets contain exactly the same number of images as the original ImageNet, i.e., 1281167 training images from 1000 classes and 50000 validation images with 50 images per class. Imagenette is a subset of 10 classified classes from Imagenet. The purpose of this dataset is to form a vision dataset researchers could use to quickly evaluate novel algorithm ideas.

CelebA The Large-scale CelebFaces Attributes (CelebA) Dataset, as the name suggests, consists of 202.599 celebrity images. The images in this dataset has large diversities and large quantities, but it also covers large pose variations and background clutter. CelebA is a very popular dataset when it comes to qualitative analysis of generative models, showcasing results for both unconditional face synthesis and conditional generation based on give attributes (i.e. hair color).

For the first two described datasets, we use the provided validation as a test set, and create a new one using 15% randomly selected data from the training set. We augment the training data by using random horizontal flips and random affine transformations and normalize the data uniformly in the range (0, 1). As for CelebA, we initially cropped the image on the 40 vertical and 15 horizontal component of the top left corner of the crop

Figure 6.1: Random samples from the (i) CIFAR-10 (ii) ImageNet and (iii) CelebA image datasets.

box, which height and width were cropped to 148. Besides the uniform normalization of the image, no other augmentation was applied. Samples from these dataset are presented in Figure 6.1.

2 Optimisation and hyper-parameters

All models were trained using AdaMax optimizer (Kingma and Ba, 2014) with learning rate 0.002 and a lower bounded (at 0.0001) exponential learning rate scheduler, that decay the learning rate of each parameter group by gamma 0.999999. The dimensionality of all the latent variables kept at $8 \times 8 \times 16 = 1024$, and as mentioned, we applied weight normalization on the parameters with data-depended initialisation (Salimans and Kingma, 2016). The building blocks of the neural networks that any method employed is described in the 5. In short, we use a composition of DenseNets (Huang et al., 2016) and channel attention (Zhang et al., 2018) with ELUs (Clevert et al., 2015) as activation functions.

3 Evaluation Metrics

In line with literature, we evaluate maximum likelihood models for image data on density estimation tasks in bits per dim (bits/dim or *bpd*) units. Specifically, *bpd* is the number of bits such a model would need to losslessly describe each pixel of the image to 8-bit accuracy. The negative log-likelihood value (*nll*) was estimated from the lower-bound using 512 weighted samples (Burda et al., 2015).

Additionally, we use the *Fréchet Inception Distance* (FID) (Heusel et al., 2017) as a metric for image generation quality. In short, the Fréchet distance between two multivariate Gaussians $\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{C}_1)$ and $\mathbf{x}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{C}_2)$ is:

$$d^2 = ||\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2||^2 + \mathrm{Tr}(\boldsymbol{C}_1 + \boldsymbol{C}_2 - 2\sqrt{\boldsymbol{C}_1 \cdot \boldsymbol{C}_2}).$$

The FID is calculated by assuming that $\mathbf{x}_1$ and $\mathbf{x}_2$ are the activations of the coding layer pool 3 of the inception model for generated samples and real world samples respectively. $\boldsymbol{\mu}_n$ is the mean and $\boldsymbol{C}_n$ the covariance of the activations of the coding layer over all real world or generated samples. As a result, the disturbance level rises from zero and increases to the highest level, indicating that FID captures the disturbance level very well by monotonically increasing. For computing the FID scores we used 10k generated images and 10k real images from the test set, but also 50k generated images and 50k real images from the train set.

4 Hardware, Software and Experimental Code

The code for our experiments is implemented using PyTorch (Paszke et al., 2017). The experiments were run using GTX 1080Ti or/and 2080Ti Nvidia GPUs. The code that is used to produce the results of the experiments is available at `https://github.com/ioangatop/srVAE`.

Chapter 7

Evaluation

"Without data, we are just another person with opinion."

— W. Edwards Deming

In this chapter, we present our results. We start with a thorough analysis of the importance of a data driven prior, where we show that a bijective network, and specifically a RealNVP architecture, results into a more flexible, multi-modal prior, that offers better density estimation performance, reconstructions and generations when evaluated on natural images. We then continue in section 2 to investigate how the Self-Supervised Variational Auto-Encoder generative model performs on natural images. We conclude by expanding the model using a different self-supervised representation, namely a sketch transformation, and by constructing a deeper architecture with multiple representations. The latter experiments will be conducted on images coming from the CelebA and Imagenette datasets, utilizing a larger resolution than the previous experiments.

1 The impact of the bijective prior on VAEs

In this section, we will evaluate the performance of Variational Auto-Encoder with using different latent variable prior distributions, both from a quantitative and qualitative view. The models named as "VAE" represent the plain architecture with standard normal prior, while "VAE + MoG" and "VAE + RealNVP" represent data-driven priors, where the former employs 10 mixture of Gaussians and the latter the bijective network RealNVP. For a fair comparison, all the models share the same neural network architectures both on the encoder and decoder, and specifically the one depicted on the right part in figure 5.7.

Quantitative Analysis Table 7.1 represents the quantitative results of density estimation on the natural image dataset CIFAR-10. We measure performance by estimating the log-likelihood with 512 importance weighted samples on the test set. Starting with the plain VAE model, we see that the achieved likelihood is far superior to the score of 4.54bpd reported in the literature (Gregor et al. (2016) for a Deep convolutional VAE, where we result in 3.88bpd). This shows that we have managed to construct a very efficient overall framework,

Table 7.1: Generative modelling performance on CIFAR-10 obtained from different priors in bits per dimension. The generation time is based on 100 runs on a singe GeForce GTX 1080 Ti GPU for 1 image (and 100 images).

Model	*nll* (*bits/dim*)	*reconstruction loss* RE$_x$	*regularization loss* KL$_z$	#params	*generation time* (sec)
VAE	3.88	*6675*	*1596*	20M	0.01 (0.07)
VAE + MoG	3.83	*6512*	*1653*	22M	0.01 (0.07)
VAE + RealNVP	3.51	*5540*	*1966*	32M	0.07 (0.14)

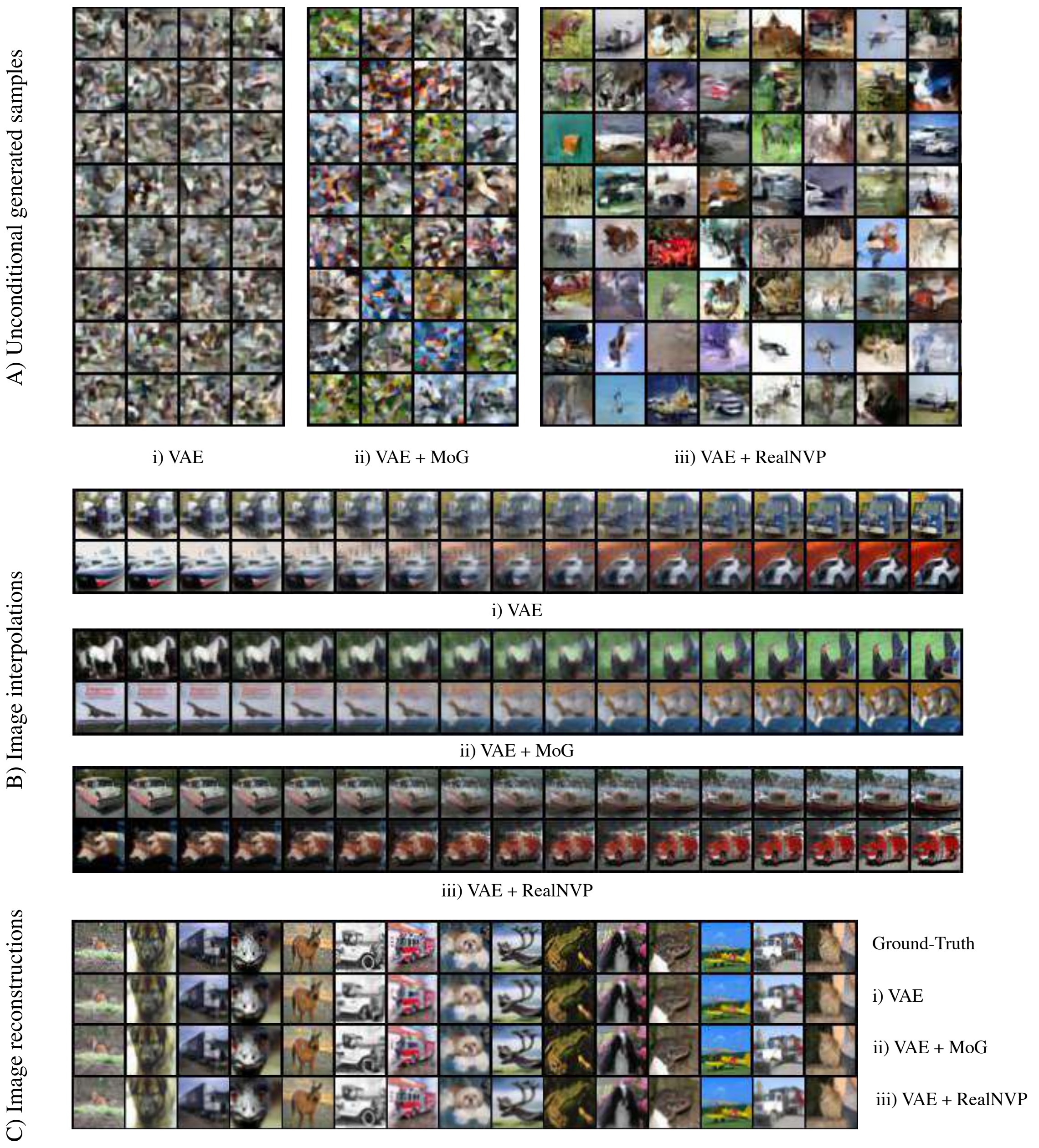


Figure 7.1: Qualitative results on CIFAR-10 for VAEs with various latent prior distributions. The depicted results show A) unconditional generative samples B) image interpolations and C) image reconstructions.

in terms of the neural network architecture, optimizer selection and the decoder distribution. The use of a mixture of Gaussians, a learned distribution, performs even better with a negligible cost on additional trainable parameters, which do not allocate extra time in sample generation. However, the adaption of RealNVP as a bijective network improves significantly the likelihood score. Interestingly, both models with learned priors perform better than the fixed one, by reaching better reconstruction loss with the cost of allocating more regularization cost. This behaviour exactly reflects the relationship between variational posterior and the prior; when a fixed unimodal Gaussian is used, it pulls the variational ones to its pick. As a result, all the variational posteriors are close to the mean of the prior, resulting into a smaller $\mathcal{KL}$ divergence, but blurry reconstructions.

On the contrary, the learned multi-modal priors allow the posteriors to move more freely and are able to adjust to the variational family (see figure 3.6). This distribution mismatch is the greatest when the RealNVP prior is used, indicating that the prior allows the encoder to form more and more complex distributions, by trading the regularization loss for better reconstructions.

Qualitative Analysis The difference in performance becomes more obvious by looking at the qualitative results, presented in Figure 7.1. Looking at the generative samples, we see a clear improvement when moving to a richer prior. The plain VAE with standard normal distribution generates random patterns, while the model with a mixture of Gaussian prior enhance them with colours. However, only VAE with RealNVP showcases generations that manage to display a global context. To examine the richness of the learned space, we also perform interpolation on the latent codes between two ground-truth images. Ideally, the internal generations should result in meaningful modifications of the provided samples. In the case of VAE, we see that the inner generations produce blurry samples, indicating the inexpressiveness of the latent codes. The generations are getting better with the MoG prior, but the most impressive results are coming from the framework with RealNVP. We can see that the samples are more influenced by their closest ground-truth image while adapting more and more core characteristics when moving closer to the other. For example, on the second row of (B:iii), the original brown horse first turns into red, adopting the colour of the firetruck, before it smoothly results into it. Lastly, on image reconstructions, we again witness that a richer prior does result in better reconstructions, confirming the quantitative results presented on table 7.1. Specifically, the VAE with the bijective prior showcases an excellent performance on the natural image reconstruction task, which is contrary to the performance of the other two approaches. This is associated with the effectiveness of the powerful, invertible, data-driven prior (in our case, the RealNVP) and its ability to boost the performance significantly with a negligible sacrifice on generation speed, and none on inference.

2 Self-Supervised Variational Auto-Encoders

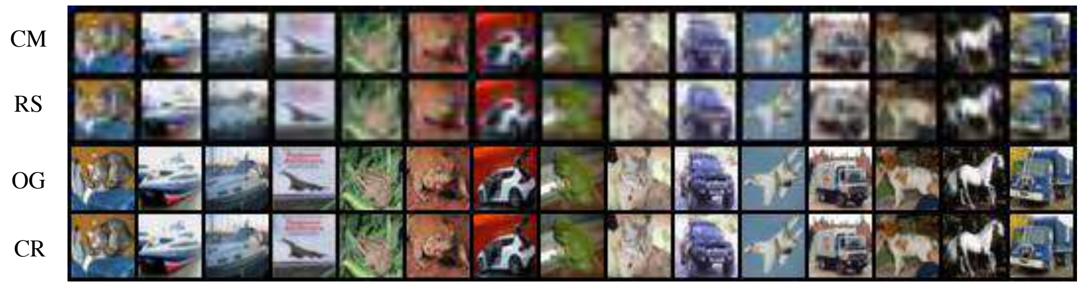
In this section, we will evaluate and compare the VAE with RealNVP prior to the Self-Supervised VAE. Here, we choose a single compressed representation for the ssVAE, a downscaled transformation of the input image, where its height and width is halved. This special case of the ssVAE, which was the first model that conceptualizes the idea for a self-supervised model, is also known as Super-Resolution VAE (or srVAE for short) (Gatopoulos et al., 2020).

Quantitative Analysis The density estimation and the image generation scores of VAE with RealNVP prior and ssVAE on CIFAR-10 and ImageNet32 are presented in Table 7.2. By analyzing the results, first, we observe that the reconstruction loss RE_x of the input image is better in case of the ssVAE approach. This is to be expected, since the ssVAE model contains a super-resolution part and, thus, can output higher quality images. However, due to the second reconstruction loss of the compressed representation, namely RE_y, the model performs worst in terms of likelihood estimation. On the sample generation performance, we see that the ssVAE significantly improves the FID score, indicating that it produces more coherent and visually pleasing generations. Interestingly, the regularization loss of the VAE is 1966 and 1707 for CIFAR-10 and ImageNet32, respectively, while the ssVAE achieves a total of around 1500 on both datasets. This result suggests that introducing an additional random variable $\mathbf{y}$ helps to match the variational posteriors and the (conditional)

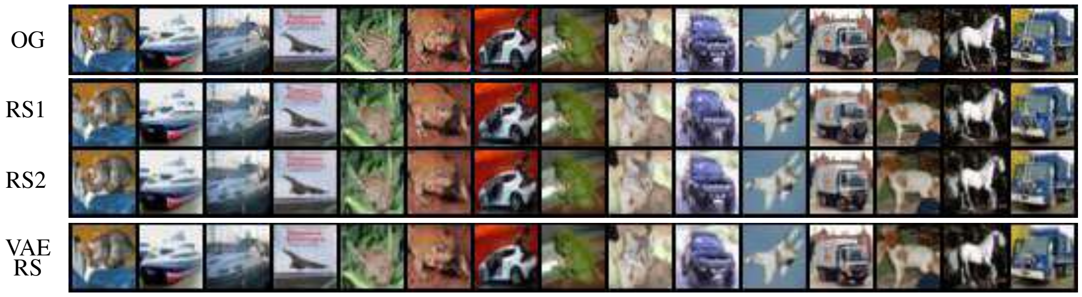
Table 7.2: Negative log-likelihood for CIFAR-10 and ImageNet32 test set. For FID, we provide values obtained on the training set and the test set (in brackets).

Dataset	**Model**	*nll*	*reconstruction loss*		*regularization loss*		FID
		(bits/dim)	RE_x	RE_y	KL_z	KL_u	
Cifar10	VAE	**3.51**	5540	-	1966	-	37.25 (41.36)
	ssVAE	3.65	5107	1241	619	819	**29.95** (**34.71**)
ImageNet32	VAE	**4.15**	7153	-	1683	-	46.93 (50.94)
	ssVAE	4.32	6548	1278	596	760	**41.34** (**45.99**)

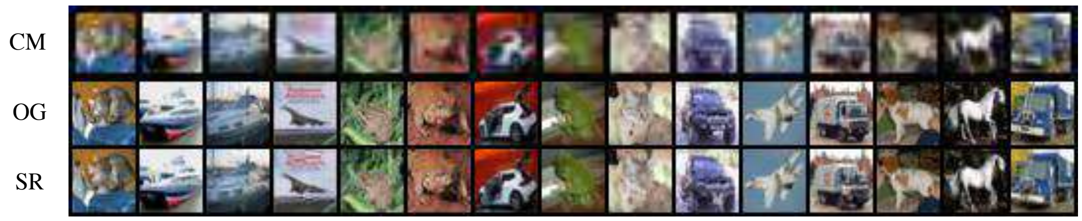
CM: Compressed, OG: Original, SR: Super-Resolution, RS: Reconstruction, CR: Conditional Reconstruction



i) Reconstruction and Conditional Reconstruction. Here we on CR we use both the ground truth compressed representation **y** and the latent code **z**.



ii) Reconstructions. On RS1 we use only one latent variable, namely $\mathbf{u}$, and sample the rest (**z**) while in RS2 we use the exact latent codes in both. VAE RS refers to the VAE+RealNVP reconstruction.

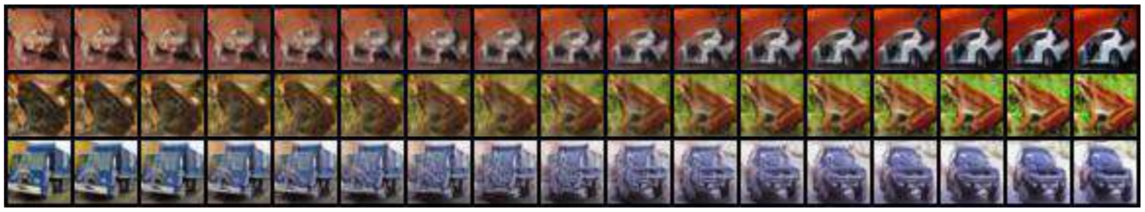


iii) Conditional Generation (Super-Resolution). Here we condition on the compressed transformation **y** (CM), sample latent variable **z** and generate the super-resolution sample (SR).

Figure 7.2: Qualitative results on CIFAR-10 for ssVAE (and specifically srVAE).

priors, which indicates that indeed splitting the generation process into two parts does result in a simpler task overall. Finally, it is important to note that since the KL_z is relatively large, the model chooses not to bypass the latent variable **z**, verifying its importance.

Qualitative Analysis Furthermore, we test the performance of these two models on image generation, reconstruction, and in the case of ssVAE, additionally for the conditional generation (super-resolution) and conditional reconstruction tasks on CIFAR-10. The results are presented on Figure 7.2. In detail, on (i) we see that the model does a great job reconstructing the downscaled transformation (compressed representation) of the image, and then when conditioned on it and using the additional latent variable **z**, the results are almost indistinguishable from the data sample. These results indicate that the model can indeed retrieve the images from any stage successfully. Additionally, the flexibility of the model allows us to generate the compressed representations from the latent variable $\mathbf{u}$, escaping the need of storing and using the ground-truth one (in our case, the downscaled one). The results of the produced reconstructions by using only one (namely $\mathbf{u}$) or two ($\mathbf{u}$ and **z**) are illustrated as RS1 and RS2 on the sub-figure (ii) respectively. In the first case, the latent codes of **z**,


i) Image Interpolations

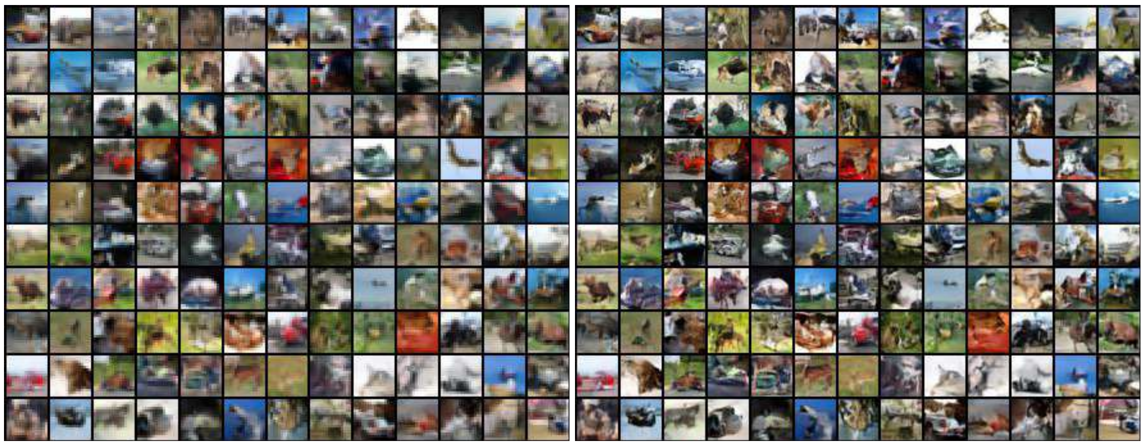

ii) Unconditional generated samples

Figure 7.3: Qualitative results on CIFAR-10 for srVAE.

are inferred through the encoder $q(\mathbf{z}|\mathbf{u}, \mathbf{y})$, while in the latter, they are the outcome of the $q(\mathbf{z}|\mathbf{x})$ distribution. Interestingly, we see that the generations of RS1 are more blurry than RS2. This confirms the impact of the latent $\mathbf{z}$ on the quality of the end result. When comparing with the reconstruction coming from VAE, we see that the results lie approximately in the middle of RS1 and RS2 in terms of image quality. Finally, by providing the downscaled image as an input to the decoder, it can be used in a manner similar to the super-resolution and in general as a conditional generation. In contrast with the conditional reconstruction (CR), here we inference the latent codes of $\mathbf{z}$ in order to recover the up-scaled image. While using likelihood loss in contrast with usual super-resolution methods, the model does a fair job to add local structure and enhance the input image.

The qualitative results and the variety of ways that one can recover an image from the ssVAE framework speculate that it can be successfully used as a very flexible solution for data compression tasks. By adjusting the number of the variables that we use and those we infer, we create the popular trade-off of memory allocation and image quality reconstruction. Specifically, by choosing to use only the latent variable $\mathbf{u}$, we result into an acceptable outcome by only using 1024 points, instead of $3 \times 32 \times 32 = 3072$ of the original image. However, by employing methods that use more variables, in our case $2 * 1024$, $3 \times 16 \times 16$ or $3 \times 16 \times 16 + 1024$ in the RS1, SR and CR approach respectively, we are able to recover the original image with more accuracy. We will analyze it further on section 3, where we test it on larger images.

Lastly, we illustrate unconditional generations and interpolation on the latent space of ssVAE in Figure 8.6. As explained, in contrast with the VAE, the ssVAE breaks the generation of an image into a two-step process. It first generates a compressed sample through the latent variable $\mathbf{u}$, and then enhance the result, which in our case adds a form of more detailed local structure, with the help of the stochastic variable $\mathbf{z}$. The results illustrate that indeed the generations of the first step output a blurry outline that sets a general concept, which is then enriched with additional components, resulting in a sharp image. While a proportion of the generations of VAE tend to be noisy and abstract, the two-staged approach seems to generate smoother, higher fidelity results. Additionally, we perform interpolation on the latent space between two randomly picked samples. The results showcase that ssVAE also create a rich latent space, where we can extract meaningful modifications of existing data points. For example, on the second row, we can see that the frog on the right slowly changes its

Table 7.3: Generative modelling performance for natural images in bits per dimension. In the case of Flow++, we provide the results of the variational dequantization in the brackets.

Model Family	Model	CIFAR-10	ImageNet 32x32
Autoregressive	PixelCNN (van den Oord et al., 2016b)	3.14	–
	PixelRNN (van den Oord et al., 2016b)	3.00	3.86
	Gated PixelCNN (van den Oord et al., 2016c)	3.03	3.83
	PixelCNN++ (Salimans et al., 2017)	2.92	–
	Image Transformer (Parmar et al., 2018)	2.90	**3.77**
	PixelSNAIL (Chen et al., 2017)	**2.85**	3.80
Non-autoregressive	RealNVP (Dinh et al., 2016)	3.49	4.28
	DVAE++ (Vahdat et al., 2018)	3.38	–
	Glow (Kingma and Dhariwal, 2018)	3.35	4.09
	IAF-VAE (Kingma et al., 2016)	3.11	–
	BIVA (Maaløe et al., 2019)	**3.08**	3.96
	Flow++ (Ho et al., 2019)	3.29 (3.08)	– (3.86)
	VAE + RealNVP prior (ours)	3.51	4.15
	ssVAE (ours)	3.65	4.32

colour to this on its left, and on the last row, roughly in the middle, the generated image looks like a hammer; rectangular edges from the trunk and overall shape of the car. Additional visual results of the VAE+RealNVP and ssVAE on CIFAR-10 and ImageNet32 are available in the Appendix (A.3).

Performance Comparison with literature Tables 7.3 and 7.4 illustrate the performance of the models in terms of likelihood estimation and generation quality respectively. While using only approximately 35 million parameters, a single latent variable and following an overall architecture without the use of any auto-regressive components, it performs closely with the flow-based generative network RealNVP on CIFAR-10, but surpasses ImageNet32. However, we witness a big improvement on the image synthesis through the given FID scores. Surprisingly, ssVAE does not only perform better than the presented likelihood-based models but also from the implicit generative models (DCGAN) which are known for their impressive unconditional samples. The results also confirm the importance of the data-driven prior to bring the gap between variational inference and implicit generative models.

Table 7.4: FID scores obtained from different models trained on CIFAR-10. Lower FID implies better sample quality. All results except ours are taken from Chen et al. (2019). In the case of our VAEs, we provide the values obtained on the training set and the test set (in brackets).

Model	FID
PixelCNN (van den Oord et al., 2016c)	65.93
PixelIQN (Ostrovski et al., 2018)	49.46
iResNet Flow (Liang et al., 2017)	65.01
GLOW (Kingma and Dhariwal, 2018)	46.90
Residual Flow (Chen et al., 2019)	46.37
DCGAN (Radford et al., 2015)	37.11
WGAN-GP (Gulrajani et al., 2017)	36.40
VAE + RealNVP prior (ours)	37.25 (41.36)
ssVAE (ours)	**29.95** (34.71)

3 Extending Self-Supervised VAE

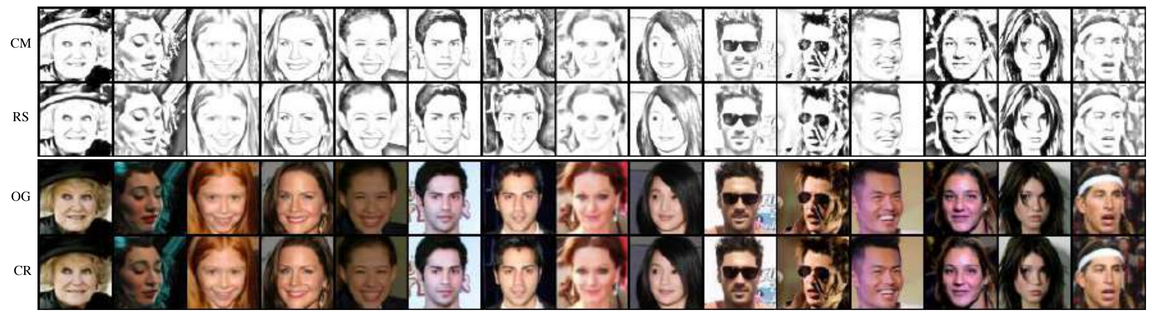
In this section, we will further experiment with ssVAE by i) employing a different self-supervised representation, namely a sketch transformation of the image, and ii) by utilizing a multiple-levelled Self-Supervised VAE. To test the scalability of our method, we will evaluate on 64×64 downscaled images, but also provide results from the plain VAE with RealNVP prior, for a more insightful comparison.

3.1 Self-Supervised VAE with sketch representations

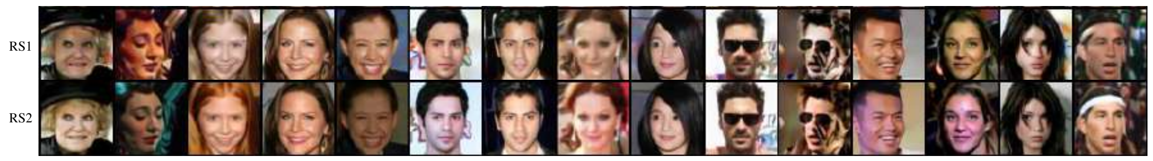
Given the astonishing performance on visual tasks, CNNs are commonly thought to recognise objects by learning increasingly complex representations of object shapes. However, Geirhos et al. (2018) showed that where humans see shapes, CNNs are strongly biased towards recognising textures. Furthermore, architectures that learn shape-based representations come with several unexpected emergent benefits such as previously unseen robustness towards a wide range of image distortions. This acted as a motivation to employ the framework of self-supervised auto-encoder with a representation that captures the shape of the object. Thus, the first part will model the outlines of the given object while the second will be responsible for its texture, resembling the human's visual system (see Chapter 1).

We can retrieve a shape-based representation of an image by detecting its edges. Edges appear when there is a sharp change in brightness and it usually corresponds to the boundaries of an object. There are many different techniques for computing the edges, like using filters that extract the gradient of the image (Sobel kernels, Laplacian of Gaussian, etc.). In this experiment, we will use the method proposed in Gastal and Oliveira (2011), as it is a fast, high-quality edge-preserving filtering of images. Specifically, in order to obtain a pencil *sketch* (that is, a black-and-white drawing) of the input image, we will make use of two image-blending techniques, known as *dodging* and *burning*. The former lightens an image, whereas the latter darkens it. A sketch transformation can be obtained by using dodge to blend the grayscale image with its blurred inverse. In this way, we produce high-quality edge-preserving filtering of the image efficiently performed in linear time.

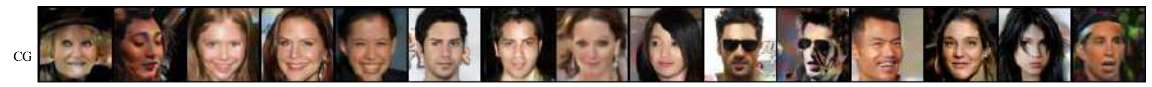
Qualitative Analysis Similar to the case when we used a downscaled transformation of the input image as conditional representation, the ssVAE framework here also allows for different ways to reconstruct an image. The qualitative results when we employ a sketch representation are visible in Figure 7.4. To start with, we



i) Sketch reconstructions and conditional reconstructions. Here we use both the ground truth compressed representation **y** and the latent code **z**.



ii) Reconstruction. On RS1 we use only one latent variable, namely **u**, and sample the other (**z**) while in RS2 we use both of the exact values.

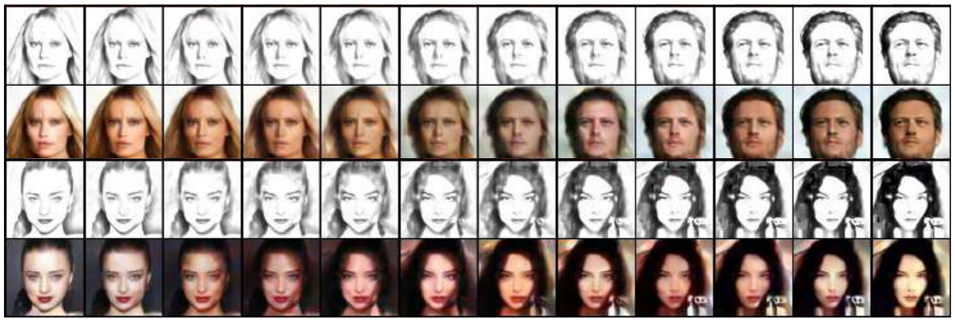


iii) Conditional Generation. Here we condition on the compressed transformation **y** (CM), sample latent variable **z** and generate the super-resolution sample (SR).

Figure 7.4: Qualitative results illustrating all the reconstruction techniques on CelebA for ssVAE-sketch.

see that even though because of the sketch representation we lose the texture of the image, we preserve not only the global information but also high-level details that characterize each person. In this way, we let the generative model emphasise on these specifics at its first step, which through the latent variable $\mathbf{u}$, manages to reconstruct with tremendous accuracy. This can be confirmed by visually comparing the images of the original compressed images (CM) with those of the reconstructed ones. These great results hold also for the conditional reconstruction, where the original sample (OG) is synthesised from conditioning on the original sketch representation (CM). Furthermore, we gain further interpretation of the model through the reconstructions that use only the latent variable $\mathbf{u}$ (RS1) and both latent variables (RS2). Given that both methods infer the sketch image, the end results still preserve all detail concepts of the human portrait. However, they are different in terms of the texture (colour) of the image, which is modelled through $\mathbf{z}$. One the one hand, in the first case its value is inferred through the sketch, and thus it produces the most probable values. That is why we see on ground-truth images that incorporate unusual lighting, it outputs a more natural outcome. On the other hand, in the second case, the latent codes of $\mathbf{z}$ are computed given the original sample. And from the results, we can see that, indeed, all the information about the texture of the image can be compressed into the latent $\mathbf{z}$, as these reconstructions are identical with the input ones. Additionally, the compressed generation (CG) process is alike to this of RS1 as they both infer the texture but different as it uses the ground-truth sketch representation. It is worth to note that the end result in both cases share the same quality, which is another indication that the reconstructions of the sketch images are excellent.

The richness of its latent space and its generative capabilities are depicted in Figure 7.5. Interpolating between two given faces does result in a meaningful mix of their characteristics. Lastly, during the unconditional image generation, the model successfully synthesizes the outline of the face before it enriches it with texture. Given that almost all the sketch representations look highly realistic, some of the end results fail to fool the human judgment mostly due to the colour inconsistency. We believe that the results can be further improved from longer training or by employing more sophisticated augmentation techniques on the first part of the algorithm during the fitting process (distort the sketch images with adversarial attacks etc.).


i) Image Interpolations

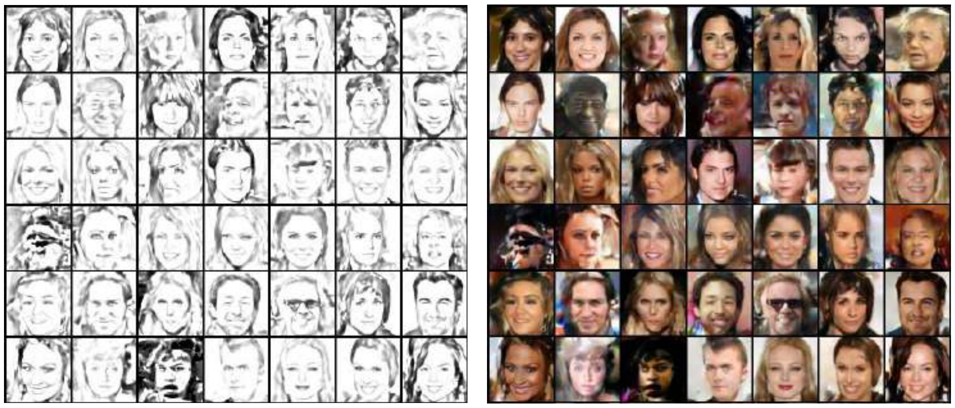
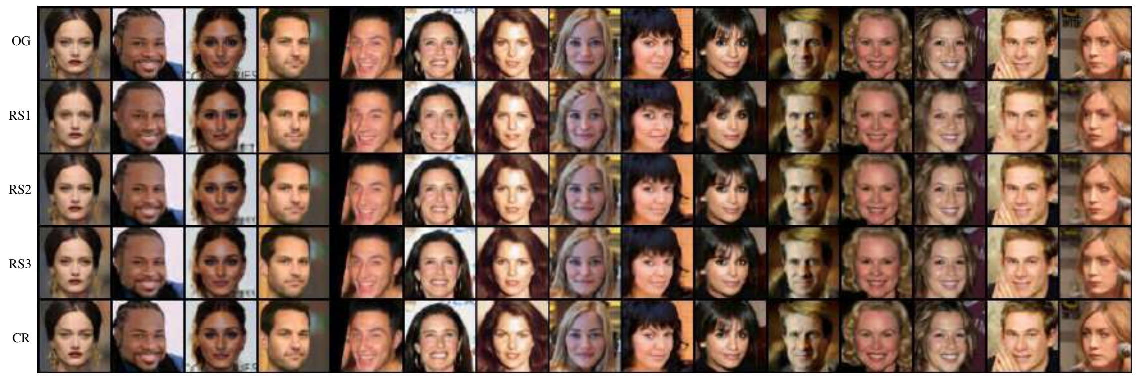

ii) Unconditional generated samples

Figure 7.5: Image interpolations and unconditional generations on CelebA for ssVAE-sketch.

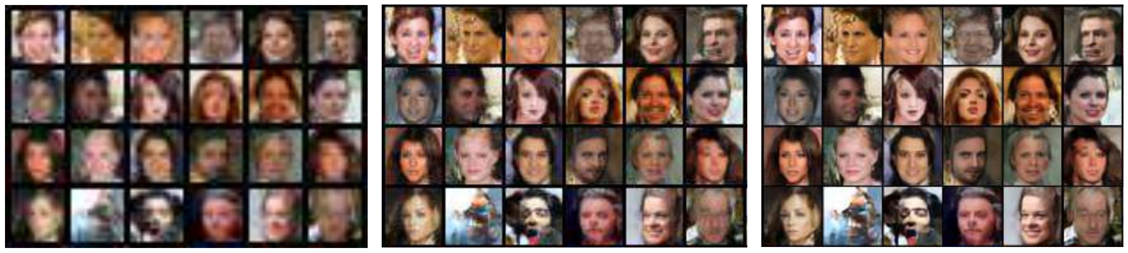
3.2 Multi-resolution Self-Supervised VAE

From the previous experimental results, we explained and studied the advantages of using a single transformation of the data, like bicubic downscaling or sketch illustration, on generative modelling tasks. However, in various settings, we can benefit from using multiple transformations. One interesting application is the process of large-dimensional images through multi-resolution training. In this way, we will upscale the dimensions sequentially, increasing at the same time steadily the difficulty of the task, until we reach the target resolution. This process, conceptually similar to the Laplacian pyramid (Burt and Adelson, 1983), will allow modelling complex, high-dimensional data efficiently, where in general latent-variable likelihood-based models fail to achieve. In this experiment, we will use a 3-levelled ssVAE to model the 64×64 images coming from the CelebA dataset. Specifically, in the second level, the model will process a downscale 32×32 version of the original ($\mathbf{y}_2$), while the first a 16×16 bicubic interpolation of the previous level ($\mathbf{y}_1$). Both of these levels will be employed with the latent variables $\mathbf{z}_3$, $\mathbf{y}_2$ and $\mathbf{z}_1$ respectively, and as usual, the latent code that follows the prior, RealNVP prior will be denoted with $\mathbf{u}$. All latent variables share the same dimensionality, which is $16 \times 8 \times 8 = 1024$. We will refer to this model as ssVAE - 3lvl.

Qualitative Analysis The 3-level structure will give us additional ways to reconstruct an image, as we employ more variables. Figure 7.6:i illustrate the results when we use one ($\mathbf{u}$), two ($\mathbf{u}$, $\mathbf{z}_1$), all three ($\mathbf{u}$, $\mathbf{z}_1$, $\mathbf{z}_2$) latent variables (while the other are sampled), denoted as RS1, RS2 and RS3 respectively. From the presented results, we see that from the prior latent codes $\mathbf{u}$ alone, the ssVAE manages to capture all the global structure of the image and output an overall representative sample. However, we can gain more accurate local structure and better results by additionally utilize the latent codes of $\mathbf{z}_1$. This can be confirmed by comparing the characteristics of the persons reconstructed by the method RS2 and RS1, as those from the first case are closer to the ground-truth ones. The use of all their latent variables result in slightly better reconstruction, and finally, from conditional reconstruction, we achieve the best overall results. Unconditional generative samples are presented at the bottom of Figure 7.6, where ssVAE first generates the leftmost images, and conditioning on them, generate sequentially those to the right.



i) Reconstruction of given image (OG) with the 3-level downsampling ssVAE model. On RS1 we use only one latent variable ($\mathbf{u}$) and sample the other ($\mathbf{z}_1, \mathbf{z}_2$) while on RS2 we sample only $\mathbf{z}_2$ and on RS3 we use the exact values from all latent variables. CR indicates the conditional reconstruction using $\mathbf{z}_2$ and $\mathbf{y}_2$.



ii) Unconditional generations of the 3-level downsampling ssVAE trained on CelebA.

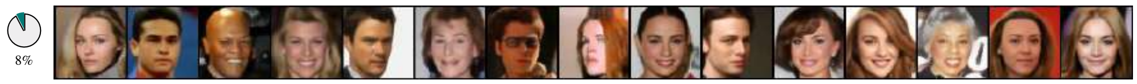
Figure 7.6: Qualitative results on CelebA for 3-level ssVAE.

3.3 Comparison with VAE

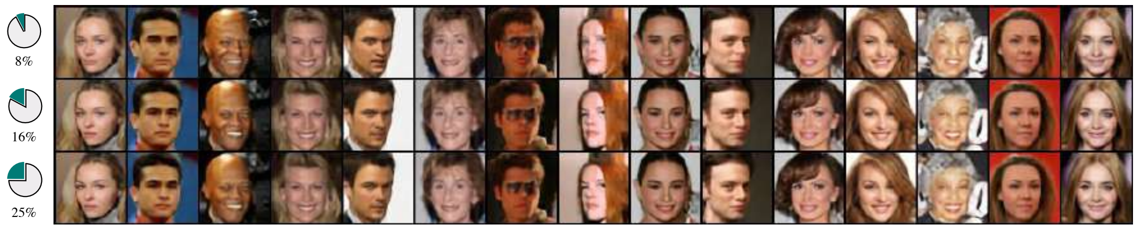
To analyze further the two presented methods, namely ssVAE - sketch and three-levelled, multi-resolution Self-Supervised VAE (or ssVAE - 3lvl), we group our results and compare them against the plain VAE framework employed with RealNVP prior, in both a quantitative and qualitative manner.

Evaluation on CelebA 64×64

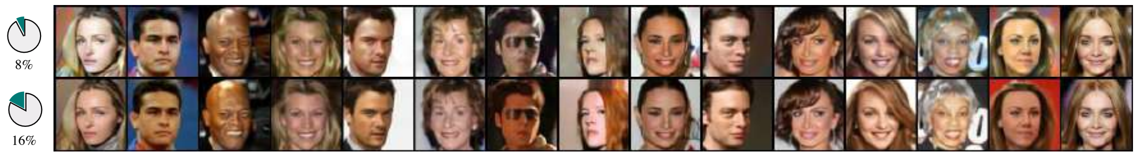
Ground-truth data samples



i) VAE with RealNVP prior reconstructions



ii) ssVAE 3-level downsample model reconstructions using; (a) **First row:** 1 latent code ($\mathbf{u}$);
(b) **Second row:** 2 latent codes ($\mathbf{u}, \mathbf{z}_1$); (c) **Third row:** (c) all three latent codes ($\mathbf{u}, \mathbf{z}_1, \mathbf{z}_2$).



iii) ssVAE 2-level sketch model reconstructions using; (a) **First row:** 1 latent code ($\mathbf{u}$);
(b) **Second row:** 2 latent codes ($\mathbf{u}, \mathbf{z}_2$).

Figure 7.7: Comparison on image reconstructions and progressive display of the data stream for images taken from the test set of CelebA. Taking advantage of the hierarchical structure of the prior and ancestral sampling, we have the option to store only a fraction of the latent variables $(\mathbf{u}, \mathbf{z}_k)$ and/or the discrete and deterministic ones $(\mathbf{y}_k)$, while the remaining variables are sampled. The global structure is already captured by smaller fragments of the bitstream, where using a more suitable data transformation (here sketch representation), can help to reconstruct the images almost perfectly, even for fragments that contain only 16% of the original size.

Quantitative Analysis The density estimation scores on CelebA are presented in Table 7.5. In contrast with the previous results, we obtain a better log-likelihood estimation by using the 3-levelled self-supervised model from VAE, even though the model employs multiple reconstruction loss. This results showcases that the benefits of incorporating self-supervised representations of the data in the modelling process became noticeable as we moving to higher dimensional data. Furthermore, the ssVAE-sketch model does also achieves better reconstruction loss than VAE but suffers from a large regularization loss. This is mainly due to the large mismatch of the variational posterior and prior of the latent variable $\mathbf{z}$ (i.e. the $\mathcal{KL}$ score), which is responsible for the texture of the image; the effect is visible when comparing the reconstructions obtained from the prior distribution (RS1) from those obtained from the posterior (RS2) on Figure 7.4. The ssVAE architecture due to their sequential architecture, require extra time to generate unconditional samples, but the dimensions becomes shorter as the number of samples increases, even though it can still be characterized as almost instant.

Qualitative Analysis On Figure 7.7 we illustrate the reconstructions of the same images from all the models along with their variants. To begin with, even though the VAE arrives into a coherent result that captures the

Table 7.5: Generative modelling performance on CelebA in bits per dimension. The generation time is based on 100 runs on a singe GeForce GTX 1080 Ti GPU for 1 image (and 20 images).

Model	*nll*	*reconstruction loss*	*regularization loss*	#params	*generation time* (sec)
	(bits/dim)	RE	KL		
VAE	3.12	*24096*	*2502*	40M	0.07 (0.1)
ssVAE - sketch	3.24	*23797*	*3816*	50M	0.16 (0.3)
ssVAE - 3lvl	**2.88**	*23336*	*1214*	50M	0.18 (0.3)

global content of the persons' face, it often alters some core characteristics and smooths out high-level details, like hair. These reconstructions are comparable with those of the 3-level VAE when only one latent code is used (RS1). However, we observer a tremendous improvement when $\mathbf{z}_1$, and additionally $\mathbf{z}_2$ are utilized, which can be interesting for compression tasks. In detail, when we use a generative model in a compression scheme setting, where the sender tries to send the information in the most efficient way to the receiver so he can recreate the message, the plain VAE allows only the use of a latent variable, and thus one way of recovering the image with a specific quality. However, as in general transferring data may take time (for example, slow internet connections or disk I/O), it is desired to progressively visualize data, i.e., to render the image with more detail as more data arrives. The framework of Self-Supervised Variational Auto-Encoder allows to sent additional latent codes as extra information that they can enhance the result and obtain a better quality. In terms of values, as the ground-truth RGB 8-bit images consist of total $3 \times 64 \times 64 = 12.288$ values, the ssVAE can reproduce it by using only 1024, 2048 or 3072 values of the latent codes, or $3 \times 32 \times 32 + 1024 = 4096$ when we a choose a conditional reconstruction approach (note the linear growth and relationship between the number of values and quality). Impressively, when we use a more sophisticated transformation as self-supervised representations like the sketch, with only the latent code we manage to preserve high-level details of the image in exchange with its true texture. However, we can retrieve it when we utilize and the second latent variables, and thus arrive at a result with remarkable accuracy by using only 16% of the original values.

Lastly, in Figure 7.8 we illustrate the unconditional generation of these models, along with their generative process for the models that belong to the ssVAE family. In general, all models exhibit great sampling diversity and rich, coherent generations, but also some minor artefacts, which may be resolved with longer training time.

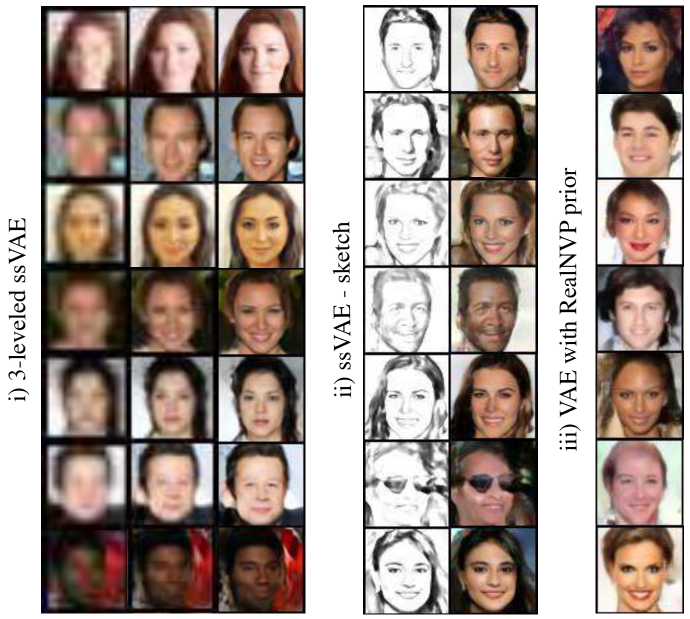


Figure 7.8: Unconditional generation of CelebA.

Evaluation on Natural Images 64×64

We also evaluate our multi-resolution method, namely the three levelled ssVAE, on 64×64 downscaled natural images samples from Imagenette. The quantitative results are presented in Table 7.6, where we see again that the ssVAE results into a better likelihood estimation, in contrast with the results on 32×32 images from CIFAR-10. The reason why our proposed method performs better on higher-dimensional data, it can be discovered by analyzing the reconstruction and the regularization loss. We see that now the accumulative reconstruction losses of the ssVAE (RE) match this of the VAE, while keeping the Kullback–Leibler divergence at lower levels. This result confirms that as the data distribution is getting more complex as we move to higher dimensions, we can heavily benefit by breaking it into multiple, more trivial steps.

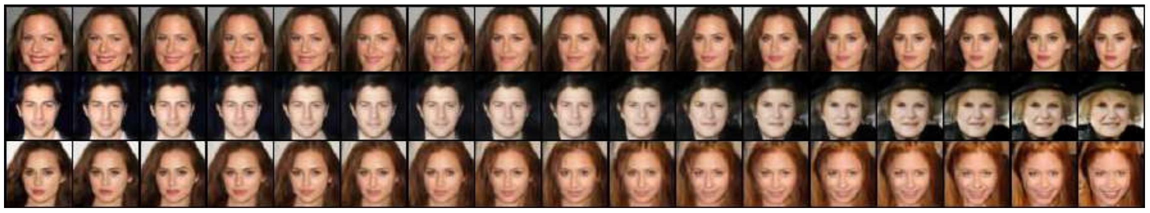
Table 7.6: Negative log-likelihood for Imagenette64 test set.

Dataset	**Model**	*nll*	*reconstruction loss*	*regularization loss*
		(bits/dim)	RE	KL
Imagenette64	VAE	3.85	*30809*	*1961*
	ssVAE	**3.70**	*30394*	*1171*

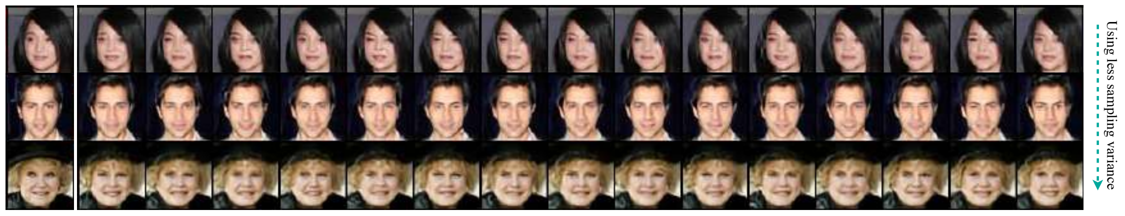
4 Interpolation through latent spaces

In this section, we will attempt to speculate the richness and interpolate the latent spaces of the multi-levelled Self-Supervised Variational Auto-Encoder through visualisation.

In Figure 7.9 we visualise i) interpolations through two ground-truth images through the latent code $\mathbf{u}$ and ii) image reconstructions, where we keep the latent code $\mathbf{u}$ but varying all the others ($\mathbf{z}_1$ and $\mathbf{z}_2$) of the 3-leveled ssVAE architecture. Just like the previous cases, in the first case, we see that the model incorporated a rich latent space $\mathbf{U}$, which is responsible for the generation and construction of the global structure of the image. Moving from one latent code $\mathbf{u}$ of a given image to another, we obtain meaningful modifications of the image that result in images that share characteristics from both of them. However, in the latter case, we see that we can alter only high-level features of the image when we keep the values of $\mathbf{u}$ but vary the others; $\mathbf{z}_1$ and $\mathbf{z}_2$. Interestingly, we see that given that ground-truth image that is illustrated on the very left, we can sample different expressions and characteristics of the same person, as the latent variable $\mathbf{u}$ is kept constant.



i) Image interpolation between two ground-truth samples though the latent variable $\mathbf{u}$.



ii) Image generation given the latent variable $\mathbf{u}$ and sampling the latent codes of $\mathbf{z}_1, \mathbf{z}_2$.

Figure 7.9: Latent space interpolation of 3-leveled Self-Supervised Variational Auto-Encoder.

Chapter 8

Conclusion

“*Research means that you do not know, but are willing to find out.*“

— Charles Kettering

In this thesis, we studied likelihood-based deep generative models, and specifically, the Variational Auto-Encoders and the flow-based networks. Initially, we proposed to extend the framework of VAEs by employing a bijective prior and then, enrich the encoder by incorporating (one or more) self-supervised representations of the data. The proposed method, namely Self-Supervised Variational Auto-Encoder, is characterized as to process images in multiple steps, by sequentially increasing the difficulty of the task at hand. Additionally, we suggest modifying the training objective to simplify the computations, and studied the effects of employing different data transformations. This novel type of generative model, which is able to perform both conditional and unconditional sampling, demonstrates overall improved performance in terms of density estimation and generative capabilities in standard image modelling benchmarks. Additionally, we demonstrate that the framework of Self-Supervised Variational Auto-Encoders utilizes multiple ways to reconstruct the trained data, which can have important applications in data compression and, given a downscale transformation as the self-supervised representation, to be used for super-resolution tasks. We showed that the benefits of employing a multi-levelled self-supervised framework against the VAE is more obvious when we move to higher-dimensional data, where the plain architecture fails to scale efficiently.

Future Work As we max out our computational resources, we believe that the results can be further improved by longer training sessions, but also obtain great results on even higher dimensional data (256×256 high-quality images). Furthermore, (Kirichenko et al., 2020) have shown that even though generative models achieve high likelihood, they are unable to detect out-of-distribution (OOD) samples, as they tend to focus more on local pixel correlations, rather than discovering semantic structure that would be specific to the training distribution. Given that a sketch represents the semantic structure of an image through only an outline while discarding many high-level features, they can potentially perform better in OOD tasks. Lastly, and most importantly, our proposed method illustrates an effecting way of injecting any type of chosen domain prior to the data generation and learning process. This allows the integration of human-level domain knowledge into the model thought handcrafted design pattern, which potentially has a tremendous impact on causal inference, boosting its performance and incorporating the requested prepossessed steps. As an example, the framework has the potential to be converted to tackle text-based data in the following way; the self-supervised part will include the core elements of a sentence (subjects, nouns, verbs etc.) that sets its skeleton, while the second part will be responsible for the generation of higher-level details that characterise it and bind it together (adverbs etc). For example, if the first part generates the word *"car"*, then the second part can sample various of sentences that directly relates to this, for example *"a red new car"* or *"a brown used car"*.

Bibliography

S. Ayzenberg V., Lourenco. Skeletal descriptions of shape provide unique perceptual information for object recognition. *Scientific Reports*, 9, 06 2019. ISSN 1552-5783.

M. S. Banks and P. Salapatek. Acuity and contrast sensitivity in 1-, 2-, and 3-month-old human infants. *Investigative Ophthalmology and Visual Science*, 17(4):361–365, 04 1978. ISSN 1552-5783.

Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives, 2012.

Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders, 2015.

P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

R. T. Q. Chen, J. Behrmann, D. Duvenaud, and J.-H. Jacobsen. Residual flows for invertible generative modeling, 2019.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder, 2016.

X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel. Pixelsnail: An improved autoregressive generative model, 2017.

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2015.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp, 2016.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL `http://jmlr.org/papers/v12/duchi11a.html`.

E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. *ACM TOG*, 30(4):69:1–69:12, 2011. Proceedings of SIGGRAPH 2011.

I. Gatopoulos, M. Stol, and J. M. Tomczak. Super-resolution variational auto-encoders, 2020.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2018.

S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/glorot10a.html`.

P. W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*, 33(10):75–84, Oct. 1990. ISSN 0001-0782. doi: 10.1145/84537.84552. URL `https://doi.org/10.1145/84537.84552`.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

K. Gregor, F. Besse, D. J. Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression, 2016.

I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville. Pixelvae: A latent variable model for natural images, 2016.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans, 2017.

A. Habibian, T. van Rozendaal, J. M. Tomczak, and T. S. Cohen. Video compression with rate-distortion autoencoders, 2019.

M. Haskett. Human Versus Computer Vision., 2019. URL https://blinkidentity.com/human-versus-computer-vision-2/.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57 (1):97–109, 04 1970. ISSN 0006-3444. doi: 10.1093/biomet/57.1.97. URL https://doi.org/10.1093/biomet/57.1.97.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2017.

J. Ho, X. Chen, A. Srinivas, Y. Duan, and P. Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design, 2019.

M. D. Hoffman and M. J. Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, volume 1, page 2, 2016.

E. Hoogeboom, J. W. T. Peters, R. van den Berg, and M. Welling. Integer discrete flows and lossless compression, 2019.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2016.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions, 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2013.

D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. ISSN 1935-8245. doi: 10.1561/2200000056. URL http://dx.doi.org/10.1561/2200000056.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc., 2016.

F. H. Kingma, P. Abbeel, and J. Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables, 2019.

P. Kirichenko, P. Izmailov, and A. G. Wilson. Why normalizing flows fail to detect out-of-distribution data, 2020.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 29–37, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL http://proceedings.mlr.press/v15/larochelle11a.html.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang. Learning for disparity estimation through feature constancy, 2017.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models, 2016.

L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. Biva: A very deep hierarchy of latent variables for generative modeling, 2019.

W. S. McCulloch and W. Pitts. *A Logical Calculus of the Ideas Immanent in Nervous Activity*, page 15–27. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262010976.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi: 10.1063/1.1699114. URL https://doi.org/10.1063/1.1699114.

Y. E. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983. URL https://ci.nii.ac.jp/naid/10029946121/en/.

G. Ostrovski, W. Dabney, and R. Munos. Autoregressive quantile networks for generative modeling, 2018.

N. Parmar, A. Vaswani, J. Uszkoreit, Łukasz Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer, 2018.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, A. Kadurin, S. Johansson, H. Chen, S. Nikolenko, A. Aspuru-Guzik, and A. Zhavoronkov. Molecular sets (moses): A benchmarking platform for molecular generation models, 2018.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models, 2014.

M. Rosca, B. Lakshminarayanan, and S. Mohamed. Distribution matching in variational inference, 2018.

F. Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks, 2016.

T. Salimans, D. P. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap, 2014.

T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications, 2017.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1948.tb01338.x.

W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, 2016.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 61(3):611–622, 1999.

J. M. Tomczak and M. Welling. Vae with a vampprior, 2017.

A. Vahdat, W. G. Macready, Z. Bian, A. Khoshaman, and E. Andriyash. Dvae++: Discrete variational autoencoders with overlapping transformations, 2018.

R. van den Berg, L. Hasenclever, J. M. Tomczak, and M. Welling. Sylvester normalizing flows for variational inference, 2018.

A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio, 2016a.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks, 2016b.

A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders, 2016c.

P. J. Werbos. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, USA, 1994. ISBN 0471598976.

Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu. Image super-resolution using very deep residual channel attention networks, 2018.

Appendix

A.1 Derivation of ELBO though Jensen's inequality

Starting from the marginal log-likelihood of the data, we will have

$$\begin{aligned}\log p(\mathbf{X}) \overset{iid}{=} \log \prod_{n=1}^{N} p(\mathbf{x}_n) = \sum_{n=1}^{N} \log p(\mathbf{x}_n) &= \sum_{n=1}^{N} \log \int p(\mathbf{x}_n, \mathbf{z}_n)\, \mathrm{d}\mathbf{z}_n \\ &= \sum_{n=1}^{N} \log \int \frac{q(\mathbf{z}_n|\mathbf{x}_n)}{q(\mathbf{z}_n|\mathbf{x}_n)} p(\mathbf{x}_n, \mathbf{z}_n)\ \mathrm{d}\mathbf{z}_n \\ &= \sum_{n=1}^{N} \log \mathbb{E}_{q(\mathbf{z}_n|\mathbf{x}_n)} \frac{p(\mathbf{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n|\mathbf{x}_n)}\end{aligned}$$

Since the function $\log(\cdot)$ is a **concave** function, meaning that

$$f\left(\mathbb{E}_{p(t)} t\right) \geq \mathbb{E}_{p(t)} f(t), \quad \text{where } \ f(\cdot) = \log(\cdot),$$

we can impose a lower bound on the above marginal log-likelihood as follows:

$$\begin{aligned}\sum_{n=1}^{N} \log \mathbb{E}_{q(\mathbf{z}_n|\boldsymbol{x}_n)} \frac{p(\boldsymbol{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n|\boldsymbol{x}_n)} &\geq \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n|\boldsymbol{x}_n)} \log \frac{p(\boldsymbol{x}_n, \mathbf{z}_n)}{q(\mathbf{z}_n|\boldsymbol{x}_n)} \\ &= \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n|\boldsymbol{x}_n)} \log \frac{p(\boldsymbol{x}_n|\mathbf{z}_n)p(\mathbf{z}_n)}{q(\mathbf{z}_n|\boldsymbol{x}_n)} \\ &= \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n|\boldsymbol{x}_n)} \log p(\boldsymbol{x}_n|\mathbf{z}_n) + \mathbb{E}_{q(\mathbf{z}_n|\boldsymbol{x}_n)} \log \frac{p(\mathbf{z}_n)}{q(\mathbf{z}_n)} \\ &= \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{z}_n|\boldsymbol{x}_n)} \log p(\boldsymbol{x}_n|\mathbf{z}_n) - \mathcal{KL}(q(\mathbf{z}_n|\boldsymbol{x}_n) \| p(\mathbf{z}_n)) \\ &\equiv \mathcal{L}(q)\end{aligned}$$

A.2 Derivation of Kullback-Leibler divergence between two isotopic Gaussians

We assume that distributions f and q as two univariate Gaussians, where $f \sim \mathcal{N}(\mu, \sigma)$ and $g \sim \mathcal{N}(\nu, \tau)$. From its definition, the Kullback-Leibler divergence will be

$$\int_{-\infty}^{\infty} f(x) \log \frac{f(x)}{g(x)} \mathrm{d}x$$

and from the hypothesis we have

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \& \quad g(x) = \frac{1}{\sqrt{2\pi\tau^2}} e^{-\frac{(x-\nu)^2}{2\tau^2}}$$

We begin with expanding the log expression inside the KL

$$\log\frac{f(x)}{g(x)} = \log\frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\frac{1}{\sqrt{2\pi\tau^2}} e^{\frac{(x-\nu)^2}{2\tau^2}}} = \log\frac{\tau}{\sigma}\frac{e^{\frac{(x-\nu)^2}{2\tau^2}}}{e^{\frac{(x-\mu)^2}{2\sigma^2}}} = \log\frac{\tau}{\sigma}e^{\frac{(x-\nu)^2}{2\tau^2}-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$= \log\frac{\tau}{\sigma} + \log e^{\frac{(x-\nu)^2}{2\tau^2}} - \frac{(x-\mu)^2}{2\sigma^2} = \log\frac{\tau}{\sigma} + \frac{(x-\nu)^2}{2\tau^2} - \frac{(x-\mu)^2}{2\sigma^2}$$

If we put this outcome into the KL we will have

$$\begin{aligned}
\int_{-\infty}^{\infty} f(x)\log\frac{f(x)}{g(x)}\mathrm{d}x &= \int_{-\infty}^{\infty} f(x)\left(\log\frac{\tau}{\sigma} + \frac{(x-\nu)^2}{2\tau^2} - \frac{(x-\mu)^2}{2\sigma^2}\right)\mathrm{d}x \\
&= \int_{-\infty}^{\infty} f(x)\log\frac{\tau}{\sigma} + f(x)\frac{(x-\nu)^2}{2\tau^2} - f(x)\frac{(x-\mu)^2}{2\sigma^2}\mathrm{d}x \\
&= \int_{-\infty}^{\infty} f(x)\log\frac{\tau}{\sigma}\mathrm{d}x + \int_{-\infty}^{\infty} f(x)\frac{(x-\nu)^2}{2\tau^2}\mathrm{d}x - \int_{-\infty}^{\infty} f(x)\frac{(x-\mu)^2}{2\sigma^2}\mathrm{d}x \\
&= \log\frac{\tau}{\sigma}\int_{-\infty}^{\infty} f(x)\mathrm{d}x + \frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)(x-\nu)^2\mathrm{d}x - \frac{1}{2\sigma^2}\int_{-\infty}^{\infty} f(x)(x-\mu)^2\mathrm{d}x \\
&= \log\frac{\tau}{\sigma} * 1 + \frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)(x-\nu)^2\mathrm{d}x - \frac{1}{2\sigma^2}\left(\sigma^2\right) \\
&= \log\frac{\tau}{\sigma} + \frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)(x-\nu)^2\mathrm{d}x - \frac{1}{2}
\end{aligned}$$

We will expand the middle part of the last expression

$$\begin{aligned}
\frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)(x-\nu)^2\mathrm{d}x &= \frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)((x-\mu)+(\mu-\nu))^2\mathrm{d}x \\
&= \frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)(x-\mu)^2\mathrm{d}x + \frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)(\mu-\nu)^2\mathrm{d}x \\
&+ \frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)2(x-\mu)(\mu-\nu)\mathrm{d}x \\
&= \frac{1}{2\tau^2}(\sigma)^2 + (\mu-\nu)^2\frac{1}{2\tau^2}\int_{-\infty}^{\infty} f(x)\mathrm{d}x + \frac{(\mu-\nu)}{\tau^2}\int_{-\infty}^{\infty} f(x)(x-\mu)\mathrm{d}x
\end{aligned}$$

Taking the last part of the previous expression:

$$\int_{-\infty}^{\infty} f(x)(x-\mu)\mathrm{d}x = \int_{-\infty}^{\infty} f(x)x\mathrm{d}x - \int_{-\infty}^{\infty} f(x)\mu\mathrm{d}x = \mu - 1*\mu = 0$$

Thus

$$\frac{\sigma^2}{2\tau^2} + \frac{(\mu-\nu)^2}{2\tau^2} * 1 + 0 = \frac{\sigma^2}{2\tau^2} + \frac{(\mu-\nu)^2}{2\tau^2}$$

And therefore it is

$$\log\frac{\tau}{\sigma} + \frac{\sigma^2}{2\tau^2} + \frac{(\mu-\nu)^2}{2\tau^2} - \frac{1}{2}$$

or

$$\frac{1}{2}\left(2\log\frac{\tau}{\sigma} + \frac{\sigma^2+(\mu-\nu)^2}{\tau^2} - 1\right) = \frac{1}{2}\left(\log\frac{\tau^2}{\sigma^2} + \frac{\sigma^2+(\mu-\nu)^2}{\tau^2} - 1\right)$$

A.3 Additional qualitative results

VAE with RealNVP prior

- **CIFAR-10**: Figures 8.1, 8.2;
- **ImageNet32**: Figures 8.3, 8.4;
- **CelebA**: Figures 8.9, 8.10, 8.11.

ssVAE

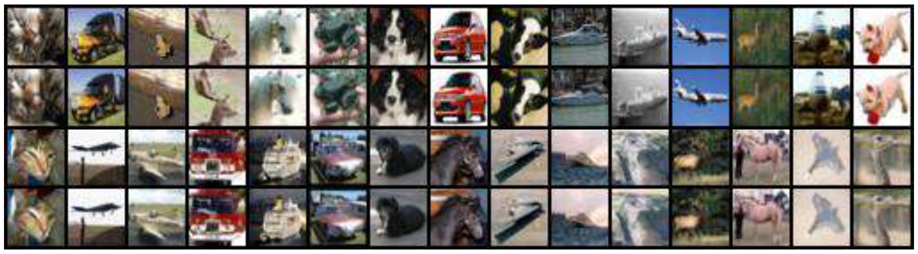
- **CIFAR-10**: Figures 8.5, 8.6;
- **ImageNet32**: Figures 8.7, 8.8;
- **CelebA** (sketch): Figures 8.12, 8.13, 8.14.
- **CelebA** (3-lvl): Figures 8.15, 8.16.

Comparison

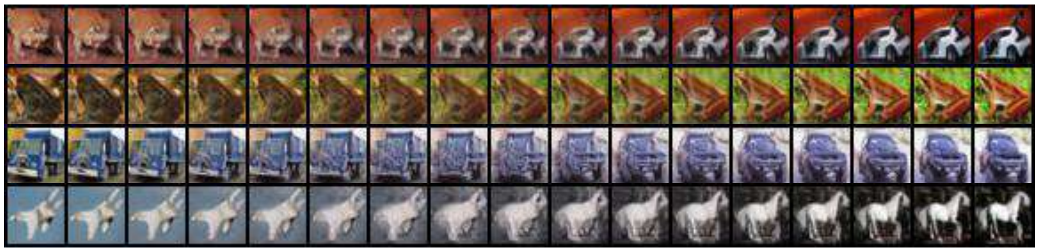
- **CelebA**: Figures 8.17, 8.18.

A.4 Thesis Statistics

This thesis consists of 16886 words, 31 figures, 7 tables, 71 references and 53 pages.



i) Reconstructions



ii) Interpolations

Figure 8.1: VAE with RealNVP prior on CIFAR-10 reconstructions and interpolations.

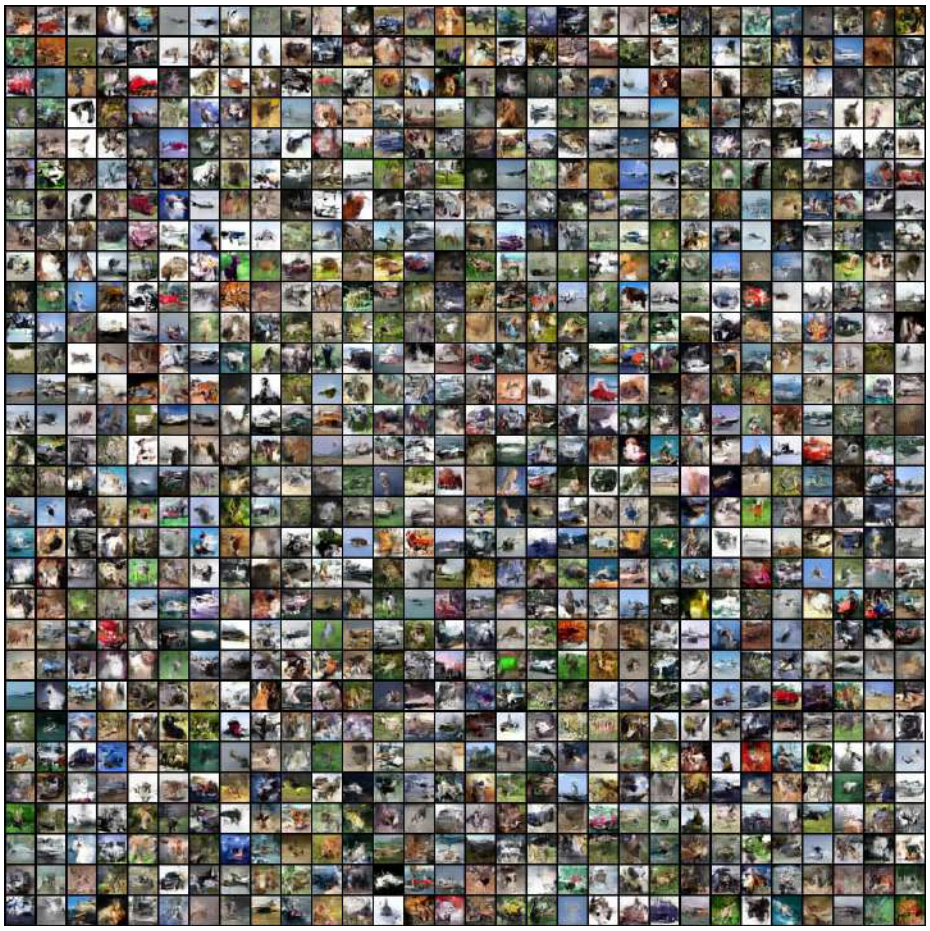
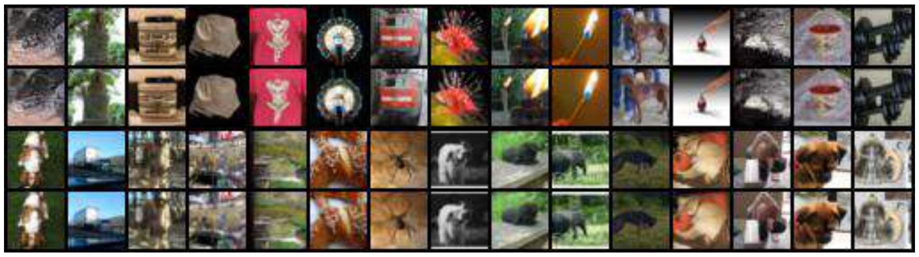
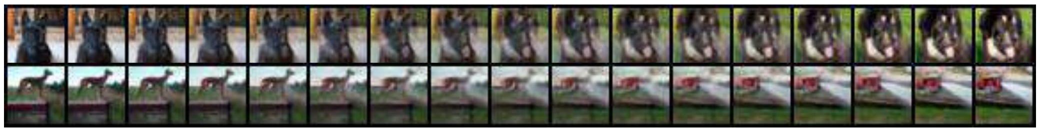


Figure 8.2: VAE with RealNVP prior CIFAR-10 unconditional generation samples.



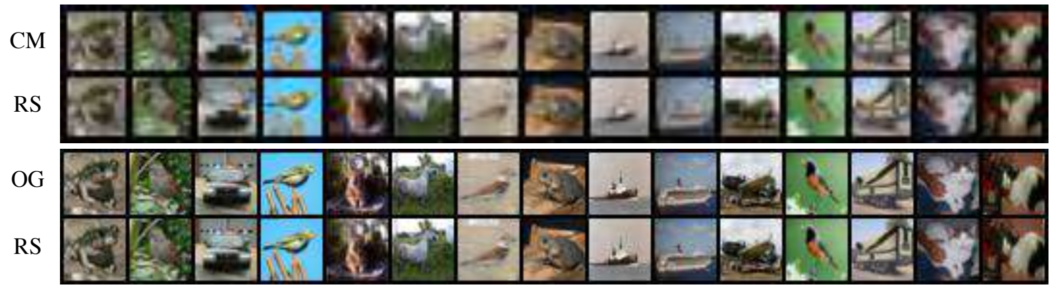
i) Reconstructions



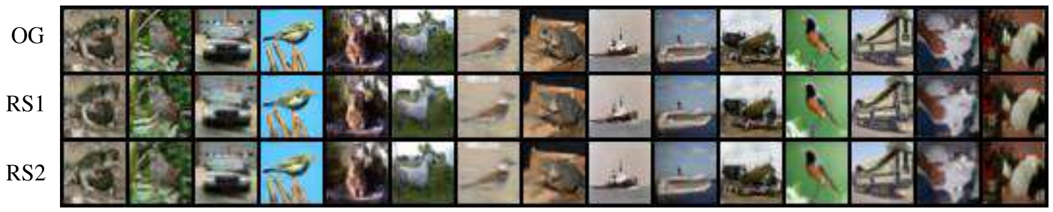
ii) Interpolations

Figure 8.3: VAE with RealNVP prior on ImageNet32 reconstructions and interpolations.

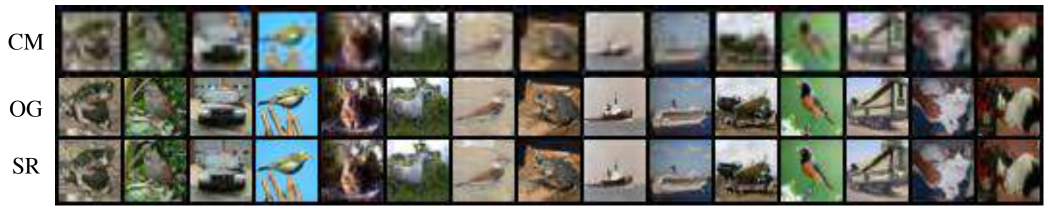
Figure 8.4: VAE with RealNVP prior ImageNet32 unconditional generation samples.



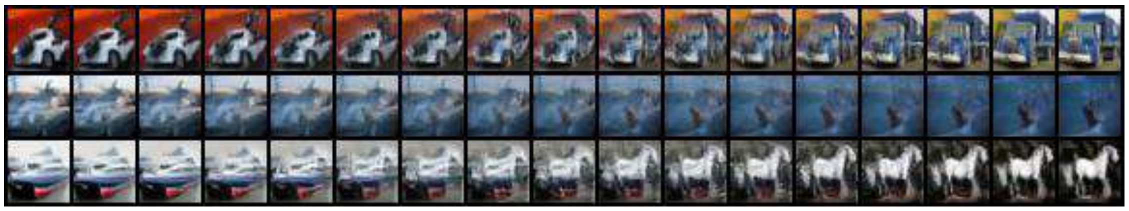
i) Conditional Reconstructions. Here we use both the ground truth compressed representation and the latent code z.



ii) Reconstructions. On RS1 we use only one latent variable, namely **u**, and sample the rest (**z**) while in RS2 we use the exact latent codes in both.



iii) Conditional Generation (Super-Resolution). Here we condition on the compressed transformation **y** (CM), sample latent variable **z** and generate the super-resolution sample (SR).



iv) Image interpolation between two ground-truth samples though the latent variable **u**.

Figure 8.5: Qualitative results on CIFAR-10 of the ssVAE.

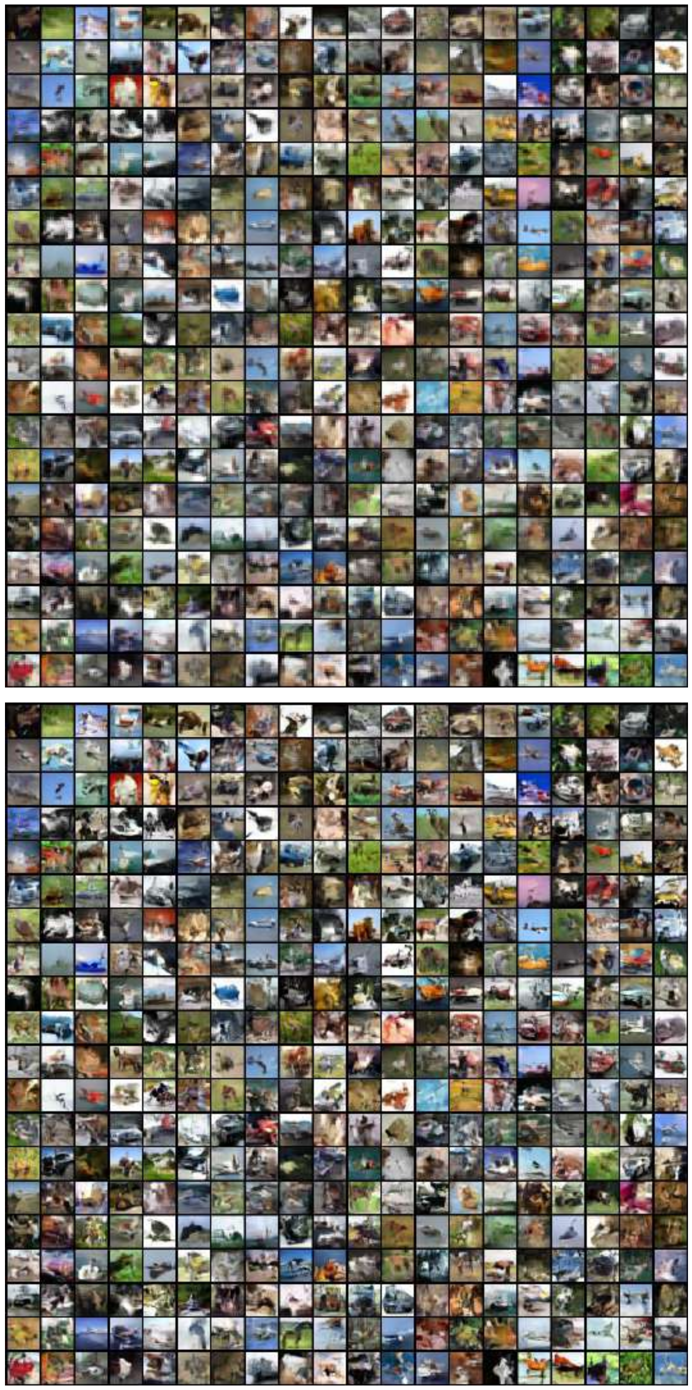
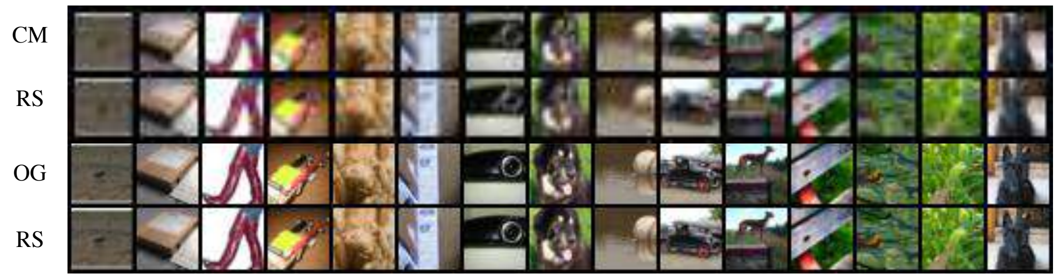
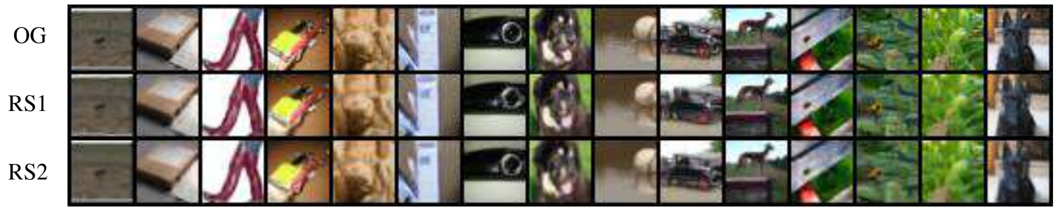


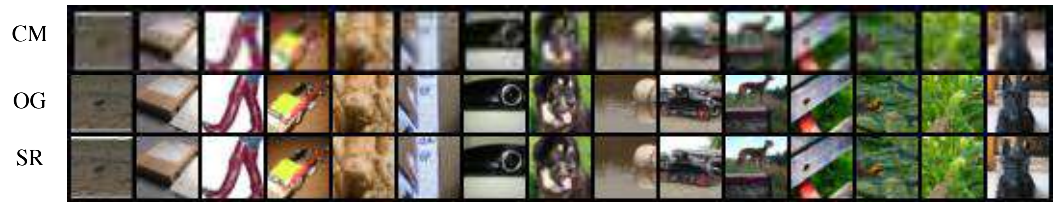
Figure 8.6: Unconditional generations of the ssVAE trained on CIFAR10. The model initially generates the $\times 2$ downscaled (compressed) representation of the image which intends to captures the "global" structure and then adds the local structure (the "details") while increasing its receptive field. These results indicate that the model successfully captures the global structured information from data on the first stage with the latent variable $\mathbf{u}$ and then adds a local structure with the help of the latent variable $\mathbf{z}$, whose responsibility is to capture the missing information between the compressed and the original data.



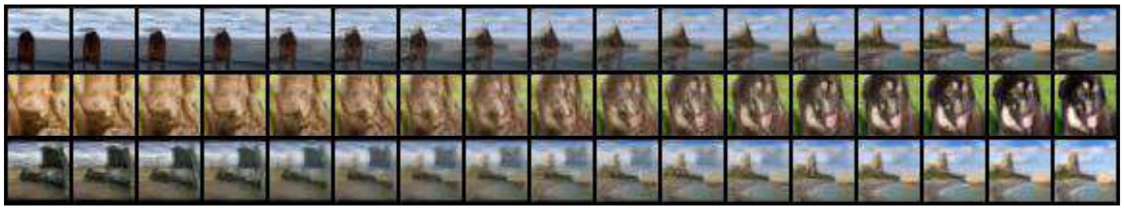
i) Conditional Reconstructions. Here we use both the ground truth compressed representation **y** and the latent code **z**.



ii) Reconstructions. On RS1 we use only one latent variable, namely **u**, and sample the rest (**z**) while in RS2 we use the exact latent codes in both.



iii) Conditional Generation (Super-Resolution). Here we condition on the compressed transformation **y** (CM), sample latent variable **z** and generate the super-resolution sample (SR).



iv) Image interpolation between two ground-truth samples though the latent variable **u**.

Figure 8.7: Qualitative results on ImageNet32 of the ssVAE.

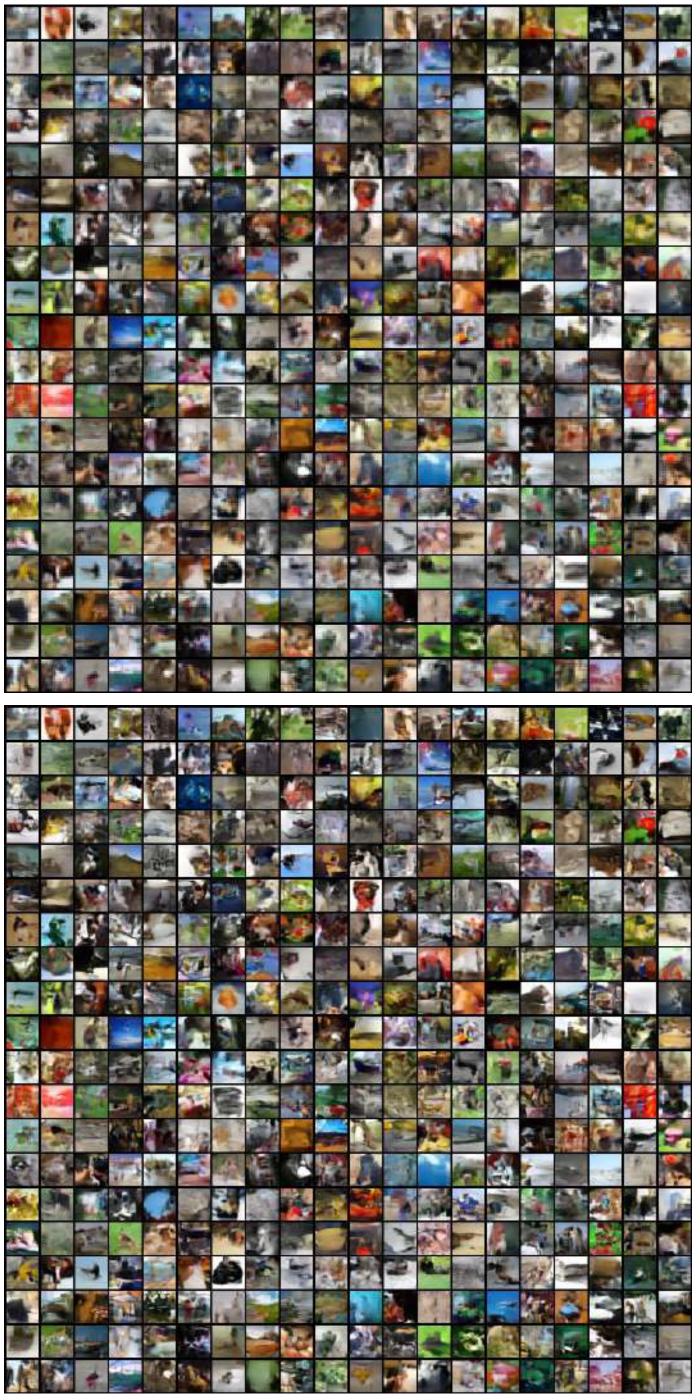
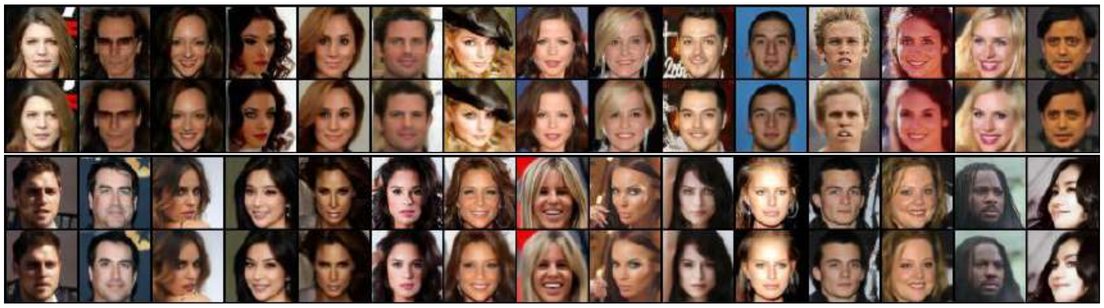
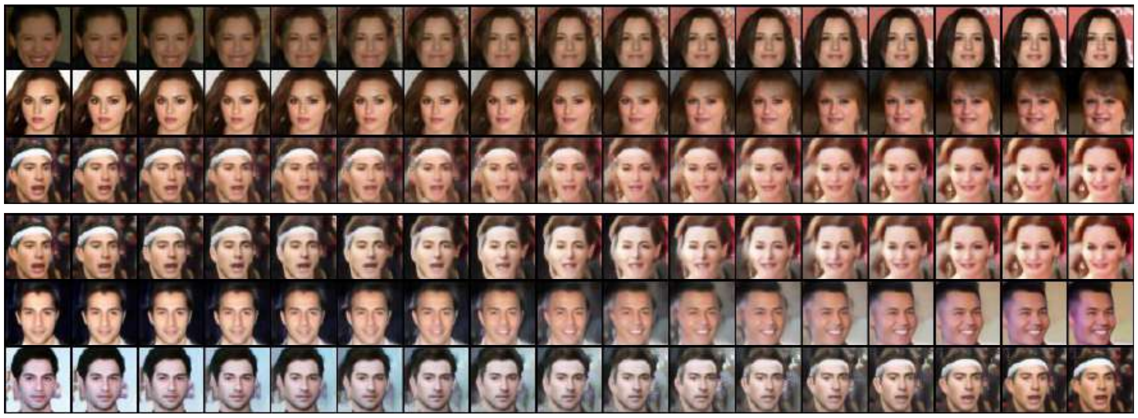


Figure 8.8: Unconditional generations of the ssVAE trained on ImageNet32. The model initially generates the $\times 2$ downscaled (compressed) representation of the image which intends to captures the "global" structure and then adds the local structure (the "details") while increasing its receptive field. These results indicate that the model successfully captures the global structured information from data on the first stage with the latent variable $\mathbf{u}$ and then adds a local structure with the help of the latent variable $\mathbf{z}$, whose responsibility is to capture the missing information between the compressed and the original data.

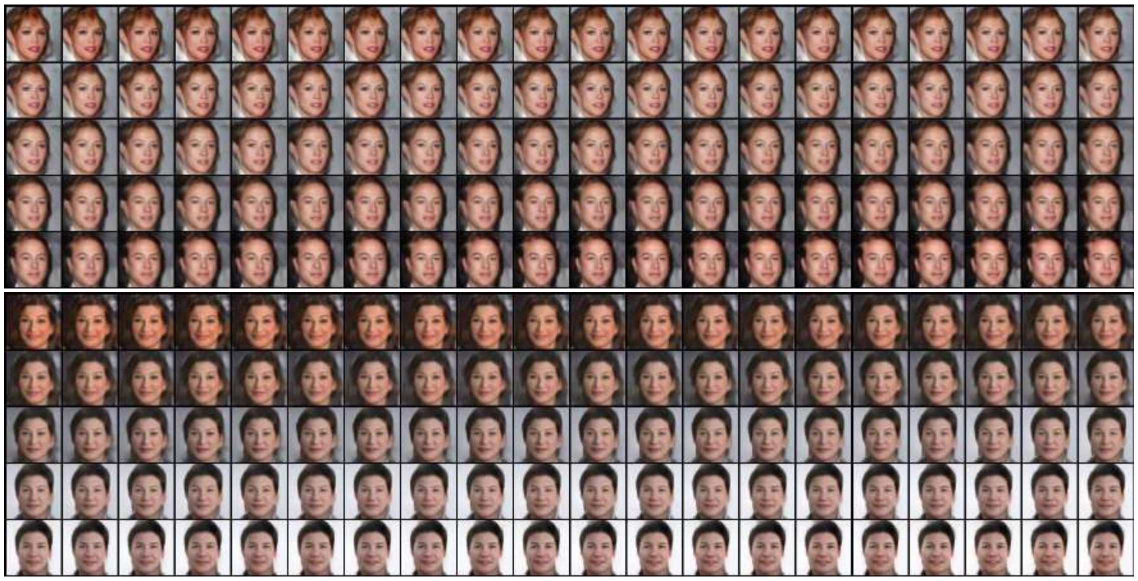


i) Reconstructions



ii) Interpolations. The top three rows are interpolations though the variational posterior codes and the next three are though the latent codes of the prior of the flow network.

Figure 8.9: VAE with RealNVP prior on CelebA reconstructions and interpolations.



ii) Interpolation though generative samples.

Figure 8.10: VAE with RealNVP prior on CelebA reconstructions and interpolations.

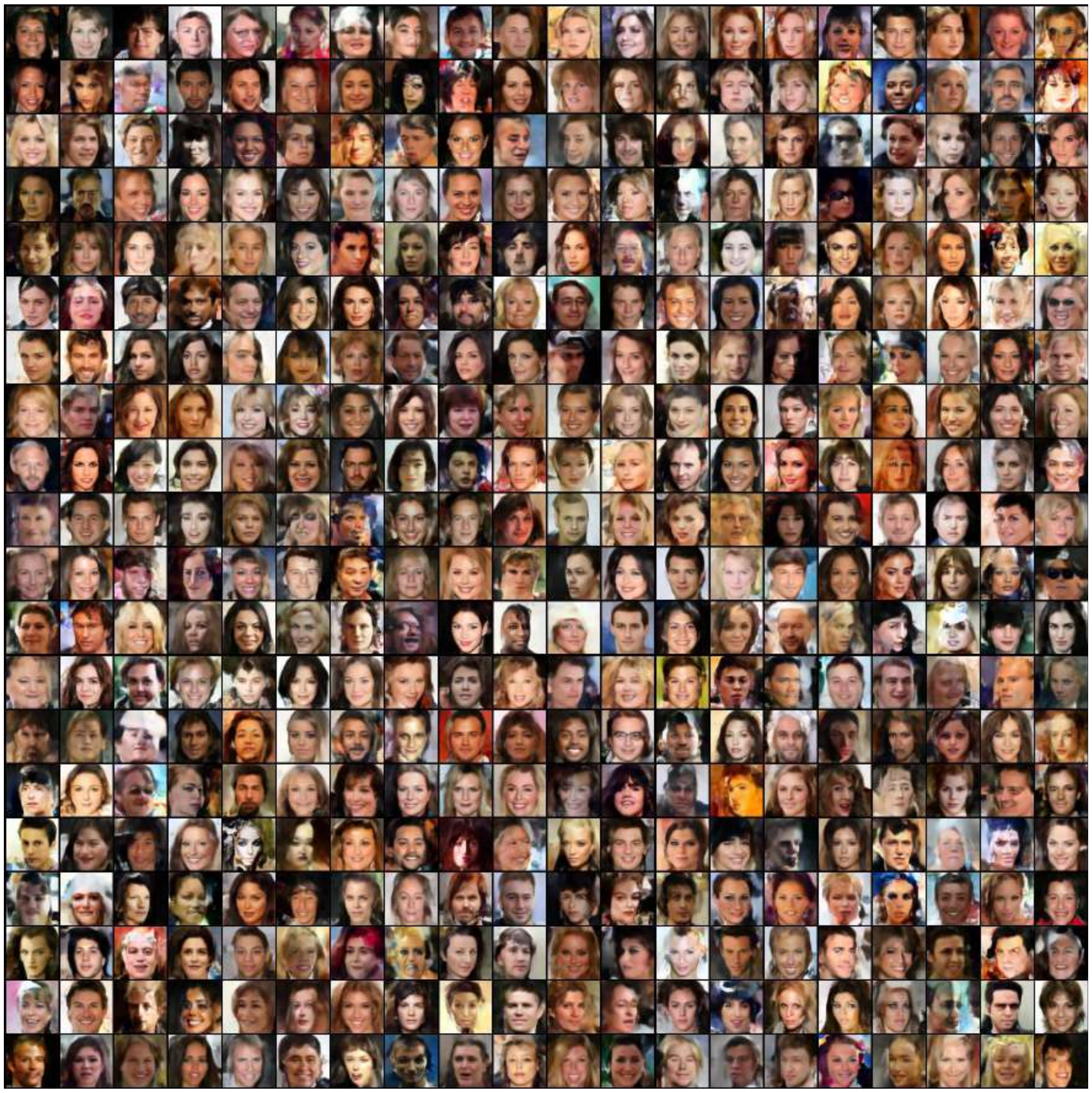
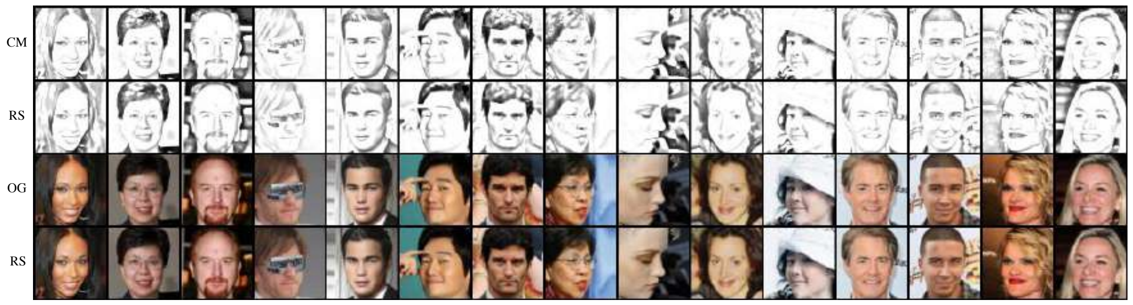
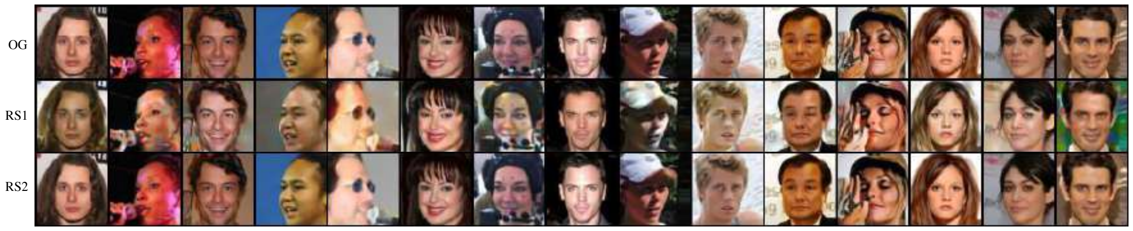


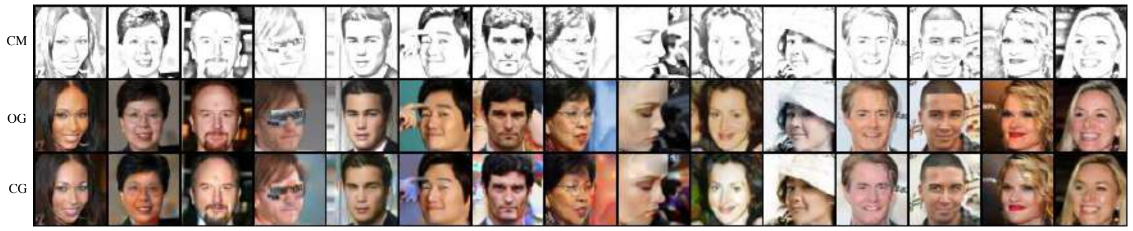
Figure 8.11: VAE with RealNVP prior on CelebA reconstructions and interpolations.



i) Conditional Reconstructions. Here we use both the ground truth compressed representation **y** and the latent code **z**.



ii) Reconstruction. On RS1 we use only one latent variable, namely **u**, and sample the other (**z**) while in RS2 we use both of the exact values.



iii) Conditional Generation. Here we condition on the compressed transformation **y** (CM), sample latent variable **z** and generate the super-resolution sample (SR).

Figure 8.12: Qualitative results on CelebA of the ssVAE-sketch.

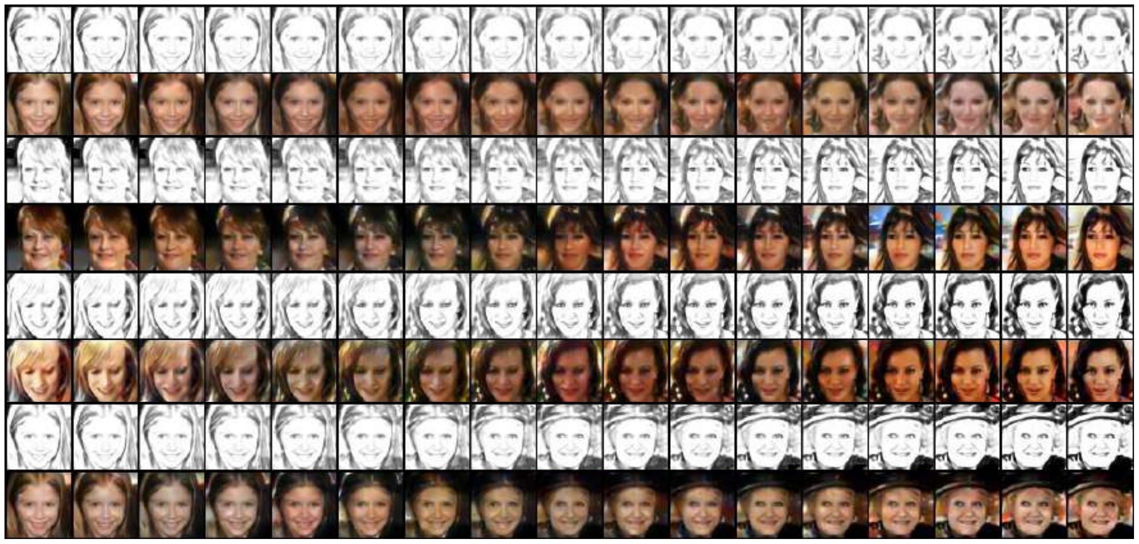


Image interpolation between two ground-truth samples though the latent variable u.

Figure 8.13: Image interpolations on CelebA of the ssVAE-sketch.

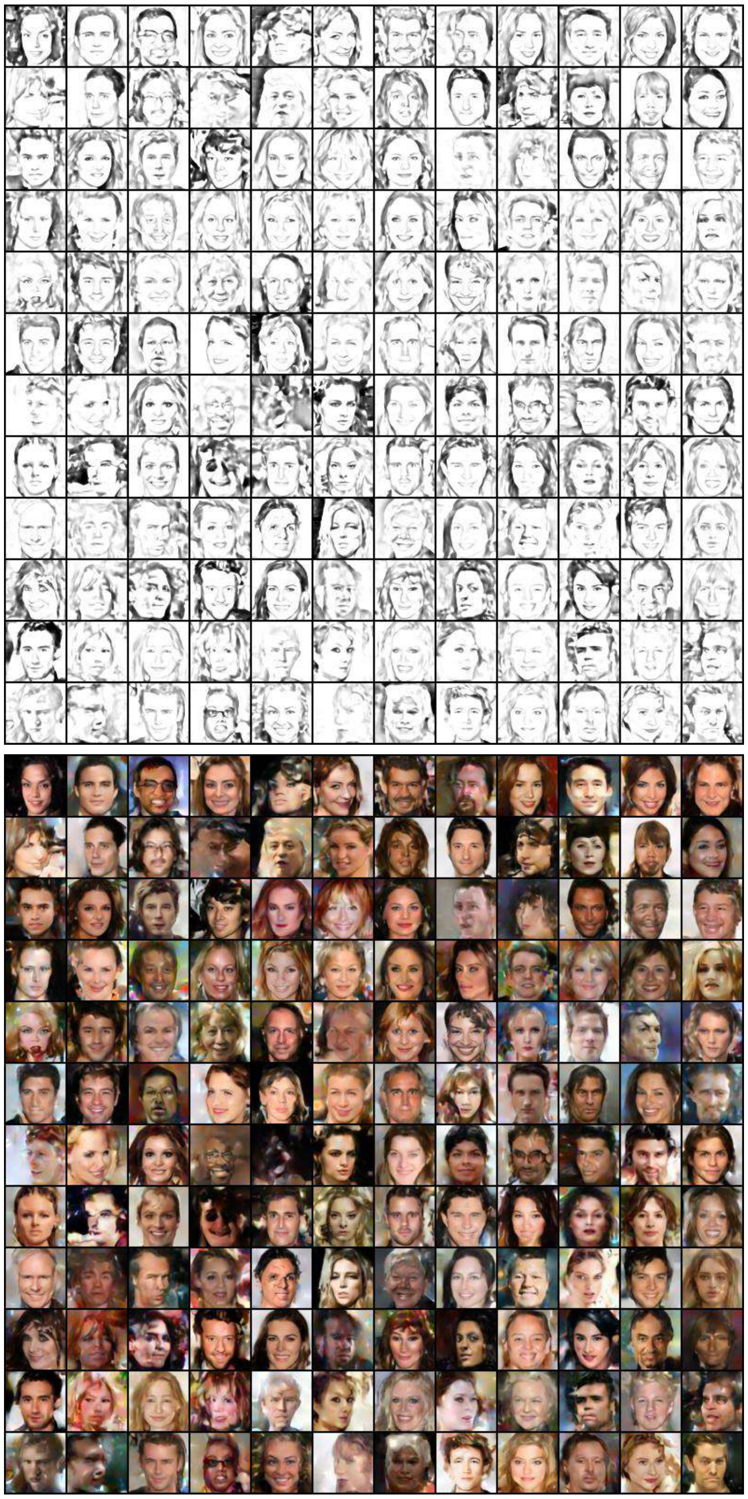
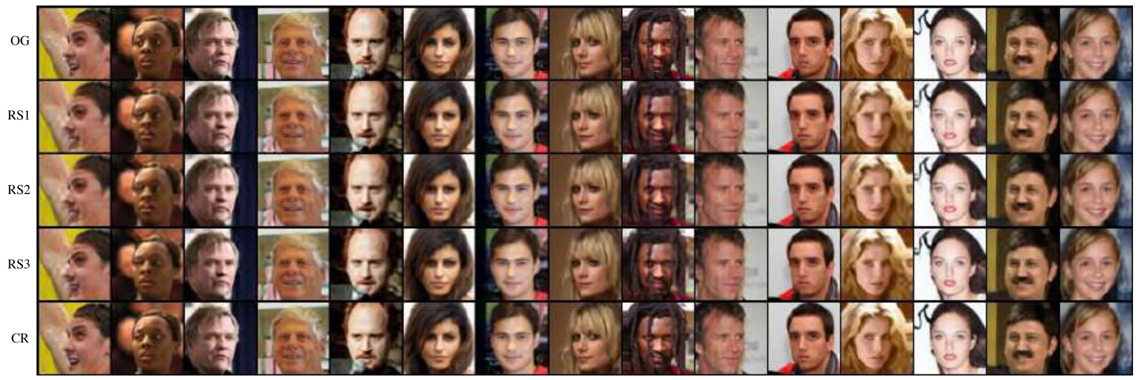
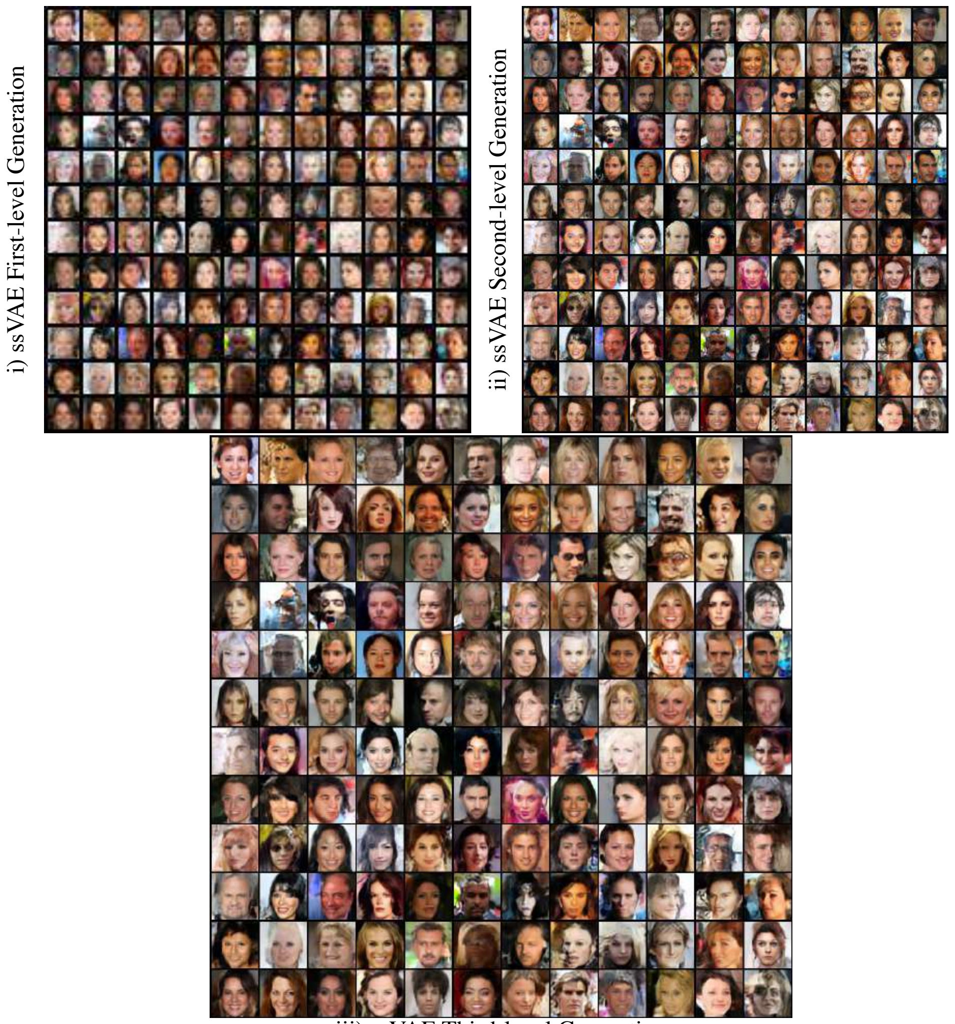


Figure 8.14: Unconditional generations of the ssVAE-sketch trained on CelebA.



Reconstruction of given image (OG) with the 3-level downsampling ssVAE model. On RS1 we use only one latent variable ($\mathbf{u}$) and sample the other ($\mathbf{z}_1, \mathbf{z}_2$) while on RS2 we sample only $\mathbf{z}_2$ and on RS3 we use the exact values from all latent variables. CR indicates the conditional reconstruction using $\mathbf{z}_2$ and $\mathbf{y}_2$.

Figure 8.15: Qualitative results on CelebA of the ssVAE-3lvl.


i) ssVAE First-level Generation
ii) ssVAE Second-level Generation

iii) ssVAE Third-level Generation

Figure 8.16: Unconditional generations of the ssVAE-3lvl trained on CelebA.

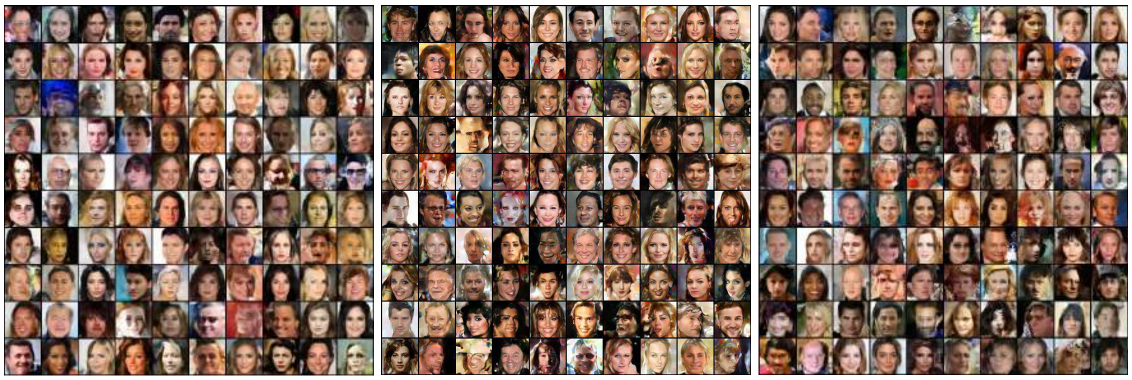
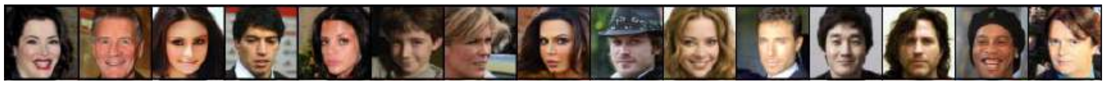
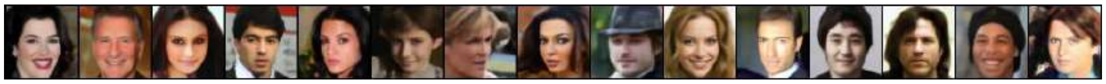


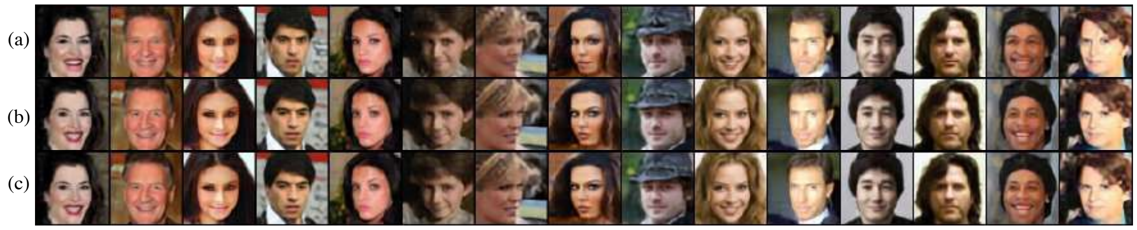
Figure 8.17: Comparison on CelebA unconditional generations. From left to right: VAE with RealNVP prior, ssVAE 2-level sketch, ssVAE 3-level downsample.



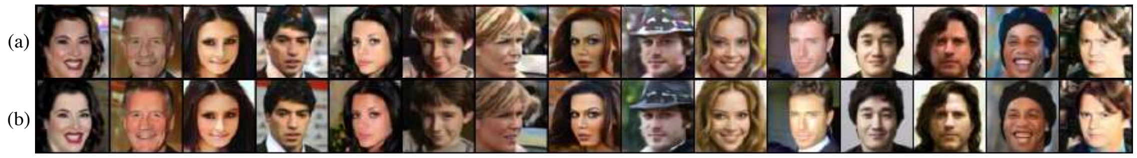
Ground-truth data samples



i) VAE with RealNVP prior reconstructions



ii) ssVAE 3-level downsample model reconstructions using (a) 1 latent code ($\mathbf{u}$)
(b) 2 latent codes ($\mathbf{u}, \mathbf{z}_1$) and (c) all three latent codes ($\mathbf{u}, \mathbf{z}_1, \mathbf{z}_2$).



iii) ssVAE 2-level sketch model reconstructions using (a) 1 latent code ($\mathbf{u}$)
and (b) 2 latent codes ($\mathbf{u}, \mathbf{z}_2$).

Figure 8.18: Comparison on CelebA reconstructions.