Outlier Detection with Hierarchical VAEs and Hamiltonian Monte Carlo

Haoran Peng

Supervisor: Prof. José Miguel Hernández-Lobato

This thesis is submitted for the degree of
Master of Philosophy in Machine Learning and Machine Intelligence

Gonville & Caius College August 2022

Declaration

I, Haoran Peng of Gonville & Caius College, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

The code used in this report is my own, written using standard Python packages.

The word count is 8618, excluding declarations, bibliography, photographs and diagrams, but including tables, footnotes, figure captions and appendices.

Haoran Peng
August 2022

Acknowledgement

I would like to sincerely thank my supervisor Prof. José Miguel Hernández-Lobato for his advice on both theoretical concepts and implementation details throughout this project.

Abstract

For humans to trust machine learning systems to make real-world high-stake decisions, such systems need to know what they do not know. Unsupervised outlier detection is a task which addresses this, where machine learning systems are trained to detect input data which are dissimilar to the training set in an unsupervised setting. Variational autoencoders (VAEs; Kingma & Welling, 2014) have been popular for unsupervised learning and practitioners have used VAEs' reconstruction accuracy as a metric for outlier detection, where a lower reconstruction accuracy suggests the data is likely an outlier. A well-known issue with this method is that VAEs can sometimes reconstruct outliers better than inliers. In addition, Peis et al. (2022) found that VAEs with hierarchical latent variables and VAEs that adopt Hamiltonian Monte Carlo (HMC) for posterior sampling deteriorate outlier detection performance, despite the reconstruction improvements they offer. In this work, we first reimplement VAEs with hierarchical latent variables and HMC and we show that hierarchies bring latent disentanglement and HMC improves reconstruction. We then present a mini survey on the outlier detection problem that summarizes probable causes and solutions. We then apply various outlier detection methods to VAEs, including VAEs with hierarchical latent variables and VAEs with HMC. On the standard set of image-based benchmarks, we find that hierarchical latent variables improve outlier detection while HMC does not seem to affect the performance. We also find that while certain methods achieve better outlier detection results, there are usually trade-offs to be made between performance and the computational resources required.

Table of contents

Chapter 1

Introduction

Questions regarding reliability and robustness of machine learning systems have attracted a lot of attention from practitioners in recent years. Machine learning systems based on artificial neural networks can often predict the wrong answer with high confidence and fail catastrophically when the input data differs from the training set. For example, a neural network that is trained to recognize images of cats and dogs can predict an image of a cat (**inlier**) to be a dog with high confidence; or worse, predict a random image (**outlier**) to be a dog with high confidence.

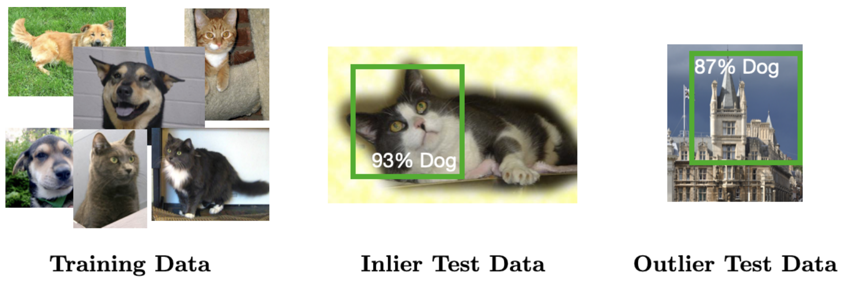

Fig. 1.1 Neural network predicting an image of a cat to be a dog

In this work, we focus on the latter problem which is called **outlier detection** where we want to detect data that is (broadly speaking) different from the training set. Outlier detection can be used in banks to detect abnormal transactions, in autonomous vehicles to alert the driver of unfamiliar driving situations, and to prevent general misuse of machine learning systems.

There are many techniques for outlier detection which Yang et al. (2021) enumerated in their survey paper. In this work, we will experiment with **unsupervised** outlier detection using **variational autoencoders (VAEs)**. VAEs are trained to encode input data into low-dimensional latent representations, and then use the latent representations to reconstruct the input data. The assumption is that if a trained VAE can reconstruct an input data well, then the data is similar to the training data and is probably an inlier; on the contrary, if the VAE cannot reconstruct an input data well, then the data is probably an outlier. It has become widely known recently that this assumption does not always hold and that outliers can sometimes be reconstructed better than inliers. A goal of this work is to find and apply solutions that address this issue.

This work also investigates outlier detection using VAEs with hierarchical latent variables and VAEs that leverage Hamiltonian Monte Carlo (HMC) for posterior sampling. Peis et al. (2022) showed that while HMC improves reconstruction performance, it reduces outlier detection performance. The main goal of this work is to find outlier detection methods that perform well and can retain the gains in reconstruction accuracy offered by hierarchical latent variables and HMC.

The main contributions are as follows:

- In Chapter 2, we provide the necessary theoretical background on VAEs (Kingma & Welling, 2014), hierarchical VAEs (Zhao et al., 2017), and VAEs with HMC (Hoffman, 2017). We unify the mathematical notation and the style of graphical models, and we present the benefits of hierarchical latent variables and HMC for posterior sampling.
- In Chapter 3, we detail our implementation of VAEs, hierarchical VAEs, and VAEs with HMC. We report their reconstruction results and discuss the benefits and drawbacks of the models. The source code is available at https://github.com/GavinPHR/HMC-VAE.
- In Chapter 4, we introduce the problem of outlier detection using VAEs and present a mini survey of papers which propose possible causes and solutions (Choi et al., 2019; Daxberger & Hernández-Lobato, 2019; Denouden et al., 2018; Serrà et al., 2020; Xiao et al., 2020).
- In Chapter 5, we adapt and apply outlier detection methods from the mini survey and show their performance under different VAE models. We conclude that while some methods achieve better results, there are usually trade-offs to be made between performance and the computational resources required.

Chapter 2

Theoretical Background on Variational Autoencoders

This chapter introduces the necessary background on variational autoencoders, variational autoencoders with hierarchical latent variables, and variational autoencoders that leverage Hamiltonian Monte Carlo for posterior sampling.

2.1 Variational Autoencoders

A variational autoencoder (VAE; Kingma & Welling, 2014) is a deep latent generative model that is specified by the distribution $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ where $\mathbf{x}$ is the observed data from a dataset $\mathbf{X}$ containing N i.i.d. samples and $\mathbf{z}$ is the latent variable. It is assumed that both the prior $p(\mathbf{z})$ and the likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ come from some parametric families of distributions whose PDFs are differentiable almost everywhere w.r.t. $\boldsymbol{\theta}$ and $\mathbf{z}$.

Direct optimization of $\boldsymbol{\theta}$ is intractable and a variational posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is introduced to enable optimization of the evidence lower bound (ELBO):

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right] \leq \log p(\mathbf{x}). \tag{2.1}$$

The ELBO is a lower bound of the log marginal likelihood. Maximizing the ELBO attempts to maximize the data marginal likelihood. The ELBO is usually rewritten in the form:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{x}) = \underbrace{\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]}_{\text{Negative Reconstruction Loss}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\text{Regularization}}. \tag{2.2}$$

The first term encourages the model to reconstruct the input as well as possible and the second term regularizes the model by restricting the variational posterior to be close to the prior.

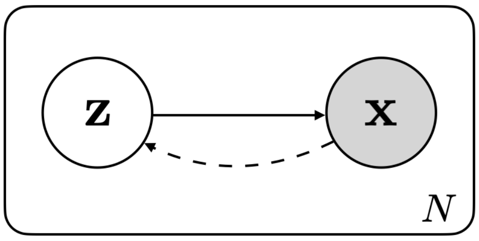

Fig. 2.1 Graphical model of a VAE. The solid line denotes the generative model and the dashed line denotes the inference model.

The graphical model of a VAE is shown in Figure 2.1. In practice, the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are neural networks, and they are jointly optimized via stochastic gradient descent with the optimization objective being the ELBO across the dataset:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi} | \mathbf{X}) \approx \frac{N}{M} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi} | \mathbf{X}^M) = \frac{N}{M} \sum_{\mathbf{x} \in \mathbf{X}^M} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi} | \mathbf{x}) \tag{2.3}$$

where $\mathbf{X}^M$ is a subset of $\mathbf{X}$, containing M of the N data points.

There are some further implementation details from Kingma & Welling (2014) that are widely adopted:

- With high enough batch size M, it is enough to take 1 sample to estimate the expectation in Equation 2.2.
- The prior is chosen to be the standard Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \boldsymbol{I})$ and the variational posterior is chosen to be the factorized Gaussian $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}^2(\mathbf{x})\boldsymbol{I})$. These choices make it possible to evaluate the KL-divergence term analytically.
- The reparameterization trick is used to turn a stochastic node (sampling of $\mathbf{z}$) into a deterministic one, which allow for gradient backpropagation.

With these implementation details, the ELBO can be further simplified to:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi} | \mathbf{x}) \approx \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \tag{2.4}$$

where $\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \odot \boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. We will be working with the simplified form of the ELBO in Equation 2.4 unless otherwise stated.

2.2 Hierarchical VAEs

A basic VAE has only one latent variable. Akin to deep neural networks where shallow layers learn more concrete features and deeper layers learn more abstract features, adding a hierarchy of latent variables to VAEs could achieve a similar result where each latent variable controls a different aspect of the generated data. This is desirable because the latent representations are more disentangled and interpretable. A basic hierarchical variational autoencoder has the following generative process:

$$\begin{aligned} p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L) &= p(\mathbf{x}|\mathbf{z}_1)p(\mathbf{z}_1|\mathbf{z}_2)\cdots p(\mathbf{z}_{L-1}|\mathbf{z}_L)p(\mathbf{z}_L) \\ &= p(\mathbf{x}|\mathbf{z}_1)\left[\prod_{i=1}^{L-1} p(\mathbf{z}_i|\mathbf{z}_{i+1})\right] p(\mathbf{z}_L). \end{aligned} \tag{2.5}$$

And the variational posterior has the form:

$$\begin{aligned} q(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L|\mathbf{x}) &= q(\mathbf{z}_L|\mathbf{z}_{L-1})\cdots q(\mathbf{z}_2|\mathbf{z}_1)q(\mathbf{z}_1|\mathbf{x}) \\ &= \left[\prod_{i=1}^{L-1} q(\mathbf{z}_{i+1}|\mathbf{z}_i)\right] q(\mathbf{z}_1|\mathbf{x}). \end{aligned} \tag{2.6}$$

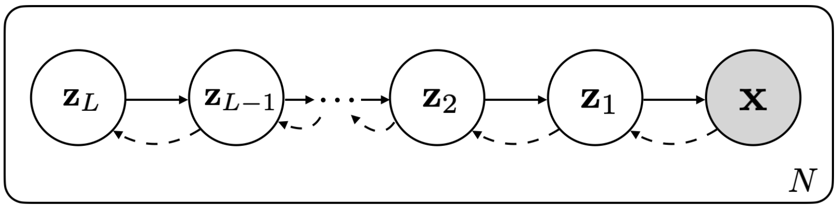

Fig. 2.2 Graphical model of a basic hierarchical VAE. The solid lines denote the generative model and the dashed lines denote the inference model.

The graphical model a basic hierarchical VAE is shown in Figure 2.2. This basic hierarchical structure suffers from the *posterior collapse* problem (Bowman et al., 2016; Burda et al., 2016; Sønderby et al., 2016). The model tends to ignore the high level latent variables (i.e. $\mathbf{z}_L, \mathbf{z}_{L-1}$) and uses only the shallow ones (i.e. $\mathbf{z}_1, \mathbf{z}_2$), making the model difficult to train. To overcome this, various hierarchical structures are proposed, but they are mostly based on the ladder VAE proposed by Sønderby et al. (2016).

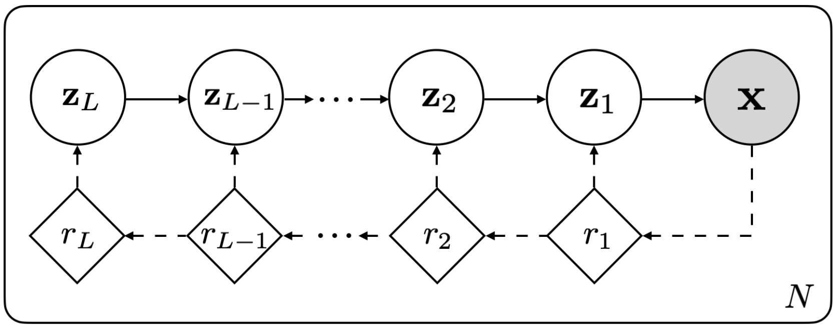

Fig. 2.3 Graphical model of the LVAE. The solid lines denote the generative model and the dashed lines denote the inference model. The diamond nodes are deterministic.

In a ladder VAE (LVAE; shown in Figure 2.3), the inference model and the generative model share the same path and the new variational posterior has the form:

$$\begin{aligned} q(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L|\mathbf{x}) &= q(\mathbf{z}_1|\mathbf{z}_2, \mathbf{x}) \cdots q(\mathbf{z}_{L-1}|\mathbf{z}_L, \mathbf{x}) q(\mathbf{z}_L|\mathbf{x}) \\ &= \left[\prod_{i=1}^{L-1} q(\mathbf{z}_i|\mathbf{z}_{i+1}, \mathbf{x})\right] q(\mathbf{z}_L|\mathbf{x}). \end{aligned} \tag{2.7}$$

Notice the similarity between Equation 2.7 and the generative process in Equation 2.5. Path sharing enables the inference model to have information about the current state of the generative model and thus enables better inference.

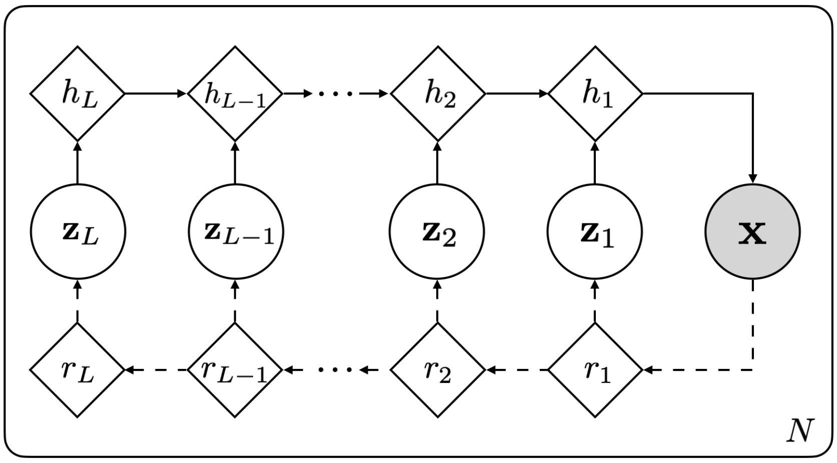

Fig. 2.4 Graphical model of a VLAE. The solid lines denote the generative model and the dashed lines denote the inference model. The diamond nodes are deterministic.

In this work, we will be working with a variation of the LVAE called variational ladder autoencoder (VLAE; Zhao et al., 2017). The graphical model of a VLAE is shown in Figure 2.4. Although there is no hierarchical dependencies between the latent variables, the latent variables are able to learn hierarchical features because they are implicitly embedded in the deterministic paths. This largely removes the posterior collapse problem because there is effectively only one layer of latent variables. In addition, this particular structure makes it possible to incorporate Hamiltonian Monte Carlo into the model in subsequent sections. The generative process and the variational posterior are:

$$p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L) = p(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L) \prod_{i=1}^{L} p(\mathbf{z}_i), \tag{2.8}$$

$$q(\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L|\mathbf{x}) = \prod_{i=1}^{L} q(\mathbf{z}_i|\mathbf{x}). \tag{2.9}$$

And the ELBO has the form:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{x}) = \mathbb{E}_{\mathbf{z}_1,\mathbf{z}_2,\dots,\mathbf{z}_L \sim q_{\boldsymbol{\phi}}(\mathbf{z}_1,\mathbf{z}_2,\dots,\mathbf{z}_L|\mathbf{x})} \left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L)\right] - \sum_{i=1}^{L} D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}) \| p(\mathbf{z}_i)). \tag{2.10}$$

2.3 Generative Modelling with Hamiltonian Monte Carlo

In the previous sections, we used variational inference (VI) to optimize the parameters of VAEs. Markov chain Monte Carlo (MCMC) has long been a competitor to VI. VI is usually faster but there is always a gap between the variational distribution and the true distribution. MCMC on the other hand is slower but is asymptotically exact. Hamiltonian Monte Carlo (HMC) is a MCMC method that is especially well-suited for sampling from continuous distributions with differentiable PDFs[1], given that the PDFs can be evaluated relatively easily. The posterior distribution of the generative model fits the criteria[2] and we can trade computation for improved posterior inference by using HMC instead of VI. We outline the basic HMC procedure in Algorithm 1, though it is worth noting that one can use any HMC variants here (e.g. NUTS). Readers interested in the inner workings of HMC are advised to consult Hoffman & Gelman (2011); Neal (2001).

One problem remains before we can use HMC instead of VI to optimize the parameters of the generative model using SGD. We can take samples from HMC but we cannot

[1]The PDFs can be unnormalized.

[2]The posterior $p(\mathbf{z}|\mathbf{x}) \propto p(\mathbf{x}, \mathbf{z})$ is continuous and we can evaluate and differentiate $p(\mathbf{x}, \mathbf{z})$ w.r.t. $\mathbf{z}$ relatively easily.

Algorithm 1 HMC for sampling from the posterior $p(\mathbf{z}|\mathbf{x})$

```
Input: data $\boldsymbol{x}$, initial latent state $\boldsymbol{z}$, number of HMC iterations $T$, number of leapfrog steps $L$, leapfrog step size $\epsilon$.
for $t = 1$ to $T$ do
    Sample momentum $\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
    $\boldsymbol{r}' = \boldsymbol{r}$
    $\boldsymbol{z}' = \boldsymbol{z}$
    for $l = 1$ to $L$ do
        $\boldsymbol{r}' = \boldsymbol{r}' + \frac{\epsilon}{2} \cdot \nabla_{\boldsymbol{z}'} \log p(\boldsymbol{x}, \boldsymbol{z}')$
        $\boldsymbol{z}' = \boldsymbol{z}' + \epsilon \cdot \boldsymbol{r}'$
        $\boldsymbol{r}' = \boldsymbol{r}' + \frac{\epsilon}{2} \cdot \nabla_{\boldsymbol{z}'} \log p(\boldsymbol{x}, \boldsymbol{z}')$
    end for
    Accept with probability $\min(1, \exp(-H(\boldsymbol{z}', \boldsymbol{r}')) + H(\boldsymbol{z}, \boldsymbol{r}))$
    where $H(\boldsymbol{z}, \boldsymbol{r}) = -\log p(\boldsymbol{x}, \boldsymbol{z}) + \frac{1}{2}\boldsymbol{r}^T\boldsymbol{r}$
        $\boldsymbol{z} = \boldsymbol{z}'$
    Otherwise reject $\boldsymbol{z}'$
        $\boldsymbol{z} = \boldsymbol{z}$
end for
return $\boldsymbol{z}$
```

evaluate the density which means we cannot evaluate the data marginal likelihood nor the ELBO. Thus we use a different maximization objective shown below:

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right]. \tag{2.11}$$

The difference between this new objective and the ELBO is a differential entropy term $\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x})}\left[-\log p(\mathbf{z}|\mathbf{x})\right]$ which serves as a regularizer and specifies the posterior should not be too concentrated. In spite of that, Hoffman (2017) and Peis et al. (2022) used this objective and showed that it is effective.

2.4 Hierarchical VAE with HMC

In the previous section, we showed how to train the generative model with HMC. While the procedure works, it is not very practical for a few reasons:

- For each data point, we need to run HMC from a random initial latent state and it can take a long time for the chain to converge sufficiently close to the target distribution.

- It can take many epochs of SGD for the parameters of the generative model to converge and it exacerbates the problem with computation time.
- The ability to generate new data is lost because we cannot sample from HMC without a target data $\boldsymbol{x}$.

As Hoffman (2017) points out, the addition of the inference model can alleviate these issues. Starting HMC from a sample from the variational posterior is much better than starting from a random position and can make the chain converge much faster. Peis et al. (2022) showed that if we train the model using only VI initially and then add in HMC, only a few more epochs are needed for convergence. Also, the ability to generate new data is regained because we can sample from the variational posterior.

To formalize this, let the variational distribution be $q^0_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and let the distribution of the HMC chain after T steps be $q^T(\mathbf{z}|\mathbf{x},\mathbf{z}^0)$[3]. The VAE (inference model and generative model) is trained to maximize the ELBO until convergence:

$$\mathcal{L}_{\text{VI}}(\boldsymbol{\theta},\boldsymbol{\phi}|\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q^0_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - D_{\text{KL}}(q^0_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})). \tag{2.12}$$

Then the inference model is continued to be optimized using the ELBO while the generative model is optimized using the objective stated in Equation 2.11 and restated below:

$$\mathcal{L}_{\text{HMC}}(\boldsymbol{\theta}|\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q^T(\mathbf{z}|\mathbf{x},\mathbf{z}^0)}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})\right] \quad \text{where} \quad \mathbf{z}^0 \sim q^0_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}). \tag{2.13}$$

It is straightforward to add HMC to a hierarchical VAE model like the one shown in Figure 2.4. As we discussed before, despite that model having multiple latent variables that can learn hierarchical features, there are no hierarchical dependencies between the latent variables. We can thus treat the latent variables as one entity during the HMC stage. For simplicity of notation in later chapters, we will overload $\mathbf{z}$ to denote both a single latent variable and a collection of them, and we will use **HMC-VAE** when referring to VAEs with HMC.

[3] $q^T(\mathbf{z}|\mathbf{x},\mathbf{z}^0)$ instead of the true posterior $p(\mathbf{z}|\mathbf{x})$ because we cannot guarantee convergence after T HMC iterations.

Chapter 3

VAE Implementation Details and Results

In Chapter 2, we discussed the theoretical background on VAEs and HMC-VAEs. In this chapter, we will establish the implementation details which underpin all the experiments in later chapters. Code is available at https://github.com/GavinPHR/HMC-VAE.

3.1 Datasets and Likelihood Function

The datasets we will use are MNIST (LeCun et al., 2010), FashionMNIST (Xiao et al., 2017), CIFAR10 (Krizhevsky, 2009), SVHN (Netzer et al., 2011). These are all image datasets which are either coloured (3 channels) or in greyscale (1 channel). All the images are rescaled to 32×32 pixels and each pixel takes on an integer value in the interval $[0 \mathinner{\ldotp\ldotp} 255]$.

We will use the standard benchmarks for the VAE outlier detection task:

- FashionMNIST (inlier) vs MNIST (outlier)
- CIFAR10 (inlier) vs SVHN (outlier)

Furthermore, we will follow Nalisnick et al. (2019b) and Xiao et al. (2020) and use a mixture of categorical distributions as the data likelihood:

$$p(\mathbf{x}|\mathbf{z}) = \prod_{\text{pixels } \mathrm{x} \in \mathbf{x}} p(\mathrm{x}|\mathbf{z}) \tag{3.1}$$

where x is categorically distributed with 256 categories.

3.2 VAEs with Convolutional Neural Networks

With image datasets, it is natural to use convolutional neural networks. The network architecture we used is akin to that in Xiao et al. (2020)[1]. We chose this architecture because of its simplicity and because it allows us to compare our subsequent outlier detection results with Xiao et al. (2020), which is the state-of-the-art for these specific datasets to our knowledge.

Table 3.1 VAE (1 latent layer) encoder and decoder specifications. C is the channel size, F is the number of filters, and D is the latent dimension. BN is batch norm and ReLU is rectified linear unit.

Encoder	Decoder
Input $C \times 32 \times 32$	Input $D \times 1 \times 1$
4×4 Conv2d$_F$ Stride 2, BN, ReLU	4×4 ConvTranspose2d$_{4 \times F}$ Stride 1
4×4 Conv2d$_{2 \times F}$ Stride 2, BN, ReLU	4×4 ConvTranspose2d$_{2 \times F}$ Stride 2, BN, ReLU
4×4 Conv2d$_{4 \times F}$ Stride 2, BN, ReLU	4×4 ConvTranspose2d$_F$ Stride 2, BN, ReLU
4×4 Conv2d$_{2 \times D}$ Stride 1	4×4 ConvTranspose2d$_{C \times 256}$ Stride 2

The encoder/decoder[2] architecture used throughout this work is shown in Table 3.1. The encoder and the decoder are symmetric and each contains 3 main convolutional blocks. Visualizations of the encoder/decoder are shown in Figure 3.1 and 3.2[3]. We set $C = 1, F = 32$ for FashionMNIST and $C = 3, F = 64$ for CIFAR10.

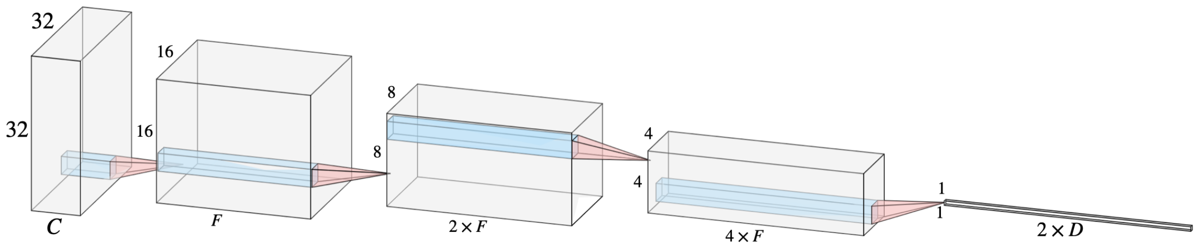

Fig. 3.1 Encoder of a single latent layer convolutional VAE. C is the channel size, F is the number of filters, and D is the latent dimension.

The encoder encodes the input into 2 D-dimensional vectors which serves as the mean and the variance for the D-dimensional factorized Gaussian variational posterior. The decoder takes in the D-dimensional sample and reconstructs the input data. We set $D = 90$ for subsequent experiments.

[1]With minor modifications to fit hierarchical latent variables.

[2]We use encoder/decoder instead of inference/generative model to emphasize these are deterministic neural networks.

[3]Figures are produced using tools developed by (LeNail, 2019)

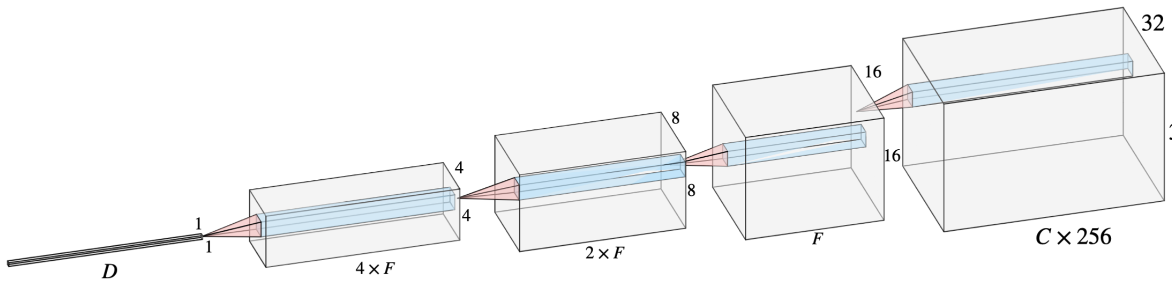

Fig. 3.2 Decoder of a single latent layer convolutional VAE. C is the channel size, F is the number of filters, and D is the latent dimension.

3.3 Adding Hierarchical Latent Variables

The VAE architecture has 3 main convolutional blocks each in its encoder and decoder. Intuitively, each block captures features at different hierarchies. We can introduce a hierarchy of latent variables by projecting each interim tensor into parameters of the variational distribution.

Concretely, let the interim tensor after the i-th convolutional block have dimension $C \times H \times W$, we can project it using a $H \times W$ Conv2d block with $2 \times D_i$ output channels and stride 1. The result is a $2 \times D_i$ vector which serves as the mean and the variance for the variational distribution at latent layer i. The sample taken from the variational distribution can be brought back to dimension $C \times H \times W$ using a ConvTranspose2d block with the same parameters, then be added to the interim tensor from the corresponding decoding block. The graphical model in Figure 2.4 describes this architecture exactly, where each solid line represents a ConvTranspose2d block, each dashed line represents a Conv2d block, and each diamond node represents an interim tensor. Consult the code at https://github.com/GavinPHR/HMC-VAE for more details.

In subsequent experiments, we will use 3 latent layers in the hierarchical VAEs, each after a convolution block. For maximum comparability between the single layer and the hierarchical VAEs, we match the total number of latent units by setting the latent dimensions to be $D_1 = 30, D_2 = 30, D_3 = 30$.

3.4 HMC Parameter Setting and Training Time

There are 3 parameters to be set for HMC. The number of iterations T, the number of leapfrog steps per iteration L, and the step size ϵ. The values chosen for these parameter are heavily dependent on the compute budget. If the compute budget is high, one should choose higher T, L, and vice versa. We set $T = 10, L = 5$ in our experiments. For ϵ, we follow previous work (Hoffman & Gelman, 2011; Neal, 2001) and tune it automatically so the acceptance rate (averaged across batch) is 65%. We achieve this by increasing ϵ by 5% if the acceptance rate is too high and decreasing by 5% otherwise.

For completeness, we observed a 50 times slowdown when training with HMC compared to training with VI. This will depend on the batch size and the hardware used, but we speculate the training times scale linearly with $T \times L$ which makes sense because a pass through the decoder is required at each leapfrog step. One should consider whether the benefits brought by HMC (Section 3.6) are worth the significant increase in training and prediction time.

3.5 Model Optimization

The models are all optimized using Adam (Kingma & Ba, 2015) with 5e-4 learning rate. The models are trained with the ELBO objective for the first 100 epochs. Then they are trained for an additional 3 epochs using HMC for posterior sampling, with the ELBO objective for the encoders and the HMC objective for the decoders.

3.6 Results and Discussion

A common metric for comparing VAEs is the log marginal likelihood $\log p(\mathbf{x})$. Approximating the log likelihood is relatively straightforward with the variational posterior (Burda et al., 2016) but poses a challenge with HMC because we cannot evaluate the sample likelihood under the posterior distribution. Instead, we will simply compare the models using reconstruction accuracy $\mathbb{E}_{\mathbf{z}}\left[\log p(\mathbf{x}|\mathbf{z})\right]$.

Table 3.2 Reconstruction accuracy of VAEs trained with the ELBO objective.

	$\mathbb{E}_{\mathbf{z}\sim q^0(\mathbf{z}\|\mathbf{x})}\left[\log p(\mathbf{x}\|\mathbf{z})\right]$	
	1 layer	3 layers
FashionMNIST	-2143 ± 8	-2189 ± 5
CIFAR10	-12572 ± 71	-12801 ± 33

Table 3.2 shows the reconstruction accuracy of models trained with the ELBO objective only and Table 3.3 shows the reconstruction accuracy of models that are continued to be trained with HMC. There are agreements and discrepancies between our results and Peis et al. (2022).

Table 3.3 Reconstruction accuracy of VAEs trained with the ELBO objective for the encoder and the HMC objective for the decoder. We show the reconstruction accuracy both using samples from the variational posterior and from HMC.

	$\mathbb{E}_{\mathbf{z}\sim q^0(\mathbf{z}\|\mathbf{x})}[\log p(\mathbf{x}\|\mathbf{z})]$		$\mathbb{E}_{\mathbf{z}\sim q^T(\mathbf{z}\|\mathbf{x})}[\log p(\mathbf{x}\|\mathbf{z})]$	
	1 layer	3 layers	1 layer	3 layers
FashionMNIST	-2188 ± 16	-2229 ± 15	$-\mathbf{2060} \pm \mathbf{5}$	-2116 ± 5
CIFAR10	-12770 ± 45	-12926 ± 51	$-\mathbf{12303} \pm \mathbf{12}$	-12476 ± 19

Firstly, HMC-VAEs achieve better reconstruction accuracy than VAEs, but the difference is not significant qualitatively. See Figure 3.3 for some reconstruction examples. The improvements observed in Peis et al. (2022) are more significant and we suspect the difference comes from the neural network architecture. HMC-VAE will only be much better if the latent distribution is very different from a Gaussian, but our convolutional networks (Peis et al. (2022) used fully-connected networks) use batch normalization layers that tend to keep layer outputs Gaussian-like.

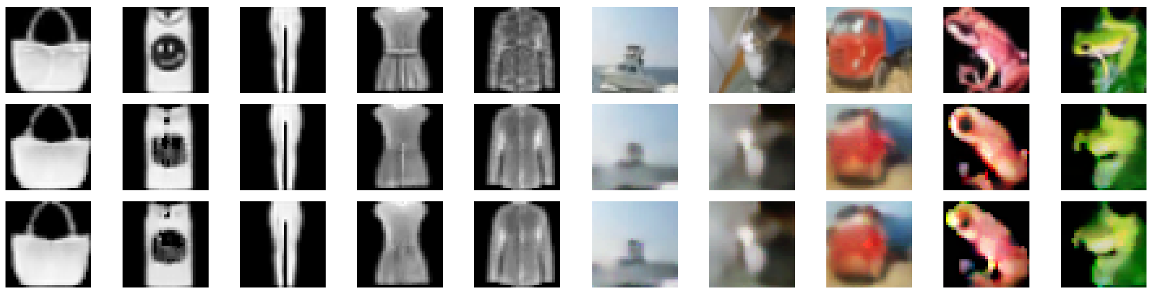
Fig. 3.3 Top: Original Images; Middle: reconstructions using a VAE; Bottom: reconstructions using an HMC-VAE

Secondly, adding more latent layers decreases reconstruction accuracy in our experiments which contradicts Peis et al. (2022) which saw improved reconstruction. Although they used more latent units in their hierarchical VAEs while we are keeping the same number of latent units[4]. This discrepancy can also be due to the difference in network architectures.

With that said, hierarchical latent variables have their benefits. As Zhao et al. (2017) pointed out, each latent layer learns somewhat disentangled features and we visualize this in Figure 3.4. We observe that the first layer controls features like foreground colour, the second layer controls features like shape and saturation, while the last layer seems to

[4]It is unclear how hierarchical and single-layer models can be fairly compared because a hierarchical neural network model almost necessarily has more capacity than a single-layer one and that we would normally expect to use more latent units too.

control the identity (class/label of the image). The features go from concrete to abstract as the layer gets deeper which is what we would expect. Depending on the downstream task, hierarchical latent variables and disentangled latent states can be quite beneficial - we will show that they do indeed improve outlier detection performance in Chapter 5.

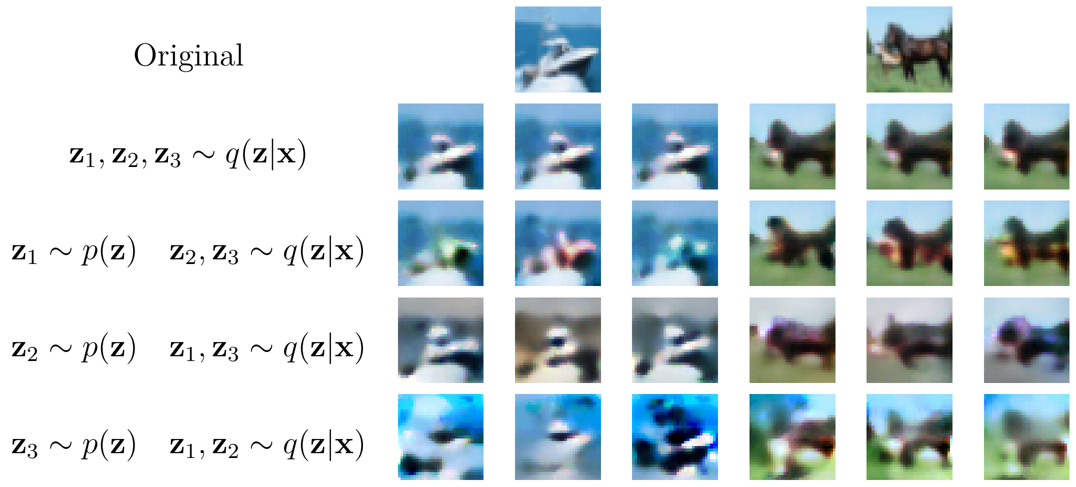
Original
$\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3 \sim q(\mathbf{z}|\mathbf{x})$
$\mathbf{z}_1 \sim p(\mathbf{z}) \quad \mathbf{z}_2, \mathbf{z}_3 \sim q(\mathbf{z}|\mathbf{x})$
$\mathbf{z}_2 \sim p(\mathbf{z}) \quad \mathbf{z}_1, \mathbf{z}_3 \sim q(\mathbf{z}|\mathbf{x})$
$\mathbf{z}_3 \sim p(\mathbf{z}) \quad \mathbf{z}_1, \mathbf{z}_2 \sim q(\mathbf{z}|\mathbf{x})$

Fig. 3.4 Example reconstructions from a hierarchical VAE. Top row are the original images and the second row are normal reconstructions. For each row after that, one of the latent variables is sampled from the prior. It shows each latent variable learns different features that control different aspects of the reconstructions.

Chapter 4

Outlier Detection Problem Statement

There are three broad categories of outlier detection techniques (Chandola et al., 2009). Supervised techniques where the data are labelled as inliers or outliers; semi-supervised techniques where the data are partially labelled; and unsupervised techniques where the data are unlabelled. Unsupervised techniques are more commonly used either because it is impossible to define what an outlier is, or because it is infeasible to collect enough labelled data. VAEs being some of the most popular models for unsupervised learning, it is natural to apply them to unsupervised outlier detection.

4.1 Outlier Detection with Reconstruction Accuracy

The most straightforward way to apply VAEs in outlier detection is through the reconstruction accuracy:

$$\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p(\mathbf{x}|\mathbf{z})\right]. \tag{4.1}$$

The assumption is that if a trained VAE can reconstruct an input data $\boldsymbol{x}$ well, then $\boldsymbol{x}$ is similar to the training data and is probably an inlier, otherwise $\boldsymbol{x}$ is likely an outlier. While this works in some cases, it has been established that VAEs sometimes reconstruct outliers better than inliers. In Figure 4.1 (Left), we show that a VAE trained on FashionMNIST can reconstruct MNIST test data much better overall than FashionMNIST test data. Figure 4.1 (Right) shows a similar story for CIFAR10 vs SVHN.

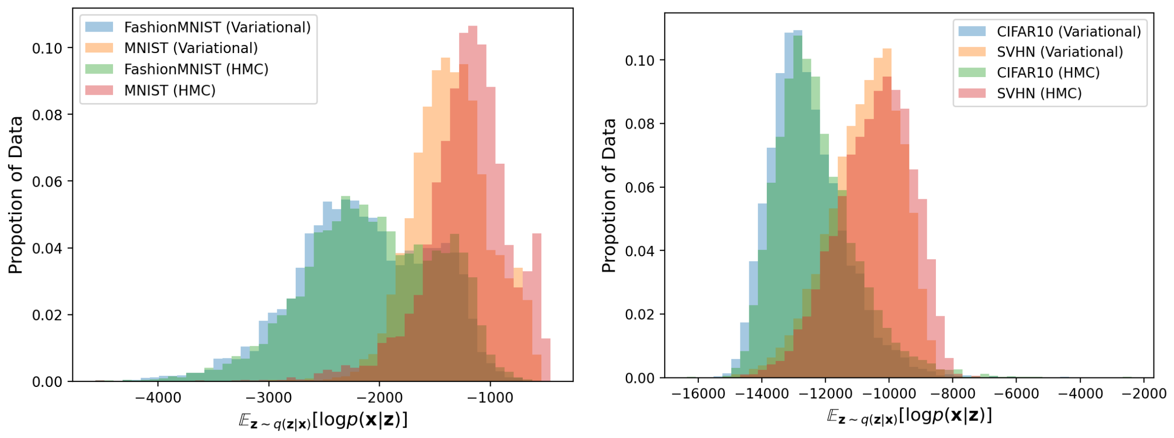

Fig. 4.1 Reconstruction accuracy of (Left) FashionMNIST and MNIST test data on VAEs trained on FashionMNIST (Right) CIFAR10 and SVHN test data on VAEs trained on CIFAR10. The outliers are reconstructed better in both cases. HMC-VAE improves reconstruction accuracy but does not really improve outlier detection performance.

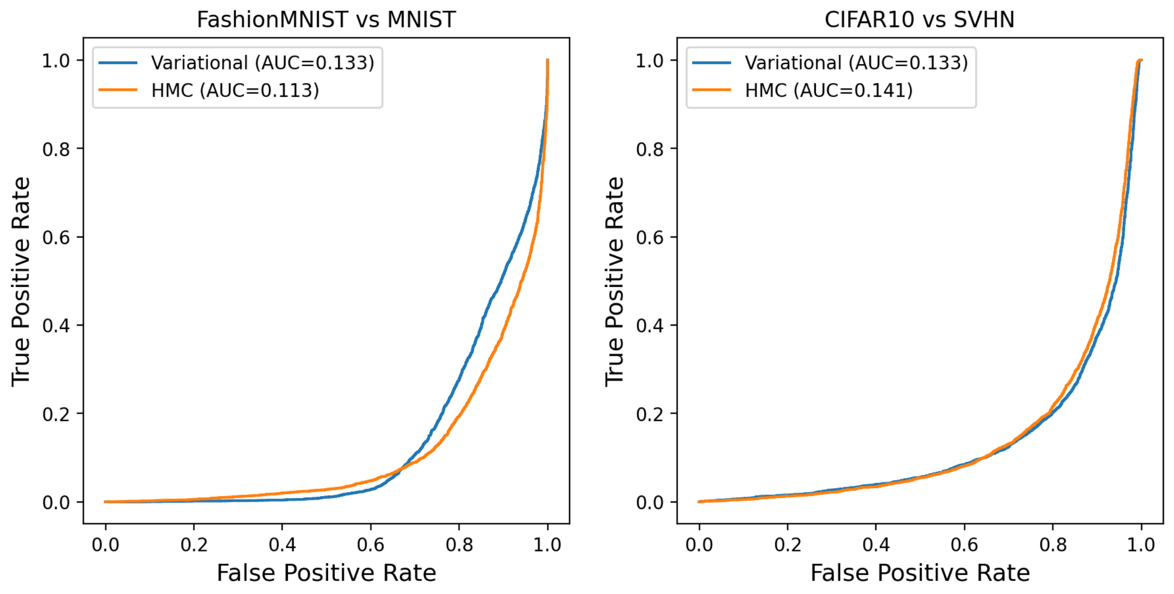

Fig. 4.2 ROC curve of the (Left) FashionMNIST vs MNIST task (Right) CIFAR10 vs SVHN task. We can see that using HMC for posterior sampling does not really improve outlier detection, it actually decreases the performance of a task.

It is worth noting that using HMC for posterior sampling does not improve outlier detection, despite the gain achieved in reconstruction accuracy. This is because using HMC improves reconstruction for both inliers and outliers. It can be especially troublesome when the improvement of outliers are greater than that of inliers (i.e. outlier detection performance becomes worse with HMC).

A standard metric for comparing outlier detection performance is the receiver operating characteristics (ROC) curve. The curve illustrates a model's ability to detect outliers (i.e. true positive and false positive rates) as the decision threshold is varied. We want the models to have high true positive rates and low false positive rates. Area under curve (AUROC) is used as a metric to quantify this, an AUROC value of 1 means the inliers and the outliers are fully separable and an AUROC value of 0.5 is equivalent to guessing randomly. Figure 4.2 shows the ROC curves and AUROC values for FashionMNIST vs MNIST and CIFAR10 vs SVHN. The models' performance under the reconstruction accuracy assumption is much worse than just guessing randomly.

4.2 VAE Outlier Detection Mini Survey

We have stated the problem of outlier detection using reconstruction accuracy, this section presents a mini survey on approaches that circumvent this problem. The papers in this survey approach the problem from different perspectives and propose different solutions. These solutions will be applied on our models in Chapter 5.

4.2.1 Denouden et al. (2018)

This paper suggested that the problem arises from the latent manifold learnt by the VAEs. As Figure 4.3 shows, the inliers' latent representations lie close to this manifold which means they can be reconstructed well. For the outliers' latent representations, while some are far away from the inlier representations, others can still lie close to the manifold which means they can be reconstructed well too.

Their solution is to use a metric that is a hybrid between the reconstruction error and the Mahalanobis distance[1] between the latent representation $\boldsymbol{z}$ of the input $\boldsymbol{x}$ and the training empirical latent distribution Q_{train}:

$$\alpha \cdot \log p(\boldsymbol{x}|\boldsymbol{z}) + \beta \cdot \text{Mahalanobis}(\boldsymbol{z}, Q_{\text{train}}) \tag{4.2}$$

where α, β are hyper-parameters. The smaller this metric is, the more likely the input is an outlier.

The paper demonstrated good results but their benchmarks were too simple with inliers and outliers already well separated using reconstruction accuracy. For more challenging

[1]Mahalanobis distance measures the distance between a (test) point and a (empirical training) distribution that takes correlation into account.

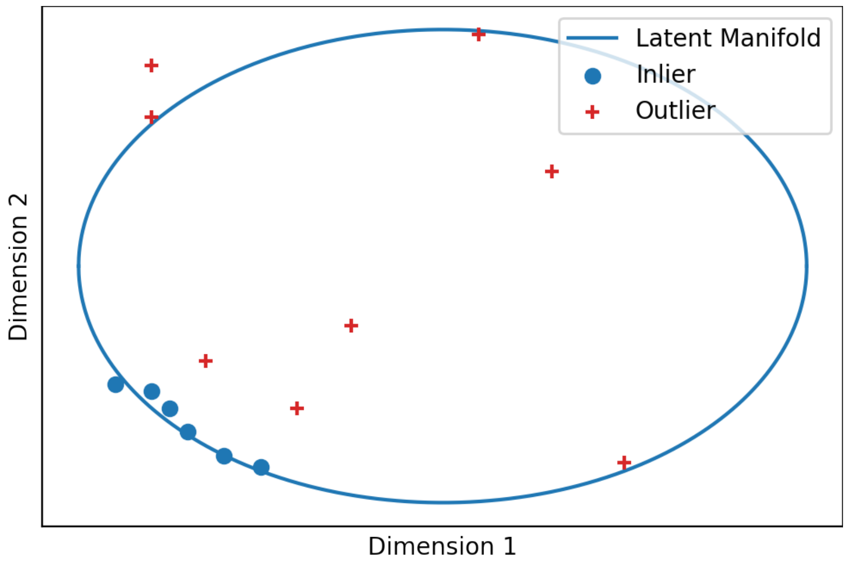

Fig. 4.3 Inlier representations lie near the manifold and can be reconstructed well. Some outlier representations also lie close to the manifold (which means they can be reconstructed well) but their distance from the inlier representations are very far.

benchmarks, where outliers are reconstructed much better than inliers on average, we can anticipate that including the reconstruction accuracy $\log p(\boldsymbol{x}|\boldsymbol{z})$ in the metric can only contribute negatively.

4.2.2 Xiao et al. (2020)

This paper makes very few assumption regarding why the outliers can be reconstructed better than inliers. The paper instead proposed a solution based on continued optimization of the model. A VAE optimizes the ELBO amortized across the whole dataset. This means for any one specific data point, the VAE can be optimized further to improve that data point's likelihood. For an inlier data point, the improvement after additional optimization should not be substantial because the model is already trained on inlier training data. While for an outlier data point, the improvement should be much larger. The authors named the difference between likelihoods before and after additional training **likelihood regret**. The algorithm for computing likelihood regret is shown in Algorithm 2.

This algorithm achieves state-of-the-art results on standard benchmarks. Though its drawbacks are also evident: it is computationally expensive as the model needs to be optimized further for every single test data.

Algorithm 2 Computing likelihood regret

```
**Input:** test data $\boldsymbol{x}$, trained VAE with parameters $\boldsymbol{\phi}_*, \boldsymbol{\theta}_*$, number of additional optimization steps $S$
$P_{\text{VAE}} = \log p(\boldsymbol{x})$ under $\boldsymbol{\phi}_*, \boldsymbol{\theta}_*$
$\boldsymbol{\phi} = \boldsymbol{\phi}_*$
**for** $t = 1$ **to** $S$ **do**
    $\boldsymbol{\phi} =$ optimize $\boldsymbol{\phi}$ to improve $\log p(\boldsymbol{x})$
**end for**
$P_{\text{OPT}} = \log p(\boldsymbol{x})$ under $\boldsymbol{\phi}, \boldsymbol{\theta}_*$
**return** $P_{\text{OPT}} - P_{\text{VAE}}$
```

4.2.3 Serrà et al. (2020)

This paper suggested that the issue is not with the model but rather with the input data. The proposition is that outlier inputs of lower complexity tend to have higher likelihoods under the model. For example, in the FashionMNIST vs MNIST task, MNIST data are of much lower complexity than FashionMNIST and thus MNIST data have higher likelihoods and can be reconstructed better. The paper proposed a new metric that penalizes the negative marginal likelihood with a complexity measure L:

$$-\log p(\boldsymbol{x}) - L(\boldsymbol{x}). \tag{4.3}$$

The higher this metric is, the more likely the input is an outlier. The authors acknowledged that it is tricky to find a good complexity measure L that works consistently - the paper set L to be the size of the image data after compression. When the input is random noise, Xiao et al. (2020) pointed out that the complexity penalty can become so big and makes the metric unreliable.

4.2.4 Choi et al. (2019); Daxberger & Hernández-Lobato (2019)

These two papers both proposed an ensemble-based approach where N sets of decoder parameters $\boldsymbol{\theta}_i$ are required. The assumption is that under different sets of parameters $\boldsymbol{\theta}_i$, the likelihoods of an inlier should have low variance and the likelihoods of an outlier should have high variance (despite the possibility that it can have a higher likelihoods).

To put this idea into concrete metrics, Choi et al. (2019) proposed to use the Watanabe Akaike Information Criterion (WAIC) that combines the mean and the variance of log likelihoods under different parameters:

$$\mathbb{E}_{\boldsymbol{\theta}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] - \mathrm{Var}_{\boldsymbol{\theta}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right]. \tag{4.4}$$

And Daxberger & Hernández-Lobato (2019) proposed to use a metric that is similar to the inverse of variance:

$$\frac{1}{\sum_{i=1}^{N} w_{\boldsymbol{\theta}_i}^2} \quad \text{where} \quad w_{\boldsymbol{\theta}_i} = \frac{p_{\boldsymbol{\theta}_i}(\mathbf{x})}{\sum_{j=1}^{N} p_{\boldsymbol{\theta}_j}(\mathbf{x})}. \tag{4.5}$$

For both of the metrics, the lower the value the more likely the input is an outlier.

To get the ensemble, Choi et al. (2019) trained N distinct models from scratch using SGD which is very expensive; Daxberger & Hernández-Lobato (2019) sampled the parameters using stochastic-gradient Hamiltonian Monte Carlo (SGHMC) which only requires one training pass but the performance of the individual models are significantly worse.

4.2.5 Additional Work and Discussions

Additional work that are well-cited include:

- Ren et al. (2019) noticed that the background of the outlier image contributes significantly to the likelihood. For example, in the FashionMNIST vs MNIST case, it is easy for the model to reconstruct the backgrounds which are mostly black pixels. To combat this, they proposed the method called likelihood ratios where an additional background model is trained to reduce the background's influence on likelihood. As the authors pointed out, this method does not work well if the foreground and the background do not separate easily.

- Nalisnick et al. (2019a) conjectured that high outlier likelihoods are a result of typicality. The inlier representations reside in the typical set of the latent distribution, which may not intersect with the region with the highest density. Instead of likelihoods, the paper proposed to use a hypothesis test for typicality - an input is an outlier if it resides far from the typical set.

- Nalisnick et al. (2019b) presented some theoretical analysis and showed that likelihood alone is not enough to distinguish inliers and outliers. They also showed that this issue does not only concern VAEs but deep generative models in general.

- Hendrycks et al. (2019) utilized an auxiliary dataset that is disjoint from both the inlier and outlier datasets. They used the auxiliary data as outliers during training and penalized the model if they are reconstructed well. It was shown that if the auxiliary dataset is well-chosen (which is not trivial), the model can generalize to the true outlier dataset.

To summarize this mini survey, there is no consensus on why outliers have higher likelihoods and there is no solution that is significantly better in both performance and computation cost. The solutions either do not depend on likelihoods at all, choose to apply some form of penalties to the likelihoods, or avoid direct comparison of likelihoods. The best solution certainly depends on the task at hand and the amount of computational resource available.

Chapter 5

Outlier Detection Experiments and Results

In Chapter 4, we stated the issue with outlier detection using VAEs and presented solutions in a mini survey. In this chapter, we will apply the solutions and investigate their effectiveness when used in combination with hierarchical latent variables and HMC.

5.1 Datasets and Benchmarks

In addition to the datasets mentioned in Chapter 3 (FashionMNIST, MNIST, CIFAR10, SVHN), we create two synthetic outlier datasets - a noise dataset where each pixel is uniformly sampled and a constant dataset where every pixel takes on the same uniformly sampled value.

With the synthetic datasets, we extend the standard benchmarks to contain the following 6 tasks:

- FashionMNIST (inlier) vs MNIST, Noise, Constant (outliers)
- CIFAR10 (inlier) vs SVHN, Noise, Constant (outliers)

These additions should make experimental results more generalizable and convincing.

5.2 Methods Based on Distance Measures

Distance based methods are intuitive and straightforward to implement. Given the latent states of the training set $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ and the latent state of a test input $\boldsymbol{z}_{\text{test}}$, we will try two different metrics for determining whether $\boldsymbol{z}_{\text{test}}$ is an outlier:

1. **Mean k-NN**: Mean L^2 distance from $\boldsymbol{z}_{\text{test}}$ to its k-nearest neighbours in $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$.
2. **Mahalanobis**: Mahalanobis distance between $\boldsymbol{z}_{\text{test}}$ and the empirical distribution of the training latent states $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$.[1]

For each metric, the higher the value, the more likely $\boldsymbol{z}_{\text{test}}$ is an outlier. Additionally, we choose $k = 20$ for the experiments.

Note that these methods can be applied in the input space as well as in the latent space. We choose to apply the methods in the latent space because it is less dependent on the input distributions which makes the conclusion more generalizable and it is computationally much faster at lower dimension.

Table 5.1 AUROC values using distance based methods (FashionMNIST vs others)

FashionMNIST vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
		Mean k-NN		
MNIST	**0.993**	**0.993**	0.921	0.973
Noise	0.674	**0.994**	0.382	0.359
Constant	**0.975**	0.874	0.944	0.845
		Mahalanobis		
MNIST	0.988	**0.992**	0.914	0.972
Noise	0.434	**0.997**	0.164	0.375
Constant	**0.980**	0.896	0.973	0.887

Table 5.1 and 5.2 show AUROC values for FashionMNIST vs others and CIFAR10 vs others, respectively. We can observe that:

- FashionMNIST vs others are easier than CIFAR10 vs others. If we only care about the two main tasks (FashionMNIST vs MNIST and CIFAR10 vs SVHN), either of the methods paired with HMC-VAEs basically achieves state-of-the-art results.
- Noise and Constant outliers are more difficult to detect. It is especially confounding when the AUROC value is close to 0 for some tasks. This means almost all outlier

[1] Denouden et al. (2018) combined Mahalanobis distance with data likelihood. We are not doing that because the outliers have overall much higher likelihoods and adding them only decreases performance.

Table 5.2 AUROC values using distance based methods (CIFAR10 vs others)

CIFAR10 vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
		Mean k-NN		
SVHN	0.726	0.769	**0.908**	0.898
Noise	0.000	0.009	0.001	0.001
Constant	0.101	0.211	**0.917**	0.848
		Mahalanobis		
SVHN	0.730	0.728	**0.930**	0.914
Noise	0.000	0.007	0.000	0.005
Constant	0.008	0.023	0.630	**0.678**

test data are closer to training data than inlier test data. This phenomenon should be studied further in future work.

- It is inconclusive whether it helps to use more layers and/or HMC as the results are contradictory.

The weakness of these methods is certainly their consistency - they can perform incredibly well with some datasets and fail terribly with others.

5.3 Methods Based on Reconstruction Accuracy

Many of the methods introduced in the mini survey in Chapter 4 are based on log marginal likelihood. Since it is difficult to evaluate marginal likelihood with HMC models, we will adapt and apply those methods to use reconstruction accuracy instead in this section.

5.3.1 Reconstruction Accuracy

For reference, we list outlier detection performance using reconstruction accuracy only[2], with the assumption that outliers should have lower reconstruction accuracy.

Table 5.3 and 5.4 show the AUROC values of the benchmarks. As expected, FashionMNIST vs MNIST and CIFAR10 vs SVHN both have terrible performance. Unsurprisingly, Noise outliers can be detected very well because they are much more difficult to reconstruct. For Constant outliers, the results indicate that VAEs trained on CIFAR10 are

[2]We are not using data marginal likelihoods because they are not straightforward to estimate for models with HMC. Though a popular method for estimating them is shown in Neal (2001).

much better at reconstructing Constant than VAEs trained on FashionMNIST. We suspect this is because VAEs trained on FashionMNIST have learnt to bias the background pixels to 0 (black), while VAEs trained on CIFAR10 have learnt to handle different colour values.

Table 5.3 AUROC values using only reconstruction accuracy (FashionMNIST vs others)

FashionMNIST vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
MNIST	0.133	0.179	0.113	0.149
Noise	**1.000**	**1.000**	**1.000**	**1.000**
Constant	0.976	**0.984**	0.871	0.892

Table 5.4 AUROC values using only reconstruction accuracy (CIFAR10 vs others)

CIFAR10 vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
SVHN	0.133	0.141	0.141	0.133
Noise	**1.000**	**1.000**	**1.000**	**1.000**
Constant	0.002	0.003	0.004	0.006

5.3.2 HMC Improvement

HMC Improvement is an adaptation of Xiao et al. (2020) (Likelihood Regret) for HMC-VAEs. Xiao et al. (2020) stipulated that if $\boldsymbol{x}$ is an outlier, then we should expect a bigger improvement in its likelihood if the VAE is trained further on $\boldsymbol{x}$ only. A similar improvement can be extracted under HMC-VAEs without further training:

$$\text{HMC Improvement}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim q^T(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q^0_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]. \tag{5.1}$$

HMC Improvement is the difference between reconstruction accuracy from an HMC sample and reconstruction accuracy from a variational sample. The intuition behind this metric is almost identical to that of likelihood regret. While HMC improves reconstruction of inliers, it improves reconstruction of outliers even more. This is because for an outlier, we expect a bigger gap between the variational posterior and the true posterior which HMC samples from.

Table 5.5 and 5.6 show AUROC values using this metric under different models. While the AUROC values suggest that using this metric is better than random guessing, the results are not good. FashionMNIST vs MNIST is the only task that shows reasonable performance.

Table 5.5 AUROC values using HMC Improvement (FashionMNIST vs others)

FashionMNIST vs	HMC-VAE (1-layer)	HMC-VAE (3-layer)
MNIST	0.863	**0.898**
Noise	**0.866**	0.674
Constant	0.536	**0.812**

Table 5.6 AUROC values using HMC Improvement (CIFAR10 vs others)

CIFAR10 vs	HMC-VAE (1-layer)	HMC-VAE (3-layer)
SVHN	0.659	**0.688**
Noise	**0.853**	0.648
Constant	0.535	**0.636**

The issue lies in HMC sampling as we observed that for outliers, HMC mostly only rejects or only accepts new samples which means the step size is either too big or too small for outliers. This makes sense because the step size is tuned for inliers but this also means HMC provides no reconstruction improvement for outliers which invalidates the assumption of this method. A way to fix this is to adaptively tune the step size for each sample and run the chain for longer, which is left for future work as doing it directly is prohibitively expensive.

5.3.3 Input Complexity

Input Complexity is a direct application of Serrà et al. (2020). The metric is as follows[3]:

$$\text{Input Complexity}(\mathbf{x}) = -\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - L(\mathbf{x}) \tag{5.2}$$

where $L(\mathbf{x})$ is the size (in bits) of $\mathbf{x}$ after PNG compression.

Table 5.7 and 5.8 show AUROC values using this metric under different models. The results are mostly as expected, this method works well on most tasks except when the outlier is Noise. As mentioned before, Noise outliers have disproportionally high complexity which make the metric unreliable. Notably, 3-layer VAEs are able to handle Noise significantly better than other models. After inspection, this is because 3-layer VAEs are the worst at reconstructing Noise and the complexity term is not high enough to skew the metric.

[3]Adapted to use reconstruction accuracy instead of data likelihood.

Table 5.7 AUROC values using Input Complexity (FashionMNIST vs others)

FashionMNIST vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
MNIST	0.934	**0.967**	0.870	0.901
Noise	0.244	**1.000**	0.175	0.689
Constant	**1.000**	**1.000**	**1.000**	**1.000**

Table 5.8 AUROC values using Input Complexity (CIFAR10 vs others)

CIFAR10 vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
SVHN	0.933	**0.935**	0.934	0.931
Noise	0.069	**0.704**	0.023	0.325
Constant	**1.000**	**1.000**	**1.000**	**1.000**

Another problem with this metric is the compatibility between likelihood function and the complexity term. It just so happens that the likelihood function used in this work and the complexity term are similar in magnitude. This method probably does not work as well if we chose to use other likelihood functions (e.g. Gaussian, Continuous Bernoulli). For future work, it would be useful to find complexity measures that are more coupled to the likelihood functions.

5.4 Methods Based on Ensemble Uncertainty

The last class of methods we will explore in this work are based on ensemble uncertainty. We will experiment with the metrics from Choi et al. (2019):

$$\mathrm{WAIC}(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\theta}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \mathrm{Var}_{\boldsymbol{\theta}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] \tag{5.3}$$

and from Daxberger & Hernández-Lobato (2019):

$$\mathrm{Agreement}(\mathbf{x}) = \frac{1}{\sum_{i=1}^{N} w_{\boldsymbol{\theta}_i}^2} \quad \text{where} \quad w_{\boldsymbol{\theta}_i} = \frac{p_{\boldsymbol{\theta}_i}(\mathbf{x}|\mathbf{z})}{\sum_{j=1}^{N} p_{\boldsymbol{\theta}_j}(\mathbf{x}|\mathbf{z})}\,. \tag{5.4}$$

We also throw in precision as a metric for comparison:

$$\mathrm{Precision}(\mathbf{x}) = \frac{1}{\mathrm{Var}_{\boldsymbol{\theta}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right]}\,. \tag{5.5}$$

For all 3 metrics, the lower the value, the more likely the input is an outlier. We apply the metrics on deep ensembles of 10 models (Lakshminarayanan et al., 2017).

Table 5.9 AUROC values using ensemble metrics (FashionMNIST vs others)

FashionMNIST vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
		Precision		
MNIST	0.554	**0.967**	0.737	0.841
Noise	0.776	**1.000**	0.645	0.999
Constant	0.993	**0.995**	0.993	0.993
		WAIC		
MNIST	0.459	**0.891**	0.539	0.712
Noise	0.939	**1.000**	0.919	0.999
Constant	0.993	**0.994**	0.993	0.993
		Agreement		
MNIST	0.637	**0.655**	0.637	0.607
Noise	0.710	0.755	0.518	**0.770**
Constant	0.761	0.743	0.748	**0.764**

Table 5.10 AUROC values using ensemble metrics (CIFAR10 vs others)

CIFAR10 vs	VAE (1-layer)	VAE (3-layer)	HMC-VAE (1-layer)	HMC-VAE (3-layer)
		Precision		
SVHN	0.729	0.664	**0.794**	0.720
Noise	0.843	0.987	0.961	**0.998**
Constant	0.976	**0.989**	0.979	0.985
		WAIC		
SVHN	0.688	0.593	**0.757**	0.651
Noise	0.867	0.988	0.965	**0.998**
Constant	0.975	**0.989**	0.978	0.984
		Agreement		
SVHN	0.530	0.520	**0.540**	0.527
Noise	0.563	0.564	0.406	**0.584**
Constant	0.572	0.564	0.570	**0.583**

Table 5.9 and 5.10 show the AUROC values for the benchmarks under these metrics. Precision and WAIC perform similarly well in most tasks under every model, while agreement performs a lot worse than expected. After inspection of the reconstruction accuracy of the ensembles, we noticed that the absolute difference between the highest and the lowest reconstruction accuracy ($\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \min_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$) are similar for inliers and outliers. Agreement is sensitive to the absolute difference in reconstruction accuracy because of the normalization step. So despite inlier reconstructions have smaller variance, the absolute difference push the Agreement metric to 1 (the minimum value of the metric).

5.5 Methods Comparison

This section serves to summarize the results in a visual format. We are comparing our results to (Likelihood Regret; Xiao et al., 2020). The AUROC values of Likelihood Regret are shown in Figure 5.11, which to our knowledge is the state-of-the-art for VAE outlier detection on this set of benchmarks.

Table 5.11 AUROC values using Likelihood Regret (Xiao et al., 2020)

FashionMNIST vs	VAE (1-layer)
MNIST	0.988
Noise	1.000
Constant	1.000

CIFAR10 vs	VAE (1-layer)
SVHN	0.875
Noise	0.994
Constant	0.974

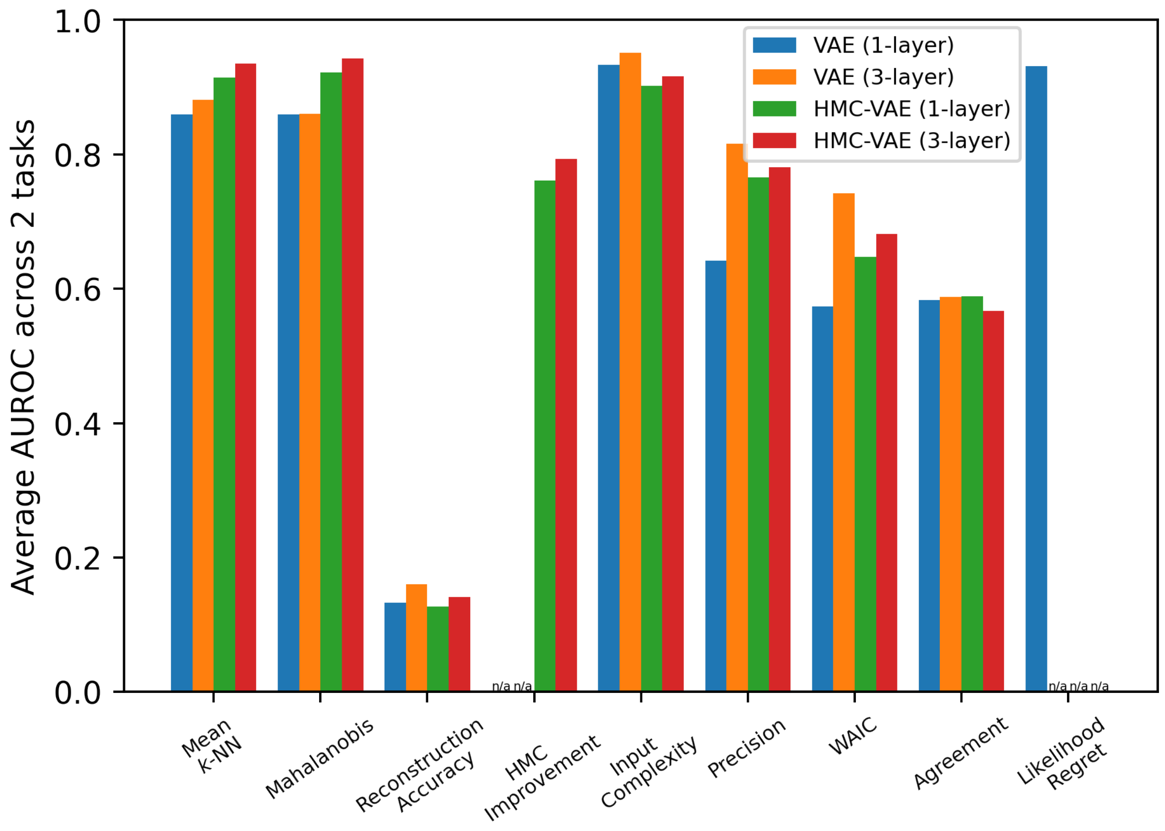

Fig. 5.1 Average AUROC values (averaged across FashionMNIST vs MNIST and CIFAR10 vs SVHN) against different method + model combination.

Throughout this chapter, we experimented with 8 outlier detection methods on 6 tasks under 4 different VAE models. We are primarily interested in knowing whether a

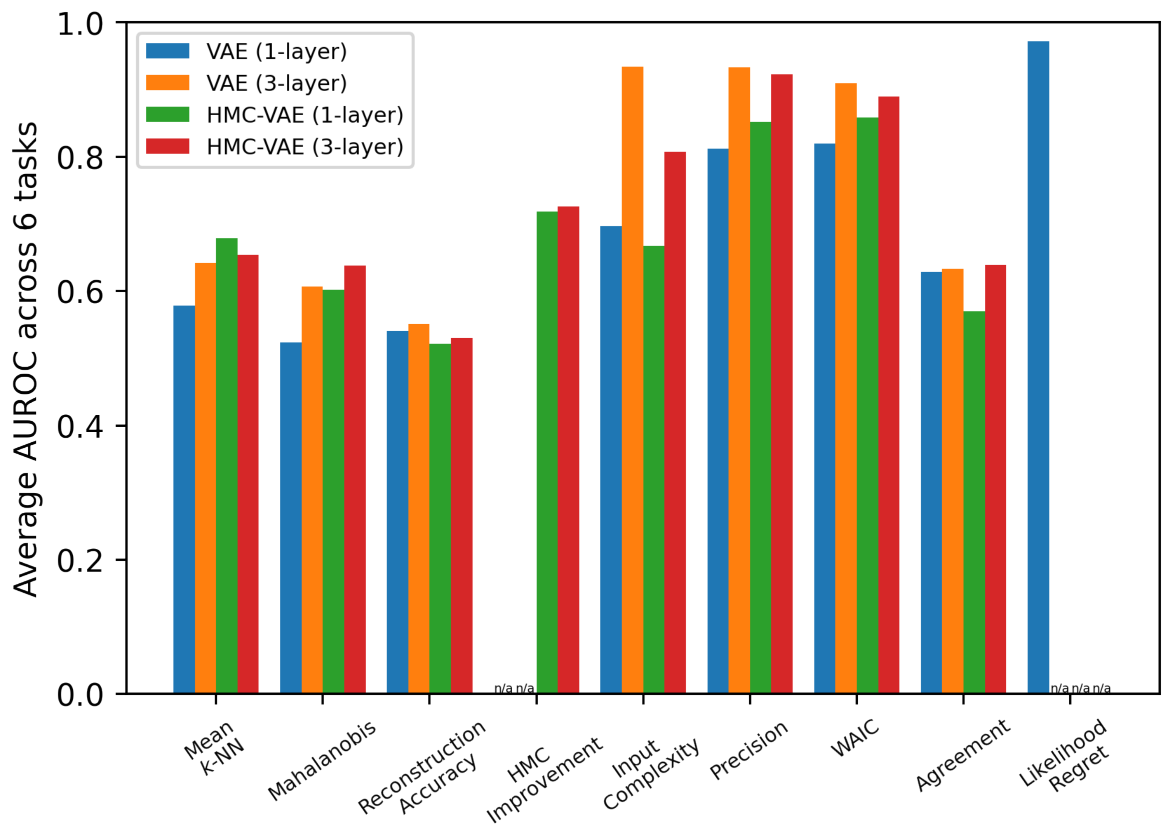

Fig. 5.2 Average AUROC values (averaged across FashionMNIST vs MNIST, Noise, Constant and CIFAR10 vs SVHN, Noise, Constant) against different method + model combination.

particular method or model is better. For each method + model combination, we average its AUROC values across tasks. In Figure 5.1, we average the AUROC values of 2 the two main tasks - FashionMNIST vs MNIST and CIFAR10 vs SVHN. And in Figure 5.2, we average the AUROC across all 6 tasks (FashionMNIST vs MNIST, Noise, Constant and CIFAR10 vs SVHN, Noise, Constant). We make the following observations:

- Averaged across the 2 main tasks, distance based methods already work very well and the results under 3-layer HMC-VAEs surpass Xiao et al. (2020). But for all 6 tasks, these methods clearly do not work as well as others.
- Input Complexity also works very well with the 2 main tasks. Though across all 6 tasks, only the 3-layer VAE worked well. In fact, its performance is very close to (Xiao et al., 2020). It is a shame that Input Complexity paired with other models does not perform nearly as well across all 6 tasks.

- For the ensemble based methods, Precision and WAIC both perform well and the performance are relatively consistent across models.
- The trend suggests that hierarchical latent structure does help outlier detection performance, despite it decreasing reconstruction performance. On the other hand, although HMC increases reconstruction performance, it does not seem to affect outlier detection much.

Regrettably, none of the method + model we have tried surpassed Likelihood Regret (Xiao et al., 2020) across all 6 tasks. Though one has to keep in mind that Likelihood Regret requires a significant amount of computation at prediction time, and methods like Input Complexity and WAIC only require a fraction of the computation while having performance close to Likelihood Regret.

Chapter 6

Conclusion and Future Work

The original objective was to determine whether we can improve outlier detection performance while retaining the gain offered by hierarchical latent variables and HMC. The objective was perhaps slightly misguided because the assumption was that VAE reconstruction accuracy is a good metric for detecting outliers. We showed in Chapter 4 that reconstruction accuracy is an unreliable metric for outlier detection, whether with or without hierarchical latent variables and HMC. The real objective has been to find methods that work well under all these VAE models.

An additional caveat was that, as shown in Chapter 3, hierarchical latent structure does not necessarily bring improvements to data reconstruction as previously shown. The improvement depends on the neural network architecture and the number of latent units used. Though we did confirm that hierarchical latent structures offer some disentangled/interpretable latent states and HMC does indeed improve reconstruction.

In the mini survey in Chapter 4, we listed some solutions from well-cited papers and subsequently applied them to our VAE models in 5. Our results suggest that some methods work much better on specific tasks than others. We would pick Input Complexity (Serrà et al., 2020) to be the best method overall (compared with the other methods used in this work) because it worked well across benchmarks, models, and required next to no additional computational resources, even though it did not achieve the best AUROC scores. We can also be fairly confident that hierarchical latent variables do help outlier detection, although the reason is unknown and left for future work. As for HMC, there does not seem to be a clear correlation between it and outlier detection performance. This means one can enjoy the reconstruction improvement HMC brings without worrying about it hurting outlier detection.

Most published work in VAE outlier detection theorized as to why the problem exists and proposed relevant solutions to patch it. While practically useful, these approaches are deeply unsatisfying because reconstruction accuracy (or data marginal likelihood) should be enough to detect outliers, contingent on the VAE actually being capable of learning the data distribution. While VAEs have proven to be very good at data reconstruction and data generation, they do not seem to be able to learn the true underlying data distribution. To achieve this, we likely need substantially more understanding of amortized variational inference and distributions parameterized by neural networks, we hope to investigate these in future work.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 10–21, Berlin, Germany, Aug 2016.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv:1509.00519 [cs, stat]*, Nov 2016.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

Hyunsun Choi, Eric Jang, and Alexander A. Alemi. Waic, but why? generative ensembles for robust anomaly detection. (arXiv:1810.01392), May 2019.

Erik Daxberger and José Miguel Hernández-Lobato. Bayesian variational autoencoders for unsupervised out-of-distribution detection. *NeurIPS Workshop on Bayesian Deep Learning*, 2019.

Taylor Denouden, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Buu Phan, and Sachin Vernekar. Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance. (arXiv:1812.02765), Dec 2018.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019.

Matthew D. Hoffman. Learning deep latent gaussian models with markov chain monte carlo. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1510–1519. Jul 2017.

Matthew D. Hoffman and Andrew Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. (arXiv:1111.4246), Nov 2011.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30. 2017.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Alexander LeNail. Nn-svg: Publication-ready neural network architecture schematics. *Journal of Open Source Software*, 4(33):747, 2019.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019a.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality. (arXiv:1906.02994), Oct 2019b.

Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2): 125–139, Apr 2001.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Ignacio Peis, Chao Ma, and José Miguel Hernández-Lobato. Missing data imputation and acquisition with deep hierarchical models and hamiltonian monte carlo. *arXiv:2202.04599 [cs, stat]*, Feb 2022.

Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 32. 2019.

Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F. Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In *International Conference on Learning Representations*, 2020.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, SørenKaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 29. 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

Zhisheng Xiao, Qing Yan, and Yali Amit. Likelihood regret: An out-of-distribution detection score for variational auto-encoder. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20685–20696. 2020.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv:2110.11334 [cs]*, Oct 2021.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from deep generative models. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 4091–4099. Jul 2017.