Investigating the viability of Generative Models for Novelty Detection

Undergraduate Thesis

Submitted in partial fulfillment of the requirements of
BITS F421T Thesis

By

Vidhi Jain
ID No. 2014A7TS0113P

Under the supervision of:

Dr. Aaron Courville
&
Dr. Hari K.Babu

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS
May 2018

Declaration of Authorship

I, Vidhi Jain, declare that this Undergraduate Thesis titled, 'Investigating the viability of Generative Models for Novelty Detection' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Certificate

This is to certify that the thesis entitled, "*Investigating the viability of Generative Models for Novelty Detection*" and submitted by Vidhi Jain ID No. 2014A7TS0113P in partial fulfillment of the requirements of BITS F421T Thesis embodies the work done by her under my supervision.

Supervisor
Dr. Aaron Courville
Faculty,
Montreal Institute of Learning Algorithms (MILA) lab
Date:

Co-Supervisor
Dr. Hari K.Babu
Asst. Professor,
BITS-Pilani Pilani Campus
Date:

"Humans learn abstract concepts and generalize, yet know when something novel or anomalous is encountered. Can we make our machines respond in the same way?"

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS

Abstract

Bachelor of Engineering (Hons.) in Computer Science

Investigating the viability of Generative Models for Novelty Detection

by Vidhi Jain

Artificial Intelligence (AI) is a field that aims to enable machines with the skills and knowledge to solve the tasks that have traditionally required human mental labour. An emerging field under AI is Deep Learning, in which artificial neural networks of significantly greater depth are used to learn the representations and their mapping to the desired output [9]. In Deep Learning, generative modelling is a search for the best set of parameters θ to model $p_\theta(x)$ that can estimate the true data distribution $p_{true}(x)$, such that we may sample from the model's estimate and obtain data that have combinations of different factors of variation as represented in the training data.

One of the critical in the real-world settings is Novelty or Out-of-distribution (OOD) detection, where the closed world assumption of a trained model does not hold. We present a study of the viable signals in generative models to indicate OOD detection at test time. Explicit density models like PixelCNN, Variational Autoencoders (VAE), provide the density function estimate to compute likelihood per instance. We experiment with some hypotheses like log likelihood, prediction gain and gradient norm, for OOD signals in image datasets and evaluate Area under Precision-Recall (AUPR) for in-/out- distribution data. While MNIST/Omniglot experiment shows promising results with signals, the natural images and leave-one-class-out data have quite close local image statistics in the in-/out- distributions. We highlight the challenges in detecting semantic level differences using these signals in the tasks.

Acknowledgements

I would like to offer my special thanks to Prof. Dr. Aaron Courville for giving me the opportunity and valuable guidance to pursue my bachelor thesis research. I express my gratitude for his motivation, patience and enthusiasm, that have made this research study an enriching learning experience. I would also like to thank my co-supervisor, Prof. Dr. Hari K. Babu for his timely support and consideration. My sincere thanks goes to Faruk Ahmed, who has been a collaborator in this work. I would like to thank Montreal Institute for Learning Algorithms (MILA) at University of Montreal for hosting my research thesis as an internship. Thanks to my home university, Birla Institute of Technology and Science Pilani, for the provision to pursue my research off-campus. A big thanks goes to my parents for believing in my abilities and supporting my interests.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is novelty detection?

Novelty detection refers to the identification or detection of data that lies in a distribution that has not been used in training our model. It aims to determine if the observation at test time is within the known data distribution(i.e., inliers or in-distribution) or outside of it (i.e., outliers or out-distribution)[24].

The signals for novelty detection have been studied broadly in two major contexts. First, these signals may allow a classifier to say "I don't know" rather than giving a wrong prediction among the labels that it was trained upon. Second, this detection signal is also useful in reinforcement learning to make the agent explore or avoid a novel encounter.

1.2 What are different approaches to novelty detection?

The most common way to view the problem of novelty detection is to consider it as a one class classification problem where the entire training data is taken as a single class[22]. There have been attempts to convert it to a binary classification problem by using some representation of the anomalous data or to a $k+1$ classifier where k is the number of classes and 1 is to classify the unknown [16]. This approach can protect a model from certain kinds of anomaly but cannot cover all possible anomalous data.

The feature representations learned in the classifier are very generic and highly invariant based upon the class label. Classifier based approaches also depend heavily upon the labelled data. To make use of the abundant unlabelled data, we need to consider unsupervised models and their respective signals for novelty detection. The unsupervised approach can focus on learning the

data manifold very well while training. If any sample at the test time appears to not belong to this learned distribution, this is a signal for an anomaly.

Generative modelling is a type of unsupervised learning as it is trained to generate meaningful samples from the underlying distribution of the given data, without explicit labels. It is expected to learn some higher level representation in the hidden latent codes of this data. In the following section, we discuss generative models and the reasons why such models may be useful for novelty detection.

1.3 What are generative models?

A generative model is any model that learns to estimate the distribution of the training data as a probability density function, which can be used to generate samples from this distribution.

Given a set of training data belonging to a distribution p_{data}, a generative model p_{model} learns to estimate that distribution.

Using the generative model, we can learn a simulator of data and extract meaningful concepts out of raw data [19].

The generative models can be broadly classified based on explicit or implicit density estimation. The explicit density models can give an estimate of p_{model} for a particular value of data x. The explicit density can estimated in a tractable way, like in fully observed models such as PixelCNN, or in an approximate manner, like in latent variable models such as Variational Autoencoders (VAE). The implicit density estimate allows one to draw only samples from the distribution p_{model}. Generative Adversarial Networks (GAN) are an example of the implicit density generative models.

1.4 Novelty detection and its family of synonyms

Novelty detection is often used interchangeably with Anomaly detection, Out-of-Distribution detection and Outlier detection because several common techniques are used to solve these challenges. We observed some subtle differences in the context of each term.

The term ‘outliers’ includes a fraction of valid data that causes inconsistency in building a robust and generalizable model of normality. The issue of outliers is often discussed in the model building phase.

At model evaluation phase, we use the term ‘anomalies’ to refer to the data that do not conform to the patterns observed in training dataset. The anomalies tend to violate the well-defined

notion of normal behavior as per our estimated data model. These can be due to an encounter with either novel or useless artefacts [2]. The term novelty can be used to describe the 'interesting' anomalies, where interestingness depends on the context.

All these techniques are also related to noise removal techniques, where the corrupted data is to be removed. Noise removal is an important step before training or model building phase, whereas the 'noise' encountered during analysis or model detection phase is mostly dealt under anomaly detection.

In this work, we focus on out-of-distribution detection, which is synonymous with anomaly. For the current investigation, every anomaly is considered novel and therefore, out-of-distribution, anomaly detection and novelty detection refer to same context.

Chapter 2

Explicit Density Generative Models

An important use of generative modelling is to understand the unobservable underlying probability density function of the set of observed data points sampled from a large population. Explicit density estimation models make use of Maximum Likelihood Estimation (MLE) methods.

2.1 Fully observed Models

Fully observed models are designed to focus on the observed data directly without having any unobserved latent representations. These models can be applied to data in sequence and therefore also known as Autoregressive models. Autoregressive model can be expressed as a causal graph where the next value in the sequence is regressed over the previous values in the sequence.

Among the several fully observed models, there is a spectrum ranging from directed to undirected and from discrete and continuous models [19].

A tractable estimate of density provides the likelihood of the data, that can be directly optimized. These functions can be modelled as a joint distribution over a set of random variables $\mathbf{x} = x_1, x_2, \ldots, x_n$. This means that the likelihood of the data sample x is written as a product of 1-D conditional distributions using the chain rule of probability [28].

For example, in order to estimate the likelihood of an image x as $p_\theta(x)$, we calculate the product of the conditional probability of each pixel conditioned on all the previously seen pixels[26].

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i \mid x_{<i})$$

In order to train the model, we maximize this likelihood on our training data. The following section describes how a complex function for modelling the distribution can be done by using

neural networks. In our study, we focus on the directed and discrete set of tractable generative models, which include PixelCNN, fully visible belief networks and other auto-regressive techniques like NADE and MADE.

2.1.1 PixelCNN

PixelCNN and PixelRNN [20] capture the pixel inter-dependencies without any assumptions such as latent variable models. PixelRNN models the distribution using LSTM units or bi-LSTM units whereas PixelCNN applies convolutions over a context region.

PixelRNN model achieves slightly better performance than PixelCNN. But PixelCNN is computationally efficient to train as compared to PixelRNN because it allows parallel computation of all the conditional probabilities.

At generation time, both PixelCNN and PixelRNN perform sequential generation, which can be slow for high resolution data.

Ordering of pixels in a sequence

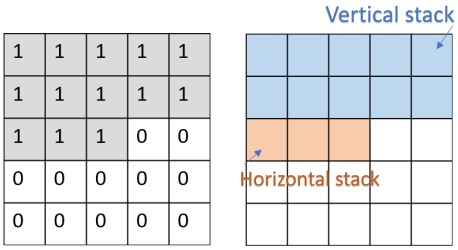
Since there is no obvious order of pixel values in images, these models define a conditioning of the pixels ordered from top left to bottom right.

Several different orderings have been explored in NADE and MADE, and related ideas about exploring structure using splitting and masking can also be seen in NICE and Real NVP. The direct modelling of conditional probabilities per pixel is implemented in NADE. However, PixelCNN makes use of the topological structure to grow the dependency chain over depth [12].

Masking

To preserve the causal structure in the image data, the masked convolutions are used. The proposed masking in [20] suggested to zero out the weights in the kernel to ensure that the information of the subsequent pixels in the sequence does not reach the previous ones. The arbitrary ordering for applying autoregressive models had to be imposed in spatial dimensions, as well as in color channels too. This masking technique had a blind spot problem [21], that is the pixels towards the right of the current pixel were not being taken into account. This was ignoring more than a quarter of the potential receptive field in the case 3x3 filters. This was solved by dividing the mask into vertical and horizontal stacks (2.1).

FIGURE 2.1: Left image shows 5x5 kernel with fixed weight masking that results in blind spots; Right image shows 5x5 kernel which was divided into veritical and horizontal stacks (Inspired from [21])



Multiplicative Units / Gated Activation Units

This paper also introduced Gated activation units, which involved a combination of sigmoid and tanh activations instead of just a ReLU activation. This was done to bring about the complex multiplicative interactions found in PixelRNN (due to LSTM gates) to the PixelCNN. It is represented as $\mathbf{y} = tanh(W_{k,f} * \mathbf{x}) \odot sigmoid(W_{k,g} * \mathbf{x})$ where $\mathbf{x}$ is the input tensor, $\mathbf{y}$ is the output tensor, k is the the number of the layer, f indicates the weights on which tanh activation is applied, and g indicates the weights on which sigmoid activation is applied.

Residual and Skip connections

The residual connections were introduced for the horizontal stack. Moreover, skip connections were added to incorporate features from all layers at the very end of the network.

Conditional PixelCNN

Conditional PixelCNN was introduced to incorporate a high level description $\mathbf{h}$ in the model. The conditional probability can be calculated by $p(\mathbf{x}|\mathbf{h}) = \prod_i^{n^2} p(\mathbf{x}_i|\mathbf{x}_{<i}, \mathbf{h})$ where $\mathbf{x}$ is the pixel under consideration, $\mathbf{x}_{<i}$ are all the pixels previous to the current one as per the assumed ordering and n^2 highlights the total number of the pixels in an image of height and width n.

The conditional information $\mathbf{h}$ is included in the gated unit by adding the terms as a bias to the output of the convolution as $\mathbf{y} = tanh(W_{k,f} * x + V_{k,f}^T \mathbf{h}) \odot \sigma(W_{k,g} * x + V_{k,g}^T \mathbf{h})$ where V represents the trainable parameter in the conditioned by the vector $\mathbf{h}$. The conditioning can also be done in location dependent way if that information is available. By extracting a spatial representation s from the vector $\mathbf{h}$ through a deterministic mapping, s can be convolved with the parameter V instead of just matrix multiplication with the vector h.

To improve on some issues in performance of the PixelCNN, we discuss a variant, namely, PixelCNN++.

2.1.2 PixelCNN++

PixelCNN++ [26] contains several modifications over PixelCNN models. The most significant among all is *discretized logistic mixture likelihood*. If a conditional distribution of a color-channel by a softmax loss function over range of 0 to 255 is taken, it is costly to train in terms of memory and data. This makes the gradients of the network parameters very sparse early in training as it learns to assign zero probability to a sub-pixel value which is never observed.

The discretized logistic loss is implemented by assuming a latent color intensity ν with continuous univariate distribution, which is a mixture of K logistic distributions. To obtain a probability for observed x given the π_i as the weights, μ_i as the mean and s_i as standard deviation of each component distribution of the mixture, we find

$$P(x|\pi,\mu,s) = \sum_{i=1}^{K} \pi_i \left[\sigma\left(\frac{(x+0.5)-\mu_i}{s_i}\right) - \sigma\left(\frac{(x-0.5)-\mu_i}{s_i}\right)\right] \tag{2.1}$$

(here K is a hyper-parameter, often chosen between 5 to 10) This equation does not hold for values of 0 and 255. These edge cases are observed (in CIFAR-10 dataset) to have higher probability than their neighbouring values[26]. Therefore, the probability for these two edge cases is just assigned as 1 for each component of mixture to ensure the probability mass remains in the valid range.

Other improvements introduced in PixelCNN++ include conditioning on pixels without channel ordering, and a few architecture increments like having short-cut connections, dropout and favoring downsampling as compared to the dilated convolutions.

2.2 Approximate Density Models

Latent Variable Models have a vast literature of models ranging from discrete to continuous, parametric to non-parametric and direct/linear to deep (in terms of architecture). These models enable compression and disentanglement of the information terms in the latent vectors. This work focuses on the deep continuous parametric models, particularly Variational Autoencoders (VAE).

To build such approximate density estimate, we assume a latent random variable $\mathbf{z}$ to explain every observed variable $\mathbf{x}$. This causal relationship can be used in estimating the observed likelihood $\log p_\theta(\mathbf{x})$ in terms of the learned latent dependence because the marginal probability of x is often intractable. A probabilistic view of the latent space z also allows us to sample from z and generate new data samples, in contrast to autoencoders (more about autoencoders in Appendix-C).

2.2.1 Variational Autoencoder (VAE)

Variational Autoencoder resulted from the combination of the graphical model theory with neural networks [14].

To maximize the log-likelihood of the true distribution as $\log p(x)$, the likelihood is expressed as a marginal of the joint distribution of the latent variable z and x. $\log p(x) = \log \int_z p(x, z)dz$ We can introduce a tractable distribution q over the latent variable z given x. A lower bound is estimated using the Jensen's inequality. This is known as the Evidence Lower Bound (ELBO) (derivation in Appendix-A).

The gap that exists between the true likelihood term and the ELBO term is called the approximation gap. This approximation gap between the true likelihood and the ELBO is a KL divergence term between the prior distribution of the latent code and the posterior distribution of the latent code given x. In order to maximize the log-likelihood of the data, we need to minimize the approximation gap, that implies, to maximize the ELBO.

In the case of amortized variational inference [6] [13], this written in the terms of the parametric inference model $q_\phi(z|x_i)$ for each batch of x_i as :

$$\mathcal{L}^{(i)}(\theta, \phi) = \mathop{\mathbb{E}}_{z \sim q_\phi(z|x_i)} \log p_\theta(x_i|z) - \mathcal{D}_{KL}(q_\phi(z|x_i) \parallel p_\theta(z))$$

The first term acts as the expectation of the negative reconstruction error and the second term is the Kullback-Leibler divergence ($\mathcal{D}_{KL}$) between the two distributions, also seen as a regularizer term. So, the VAE can be called a regularized autoencoder. It has $q(z|x)$ modelled by the encoder network and $p(x|z)$ modelled by the decoder network.

In the simplistic case, z is assumed to be as a normal distribution that has mean μ and diagonal co-variance σ. At the end of the encoder, we train the network to generate the mean and variance of the distributions. This is used to sample z using a reparameterization trick [13]. This allows one to sample $\epsilon \sim \mathcal{N}(\epsilon; 0, I)$ and compute $z = \mu + \sigma \odot \epsilon$.

z is then fed into the decoder to learn to reconstruct the image based on the mean and variance of the distributions. Thus, we may sample from these distributions and fit it to the decoder network to obtain new samples.

The assumption of a Gaussian distribution prior enables calculation of the $\mathcal{D}_{KL}$ in closed form as:

$$D_{KL}(\mathcal{N}(\mu(X), \sigma(X)) \parallel \mathcal{N}(0, 1)) = \frac{1}{2} \sum_k \big(\exp(\sigma(X)) + \mu^2(X) - 1 - \sigma(X)\big)$$

The reconstruction loss is computed according to the distribution of $p_\theta(x|z)$. In the case of multivariate Bernoulli, the cross entropy loss is taken between the true samples and the reconstructed output (which is normalized between 0 to 1 by sigmoid activation). For a continuous distribution, the probability density function is used to compute the expectation of log-likelihood of the encoder network.

2.3 Further developments

The conventional class of mean-field posterior approximations has a limited to ability to resemble true data distribution. Richer approximate posterior distributions have been experimented with. The under-estimation of the posterior variance can lead to poor results [3]. The over-estimation of variance in latent features has also been discussed as "exploding" latent space problem in VAE [29]. The limited capacity of posterior approximation may result in biases in maximum-a-posteriori(MAP) estimates [3].

The different families of structured posterior approximations include covariance models, mixture models, copula models, and normalizing flows. In the following subsections, some of the further developments are briefly discussed.

2.3.1 Normalizing Flows

The approximate posterior distribution starts with a simple initial density. The normalizing flow describes a sequence of invertible transformations applied to probability density to bring about the desired level of complexity in the distribution [23][19].

Let $f_\theta : \mathbb{R}^d \to \mathbb{R}^d$ be a smooth invertible mapping such that $g = f^{-1}$. This implies that for a random variable z belonging to a distribution $q(z)$, $g \circ f(z) = z$. The density of the random variable after transformation by f is

$$q(z_1) = q(z_0) \left| \frac{\partial f^{-1}}{\partial z_1} \right| = q(z_0) \left| \frac{\partial f}{\partial z_0} \right|^{-1}$$

Let z_K be the random variable after K invertible transformations, that is, $z_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(z_0)$. The probability density function of the distribution $q_K(z_K$ can be computed as:

$$\log q_K(z_K) = \log q_0(z_0) - \sum_{k=0}^{K} \ln det \left| \frac{\partial f_k}{\partial z_{k-1}} \right|$$

The planar and radial normalizing flows have been discussed in [23].

Inverse Autoregressive Flows

The Inverse Autoregressive Flows are a kind of normalizing flow where each of the invertible transformations in the chain are based on autoregressive neural networks, in order to scale to more complex posterior densities [15].

2.3.2 PixelVAE

PixelVAE has incorporated a PixelCNN model as the decoder to a Variational Autoencoder [10]. The complementary strengths of VAE to capture global coherence structure and that of a PixelCNN to model the local details, has been utilized together. The PixelVAE is also extended to multiple stochastic layers.

2.3.3 Variational Lossy Autoencoder

By explaining the lack of information coding in z using Bits Back coding, the work shows that extra coding cost for the KL divergence term exists and can't be reduced by improving posterior approximations. The approach to learn a lossy representation of data motivates the generator to not capture the information we want to lose [3].

Chapter 3

Experiments

3.1 Exploring Anomaly Signals in Generative models

We carried out experiments to detect anomalous or out-of-distribution samples with respect to a trained generative model. We discuss each of our hypothesis and the observations of some of the experiments.

3.1.1 Shannon's Information or (Negative) Log Likelihood

Given a trained model M on data $x \in D$, generative models like the fully observable models and prescribed latent code models can provide explicit estimate of the density.

In PixelCNN, we can measure this using likelihood estimate $\rho(x)$ which is described as the product of the conditional probabilities of each pixel given the previously seen pixels. In VAE, the variational lower bound gives us an estimate of the likelihood of the data given the parameters θ of the decoder network p and ϕ of the encoder network q.

3.1.2 Prediction Gain

Information Gain and Prediction Gain have been discussed for generative models in exploration tasks in Reinforcement Learning [1]. We calculate the prediction gain by evaluating the log ratio of the density estimate after the model makes an update step after seeing x, $\rho^{'}(x)$ to the density estimate without having seen x, $\rho(x)$. This density estimate is obtained by analyzing the cost.

This intuitively gives us an estimate of how unlikely the data is under the current model subtracted from the likelihood when we just update the model based on this data.

3.1.3 Gradient Norm

Instead of taking an update step, we can also look into how do the gradients change for an in-distribution or out-distribution sample. Intuitively, we expect a small change in the gradients for an in-distribution sample and a large change for an out-distribution sample. It can be expected that depending upon the sharing of lower level statistics between in and out distribution, different layers may reflect this gradient change differently, especially more significant differences in the abstract layers of the model.

We checked whether a model trained on a data distribution $p_{data}(x)$ will produce lower L2 norm of the gradients on any $x \in X$ as compared to some sample from unseen data distribution.

We also observed the cumulative layer-wise gradient norm, starting from the last layer. This is to observe which layers significantly contribute to the difference in the gradient norm of the samples from two different distributions.

3.2 Experiments

To understand how well we can signal out-of-distribution sample at test time, we chose two approaches to create in and out of distribution datasets. First, we consider datasets pairs that have comparable distributions yet are semantically distinct, for example, MNIST and Omniglot, CIFAR 10 vs CIFAR 100. Second, we hold one class out while training the model and treat the left out class samples as out of distribution. Both problem settings are designed to test the OOD signals such that we detect the distributional changes based upon semantic level differences in the data and not mere image statistics.

This is a harder task than taking in and out distributions to be two distinct data distributions defined on the image space as in [18]

We evaluate the Area under the Precision Recall curve (AUPR) as used in [11]. The AUPR score compares the precision of the positive class with the true positive rate (TPR) of the examples. This is slightly different from the area under the curve for receiver operator characteristics (AUC-ROC) because it is a plot between false positive rate (FPR) and true positive rate (TPR). AUPR scores are considered more reliable than AUC-ROC in case of skew class labels [5].

We are interested in observing the AUPR where the out-of-distribution data is considered as the positive class. The in-distribution is given true label of 0 and the out-distribution is given true label of 1. If the value of AUPR is high, it means that there are high number of true positives among those predicted as positives.

The higher the AUPR, the better it is. The best value for AUPR to achieve is 100.

Model(In/Out)	LL	PG	GN	[Base]
PixelCNN(MNIST/Omni)	56.10	80.48	97.21	[56.86]
PixelCNN(CIFAR10/100)	53.26	53.56	47.63	[50]
PixelVAE(CIFAR-10/100)	54.09	48.79	55.25	[50]

TABLE 3.1: Table showing the AUPR of out-distribution comparison for Log Likelihood (LL), Prediction Gain (PG) and Gradient Norm (GN) of MNIST/Omniglot and CIFAR-10/100 datasets

PixelCNN on Fashion MNIST				
Class held-out	LL	PG	GN	[Base]
0 T-shirt/top	12.219	8.47	23.35	10
1 Trouser	5.774	10.54	7.83	10
2 Pullover	17.377	7.15	25.03	10
3 Dress	7.402	10.74	9.36	10
4 Coat	17.582	11.53	10.65	10
5 Sandal	6.639	18.235	7.74	10
6 Shirt	25.162	10.11	17.82	10
7 Sneaker	6.241	12.61	6.25	10
8 Bag	20.637	14.22	19.3	10
9 Ankle boot	9.904	17.45	7.53	10
MEAN	12.8937	12.1055	13.486	10

TABLE 3.2: Table showing the AUPR OUT for hold-one-class-out comparison for Log Likelihood (LL), Prediction Gain (PG) and Gradient Norm (GN) for PixelCNN on Fashion MNIST

PixelCNN on Fashion MNIST				
Class held-out	LL	PG	GN	[Base]
9 (Ankle Boot)	88.95/ 87.25	60.72/ 63.87	32.94/ 41.67	[90/10]
5,7,9 (All Footwear)	76.57/ 79.52	94.97/ 87.18	46.66/ 63.63	[50/50]
2 (Pullover)	99.9988/99.9000	98.8293/50.0	74.4207/ 10.0	[90/10]
2,6 (All Full Sleeves)	99.9975/ 99.9500	97.7547/69.9545	59.7690/ 20.0	[80/20]

TABLE 3.3: Table showing the AUPR (In-distribution/Out-distribution) for hold-one-or-many-class-out comparison for Log Likelihood (LL), Prediction Gain (PG) and Gradient Norm (GN) for PixelCNN on Fashion MNIST

In 3.1, the base refers to the percentage of the out-distribution samples among all the samples shown at the test time. The base rate indicates the AUPR that we should get by just random guessing between in- or out- distributions.

3.3 Conclusion

We experimented with different signals like log likelihood, prediction gain, and gradient norm for out-of-distribution detection in generative models. MNIST/Omniglot experiment shows some promising results with the signals. On the other hand for natural images like CIFAR10/100, there does not seem to be any significant improvement than random prediction baseline for the tasks of separating the in-/out-distribution that differ by semantic level differences and not local

CIFAR-100 (20 coarse) labels	CIFAR-10 labels
aquatic mammals	
fish	
flowers	
food containers	
fruit and vegetables	
household electrical devices	
household furniture	
insects (bee, butterfly, cockroach)	bird
large carnivores (leopard, lion, tiger, wolf)	cat, deer, dog
large man-made outdoor things	
large natural outdoor scenes	
large omnivores and herbivores (camel, cattle)	horse, dog, cat, deer
medium-sized mammals (fox)	cat
non-insect invertebrates (crab, lobster)	frog
people	
reptiles (crocodile, dinosaur, lizard, snake, turtle)	frog
small mammals	
trees	
vehicles 1 (pickup truck)	truck
vehicles 2 (rocket, tank, streetcar, tractor)	airplane, automobile, truck, ship

TABLE 3.4: Table showing the 20 coarse labels of CIFAR-100 and the closest possible CIFAR-10 label, if any

image statistics. Similar poor strength of the signals was observed when leave one class out experiments were carried out for Fashion MNIST dataset 3.2 and 3.3.

The signals of log likelihood, prediction gain and gradient norm do not achieve any significant performance benefit than the base in the task of detecting CIFAR-10 as in-distribution and CIFAR-100 as out-distribution (CIFAR10/100) 3.1. CIFAR10/100 is hard because the categories and the images are very similar in the two settings. In 3.4, it is clear that almost all the classes in the CIFAR-10 are also present in the CIFAR-100 dataset, which makes the task of out-of-distribution detection quite ill-defined, in both semantic differences and image statistics.

These experiments are not complete and not yet conclusive because the density estimation in the case of natural images is still a challenge even with the state-of-the-art models (refer B for more details). The models used in this analysis had not been optimized by hyper-parameter search because we expected that the optimization signal on a data from known distribution should be different from a data from unknown distribution. Contrary to this, better density estimation may contribute heavily to better signal strength and this needs to be further analyzed. Also, other possible signals can be explored which are more robust and strong in detecting the change in data distributions.

Appendix A

Mathematical Concepts

The concepts for understanding variational inference have been discussed. These concepts are largely summarized from [7][8][9].

A.1 Information

To quantify information in terms of probability, we can make the assumption that the events which are less probable carry more information than events which are more probable. So, we define information as

$$I = -\log p(x)$$

where $p(x)$ is probability of an event x.

A.2 Entropy

The average of information for an event with probability $p(x)$ is defined as Entropy. Entropy is the expectation of the information, which is defined as,

$$H = -\sum p(x) \log p(x)$$

A.3 Kullback-Leibler (KL) Divergence

To understand how much information is lost when we choose an approximation for a data distribution, we compare two probability distributions. One popular way is to measure relative

entropy by Kullback-Leibler (KL) Divergence of the probability distribution q with respect to p[1].

$$\mathcal{D}_{KL}(p||q) = -\sum p(x)\log q(x) + \sum p(x)\log p(x) = -\sum p(x)\log\frac{q(x)}{p(x)}$$

A.4 Maximum mean discrepancy (MMD)

Maximum mean discrepancy (MMD) is used to compare two distributions p and q given training data $\{x\}_{i=1}^{n} \sim p$ and $\{y\}_{i=1}^{m} \sim q$. We assume that there exists some transformation ϕ that is applied to the training data. The mean in the transformed space $\mu_{\phi(x)} = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)$ and $\mu_{\phi(y)} = \frac{1}{m}\sum_{j=1}^{m}\phi(y_j)$. Then the distance between the two distributions $|p-q|^2 = |\mu_{\phi(x)} - \mu_{\phi(y)}|^2$. We do not need to compute the ϕ as it can be written in terms of some kernel like Radial Basis Function kernel (RBF).

$$\begin{aligned}
&|\mu_{\phi(x)} - \mu_{\phi(y)}|^2 \\
&= |\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(y_j)|^2 \\
&= |\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(y_j)|^T|\frac{1}{n}\sum_{i=1}^{n}\phi(x_i) - \frac{1}{m}\sum_{j=1}^{m}\phi(y_j)| \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}m\phi(x_i)^T\phi(x_j) + \frac{1}{m^2}\sum_{i=1}^{n}\sum_{j=1}m\phi(y_i)^T\phi(y_j) - \frac{2}{mn}\sum_{i=1}^{n}\sum_{j=1}m\phi(x_i)^T\phi(y_j) \\
&= K(x_i, x_j) + K(y_i, y_j) + K(x_i, y_j)
\end{aligned} \tag{A.1}$$

The kernel must be chosen carefully such that it matches the structure of the data. In practice, one needs to try different kernels until it works.

A.5 Evidence Lower Bound

The loglikelihood of the data x is given by $\log p(x)$.

This term can be expressed as the marginal probability in terms of z as:

$$\log p(x) = \log \int p(x|z)p(z)dz$$

We can introduce another distribution q for the latent random variable z. By multiplying and diving by the $q(z)$ in the previous equation, we get:

[1] KL Divergence is non-negative ($KL \geq 0$) and it is not symmetric ($KL(p||q) \neq KL(q||p)$).

$$\begin{aligned}\log p(x) &= \log \int_z \frac{p(x|z)p(z)}{q(z)} q(z)dz \\ &= \log \int_z (\frac{p(x,z)}{q(z)}) q(z)dz \\ &= \log \mathop{\mathbb{E}}_z \frac{p(x,z)}{q(z)}\end{aligned} \tag{A.2}$$

As log is a concave function, log of average of a set of positive real numbers will be greater than the average of log values of the same set of positive real numbers (by Jensen's inequality).

$$\log \mathop{\mathbb{E}}_z \frac{p(x,z)}{q(z)} \geq \mathop{\mathbb{E}}_z \log \frac{p(x,z)}{q(z)}$$

The latter expression is known as the variational lower bound or ELBO.

$$\log p(x) - \mathop{\mathbb{E}}_z \log \frac{p(x,z)}{q(z)} = \mathop{\mathbb{E}}_z \log \frac{q(z)}{p(z|x)} = KL(q(z) \parallel p(z|x))$$

Appendix B

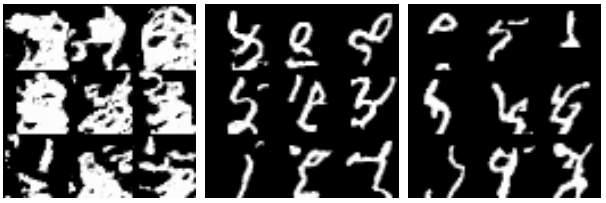
Experimental Results

B.1 PixelCNN

B.1.1 MNIST

MNIST is a dataset of handwritten digits [17]. In order to understand the need for vertical and horizontal stacking in PixelCNN, we implemented the vanilla PixelCNN which has blind spots[1] in Figure B.1.

FIGURE B.1: Samples after epoch 0, 8 and 21 during training Vanilla PixelCNN with blind spots on MNIST dataset



A PixelCNN with vertical and horizontal stacks to prevent blind spots is implemented. Training this model on MNIST digits yields samples as shown in Figure B.2

The train and test negative likelihood curves in Figure B.3 indicate that the model has not reached overfitting.

To see if we can improve further from here, the learning rate was reduced from $1e-3$ to $1e-6$ and the training was done another 20000 iterations in Figure B.4.

The visual quality of the images has not improved much and further tuning of model parameters is required.

[1] The vanilla PixelCNN model is very small and therefore, it is run till only 25 epochs to observe the performance.

FIGURE B.2: Samples after iterations 9999, and 19999 during training PixelCNN on MNIST dataset

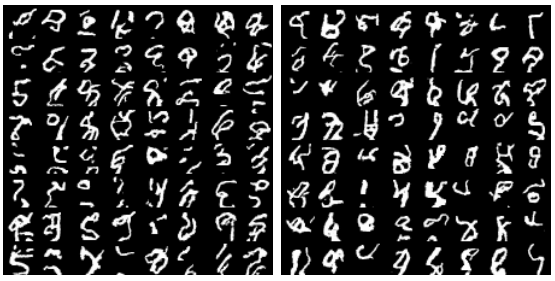


FIGURE B.3: Train and test curves during training PixelCNN on MNIST dataset till 19999 iteration

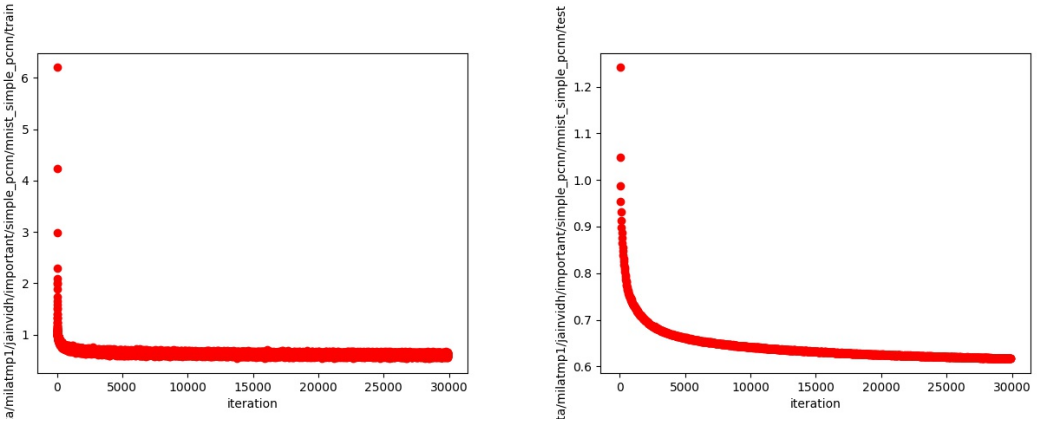


FIGURE B.4: Samples after iterations 29999, 39999 and 49999 during training PixelCNN on MNIST dataset

B.1.2 Fashion MNIST

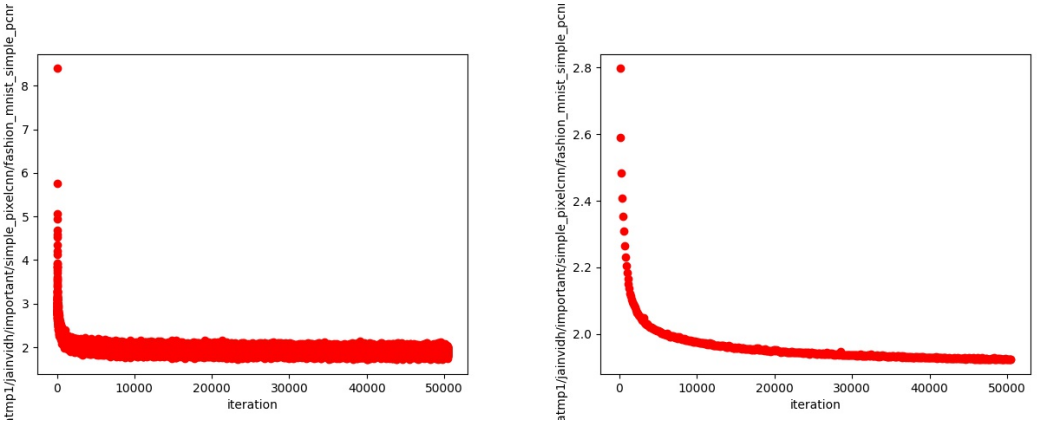
Fashion MNIST dataset [27] is similar to MNIST dataset but contains 10 classes of clothing objects instead. We trained the same PixelCNN model as used in subsection B.1.1) and obtained the samples in Figure B.5.

The train and test negative log likelihoods in Figure B.6 were observed to ensure that there is no over-fitting.

These results indicate that the current setting of our PixelCNN model can learn this data quite well. We observe that the same PixelCNN model architecture seems to learn continuous

FIGURE B.5: Samples after iterations 99, 14999 and 49999 during training PixelCNN on Fashion MNIST dataset

FIGURE B.6: Train and Test Negative Log Likelihoods for PixelCNN trained on Fashion MNIST



structures of Fashion MNIST dataset better than the sparse features in MNIST.

We test our hypothesis by carrying hold-some-class(es)-out experiments to evaluate the out-of-distribution signal strength. The samples after training Fashion MNIST dataset without the footwear classes, that is, without classes 5,7, and 9 looks like Figure B.8

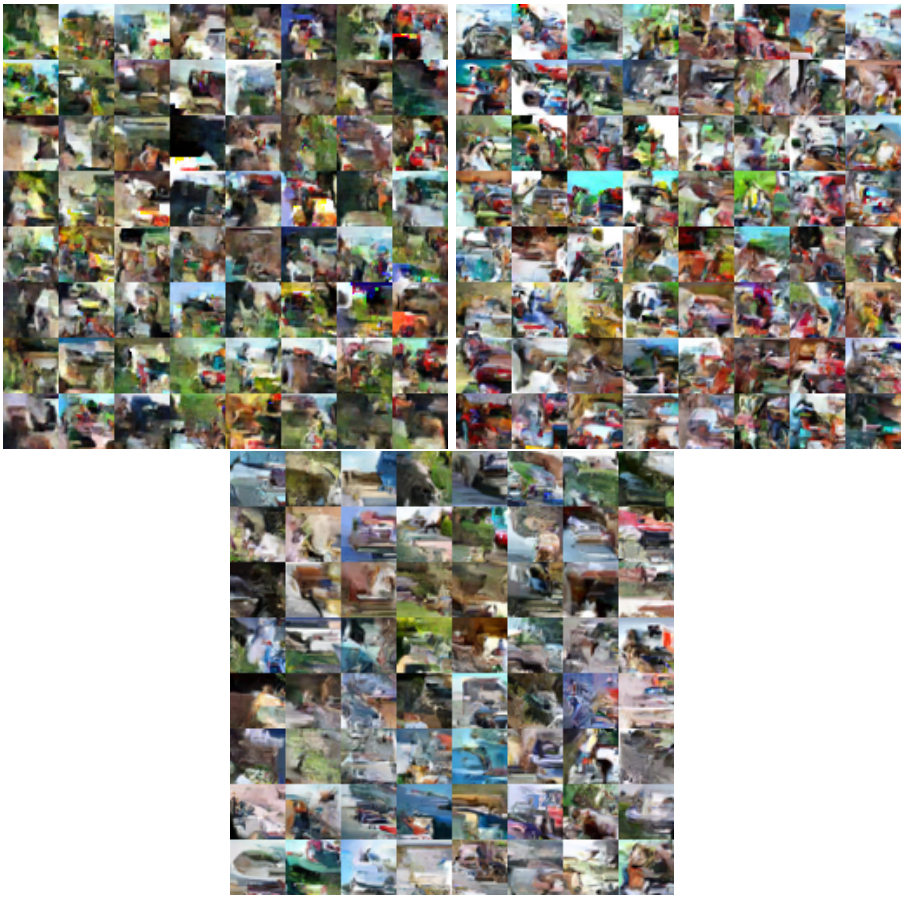
FIGURE B.7: Samples after iterations 9999, 29999 and 49999 during training PixelCNN on Fashion MNIST without Footwear (classes 5,7,9)

The results for novelty detection signals is yet to be evaluated.

B.1.3 CIFAR-10

The pixelcnn model generates the images in ??. The images have a lot of local structure but most often don't make global coherent sense.

FIGURE B.8: Samples after iterations 4999, 59999 and 449999 during training PixelCNN on CIFAR-10 dataset



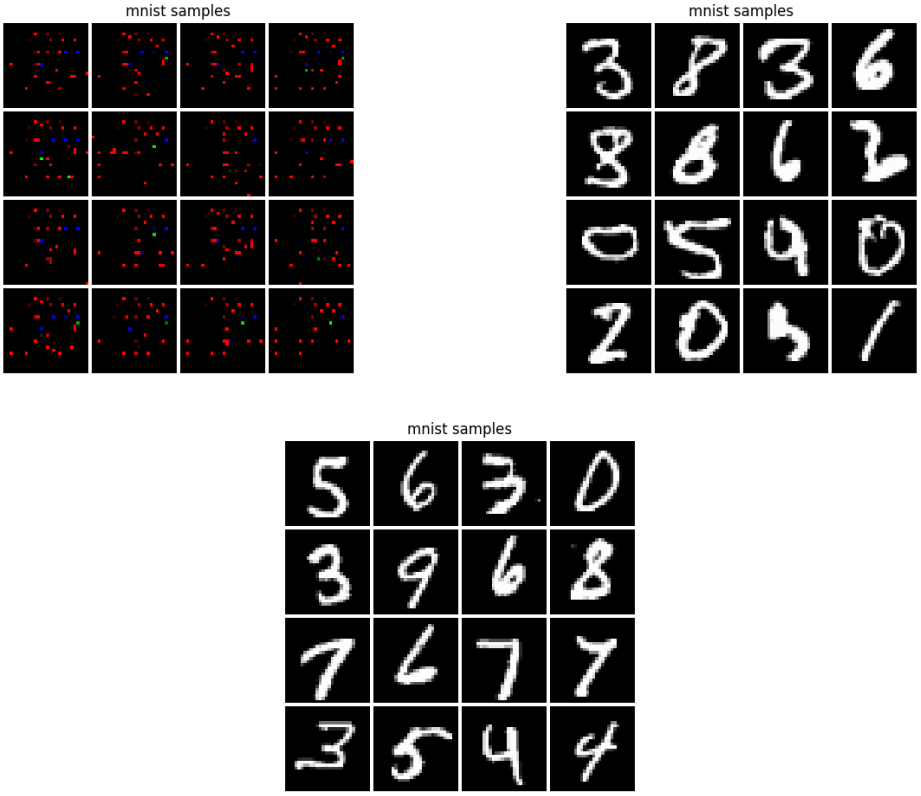
B.2 PixelCNN++

PixelCNN++ helps to overcome some of the shortcomings of the PixelCNN [25].

B.2.1 MNIST

PixelCNN++ yielded more perceptually appealing samples as compared to simple PixelCNN in the case of MNIST. The samples obtained are shown in Figure B.9.

FIGURE B.9: Samples after epochs 0, 10 and 72 during training PixelCNN++ on MNIST data



B.2.2 CIFAR-10 conditioned on labels

PixelCNN++ on CIFAR 10 condtioned on labels generates more coherent samples than its unconditional prior (refer B.10.

B.3 VAE

B.3.1 MNIST

VAE for MNIST produces somewhat good perceptual images with a single layer multi-layer perceptron (MLP) enocder and decoder. Using a few convolutional layers provides improved quality as shown in Figure B.11.

The CIFAR-10 images generated by a similar convolutional VAE produces blurry images, and further training does not seem to improve the image quality or reconstruction loss (refer to Figure B.12)

FIGURE B.10: Samples after 0, 140, 240 epochs after training PixelCNN++ conditionally on CIFAR-10 data

FIGURE B.11: Samples after iterations 0, 40, and 90 epochs on VAE with convolutional layers for MNIST dataset

B.3.2 STL10

STL-10 dataset [4] is a subset of the ImageNet dataset but has 10 classes for train and test. We trained a VAE using Residual block layers on this dataset and observed the samples as shown in Figure B.13

FIGURE B.12: Original and reconstructed samples on VAE with convolutional layers for CIFAR-10 dataset

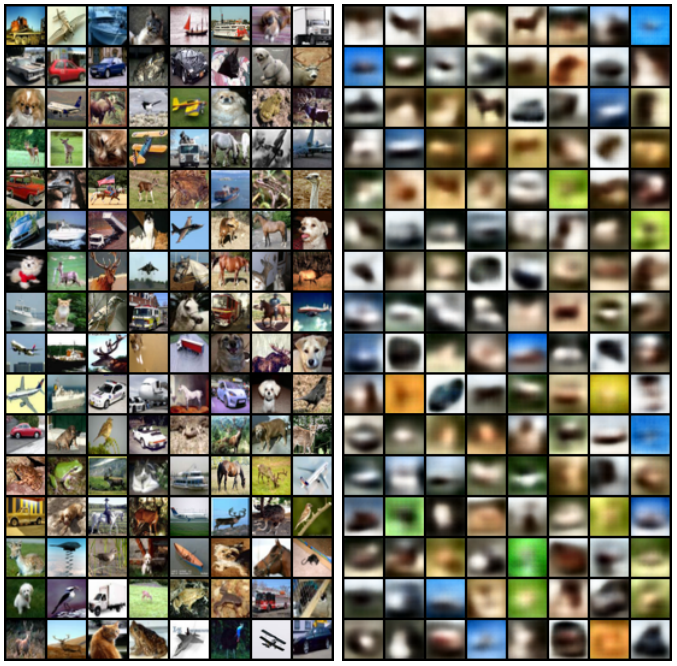
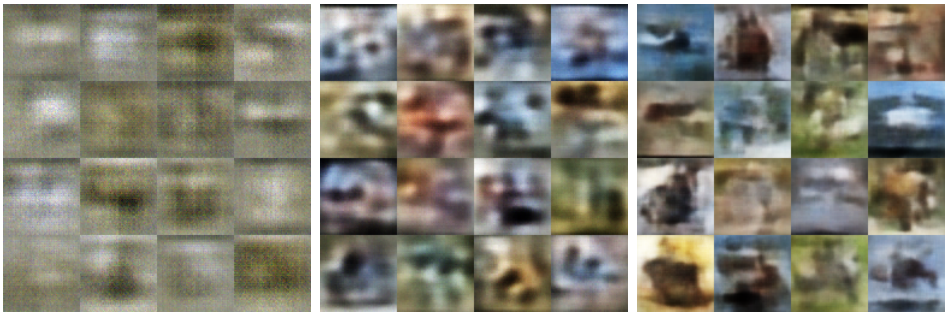


FIGURE B.13: Samples after iterations 99, 1099 and 6399 during training VAE on STL10 dataset

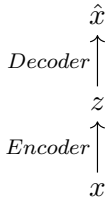


Appendix C

Autoencoder

C.1 Autoencoder

An autoencoder models the density function of the given training data so that it can be used to reconstruct the input. We have two parts of an autoencoder: an encoder, that maps the data x to latent space z, and a decoder, that maps a sample from z to reconstruct the output $\hat{x}$. It is a feed-forward network with back-propagation applied to the minimize the L2 loss between the output and the input data, that is, minimize $||x - \hat{x}||^2$.



The latent space z must be a lower dimensional feature representation than input space x so that by training, z is forced to capture meaningful factors of variation in the data.

Autoencoder can be used to initialize a supervised learning model with better features learned through the encoder. It is used as a dimensionality reduction by learning a lower dimensional representation of the latent variable z.

Autoencoder does not provide the density function of the latent space z. Though we can reconstruct using an autoencoder, we cannot sample from the latent distribution to produce new samples.

C.2 Alternatives to Variational Inference

Another way to estimate the marginal probability of x is using sampling techniques like Markov Chain Monte-Carlo approaches like Gibbs Sampling, or Metropolis-Hastings. This approach is unbiased but with high variance. On the other hand, variational inference approach has no or little variance but it is a biased estimator

Bibliography

[1] Marc G. Bellemare et al. "Unifying Count-Based Exploration and Intrinsic Motivation". In: *CoRR* abs/1606.01868 (2016). arXiv: 1606.01868. URL: http://arxiv.org/abs/1606.01868.

[2] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009), 15:1–15:58. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: http://doi.acm.org/10.1145/1541880.1541882.

[3] Xi Chen et al. "Variational Lossy Autoencoder". In: *CoRR* abs/1611.02731 (2016). arXiv: 1611.02731. URL: http://arxiv.org/abs/1611.02731.

[4] Adam Coates, Andrew Ng, and Honglak Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, 2011, pp. 215–223. URL: http://proceedings.mlr.press/v15/coates11a.html.

[5] Jesse Davis and Mark Goadrich. "The Relationship Between Precision-Recall and ROC Curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 233–240. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143874. URL: http://doi.acm.org/10.1145/1143844.1143874.

[6] Samuel Gershman and Noah Goodman. "Amortized inference in probabilistic reasoning". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 36. 36. 2014.

[7] Ali Ghodsi. *Ali Ghodsi, Lec : Deep Learning, Variational Autoencoder, Oct 12 2017 [Lect 6.2]*. Data Science Courses. Oct. 14, 2017. URL: https://www.youtube.com/watch?v=uaaqyVS9-rM.

[8] Ali Ghodsi. *Ali Ghodsi, Lec 16: Variational Autoencoders*. Data Science Courses. Mar. 23, 2017. URL: https://www.youtube.com/watch?v=weipjHmkCHk.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[10] Ishaan Gulrajani et al. "PixelVAE: A Latent Variable Model for Natural Images". In: *5th International Conference on Learning Representations*. 2017.

[11] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: *CoRR* abs/1610.02136 (2016). arXiv: 1610.02136. URL: http://arxiv.org/abs/1610.02136.

[12] Kyle Kastner. *Pixel Recurrent Neural Networks, A Review*. Magenta team, Tensorflow. URL: https://github.com/tensorflow/magenta/blob/master/magenta/reviews/pixelrnn.md.

[13] D. P Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: *ArXiv e-prints* (Dec. 2013). arXiv: 1312.6114 [stat.ML].

[14] Diederik P Kingma. "Variational inference & deep learning: A new synthesis". In: (2017).

[15] Diederik P. Kingma, Tim Salimans, and Max Welling. "Improving Variational Inference with Inverse Autoregressive Flow". In: *CoRR* abs/1606.04934 (2016). arXiv: 1606.04934. URL: http://arxiv.org/abs/1606.04934.

[16] Mark Kliger and Shachar Fleishman. "Novelty Detection with GAN". In: *CoRR* abs/1802.10560 (2018). arXiv: 1802.10560. URL: http://arxiv.org/abs/1802.10560.

[17] Yann Lecun and Corinna Cortes. "The MNIST database of handwritten digits". In: (). URL: http://yann.lecun.com/exdb/mnist/.

[18] Shiyu Liang, Yixuan Li, and R. Srikant. "Principled Detection of Out-of-Distribution Examples in Neural Networks". In: *CoRR* abs/1706.02690 (2017). arXiv: 1706.02690. URL: http://arxiv.org/abs/1706.02690.

[19] Shakir Mohamed and Danilo Rezende. *UAI 2017 Tutorial: Shakir Mohamed Danilo Rezende*. UAI Community. Nov. 8, 2017. URL: https://www.youtube.com/watch?v=JrO5fSskISY.

[20] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *CoRR* abs/1601.06759 (2016). arXiv: 1601.06759. URL: http://arxiv.org/abs/1601.06759.

[21] Aäron van den Oord et al. "Conditional Image Generation with PixelCNN Decoders". In: *CoRR* abs/1606.05328 (2016). arXiv: 1606.05328. URL: http://arxiv.org/abs/1606.05328.

[22] Marco A. F. Pimentel et al. "Review: A Review of Novelty Detection". In: *Signal Process.* 99 (June 2014), pp. 215–249. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2013.12.026. URL: http://dx.doi.org/10.1016/j.sigpro.2013.12.026.

[23] Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *arXiv preprint arXiv:1505.05770* (2015).

[24] M. Sabokrou et al. "Adversarially Learned One-Class Classifier for Novelty Detection". In: *ArXiv e-prints* (Feb. 2018). arXiv: 1802.09088 [cs.CV].

[25] Tim Salimans et al. "PixelCNN++: A PixelCNN Implementation with Discretized Logistic Mixture Likelihood and Other Modifications". In: *Submitted to ICLR 2017*. 2016.

[26] Tim Salimans et al. "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications". In: *CoRR* abs/1701.05517 (2017). arXiv: 1701.05517. URL: http://arxiv.org/abs/1701.05517.

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. Aug. 28, 2017. arXiv: cs.LG/1708.07747 [cs.LG].

[28] Serena Young, Fei-Fei Li, and Justin Johnson. *Lecture 13 — Generative Models*. Stanford University School of Engineering. Aug. 11, 2017. URL: https://www.youtube.com/watch?v=5WoItGTWV54.

[29] Shengjia Zhao, Jiaming Song, and Stefano Ermon. "InfoVAE: Information Maximizing Variational Autoencoders". In: *CoRR* abs/1706.02262 (2017). arXiv: 1706.02262. URL: http://arxiv.org/abs/1706.02262.