Improving Image Classification Through Generative Data Augmentation

Nielsen, Christopher Stephen

UNIVERSITY OF CALGARY

Improving Image Classification Through Generative Data Augmentation

by

Christopher Stephen Nielsen

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING

CALGARY, ALBERTA

MAY, 2019

Abstract

As the industrial adoption of machine learning systems continues to grow, there is incredible potential to use this technology to revolutionize how medical diagnostic imaging is performed. The ability to accurately classify the information contained within a medical image is of critical importance for clinical implementation. Successful application of machine learning classification algorithms has traditionally relied on the availability of copious amounts of labelled training data. Unfortunately, medical datasets are typically small due to privacy constraints and the large cost associated with annotating the data. To ameliorate this limitation, a training scheme is developed in this thesis which can operate on small-scale datasets by using a generative adversarial network to augment the dataset with synthetic images. Through quantifying the uncertainty in the classification network, training samples are selected to maximize the performance of the classifier while minimizing the amount of required data. Furthermore, privacy constraints are preserved as the images sampled from the generative adversarial network are inherently anonymized. The experimental results demonstrate the efficacy in this approach and viability for application in the medical domain.

Acknowledgements

I would like to thank my supervisor Dr. Michal Okoniewski for his guidance during the development of this thesis. His support has been critical to reach the completion of this project. I would also like to thank my fiancée for making life beautiful through her unwavering love and support. To my parents for the love, care, and guidance they have given me my whole life. To my brother for the thunder in his heart and his desire to keep things rad. To Dan for his friendship, support, and encouragement to take the hard road. To Karel for the enlightening conversations and his guidance to make every move like a lightning.

Dedication

This work is dedicated to my beautiful Adelina whose love continuously inspires me.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
GAN	Generative Adversarial Network
SGD	Stochastic Gradient Descent
CNN	Convolutional Neural Network
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
IS	Inception Score
FID	Fréchet Inception Distance
PGGAN	Progressive Growing of Generative Adversarial Network
DCGAN	Deep Convolutional Generative Adversarial Network
MNIST	Modified National Institute of Standards and Technology
LSUN	Large-Scale Scene Understanding
ISIC	International Skin Imaging Collaboration
HAM10000	Human Against Machine with 10000 Training Images
BALD	Bayesian Active Learning by Disagreement
RL	Reinforcement Learning
GPU	Graphics Processing Unit
CPU	Central Processing Unit
GDPR	General Data Protection Regulation
MC	Monte Carlo
ELBO	Evidence Lower Bound
VAE	Variational Autoencoder
KL	Kullback-Leibler

Chapter One: Introduction

Over the course of the last decade, machine learning algorithms have achieved unprecedented success in a wide spectrum of domains. Krizhevsky et al. (2012) released the first deep convolutional neural network to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This seminal work created a paradigm shift towards the use of deep neural networks for computer vision tasks. Goodfellow et al. (2014) published the Generative Adversarial Network (GAN) architecture, a game theoretic approach for training a pair of deep neural networks in competition to generate realistic samples from a dataset. Current state of the art GAN architectures are capable of generating images of human faces that are virtually indistinguishable from real photos (Karras et al. 2018). Silver et al. (2016) released AlphaGo, the first Artificial Intelligence (AI) system to beat the human world champion in the game of Go. The success of such systems can be largely attributed to the following factors:

- **Network Architecture**: Artificial neural networks have become the workhorse of the machine learning industry ultimately due to their incredible ability to learn which features to extract from data. Several key engineering innovations have been made to encourage convergence for deeper networks, enabling expressive hierarchical feature learning (Liu et al. 2017).
- **Computation**: Advancements in graphics processing unit (GPU) technology has catalyzed the training of deep convolutional neural networks due to the highly parallel nature of the computation. The efficiency gain when training using a GPU over a central processing unit (CPU) can be greater than an order of magnitude (Lawrence et al. 2017). The ubiquity and

relative low cost of consumer GPUs has enabled high capacity networks to be trained in a timely manner.

- **Data**: Having copious amounts of labelled training data is key for supervised learning tasks. The growth of the internet has stimulated the development of a global platform for hosting, collecting, and sharing data. Platforms for crowdsourced data labelling such as Amazon's Mechanical Turk, have enabled large scale datasets consisting of millions of datapoints such as ImageNet to be constructed (Deng et al. 2009, Callison-Burch et al. 2010).
- **Open-Source Initiative**: There has been a fundamental shift in the way that tech giants such as Google or Facebook operate in the field of machine learning. Instead of keeping all models proprietary, these companies are releasing code under an open-source license (Abadi et al. 2016). Academic research development is now accelerated through accessibility to state-of-the-art models.

1.1 Problem

The purpose of image classification is to identify the different objects contained in an image. There are a vast number of applications which rely on this technology. These include the use of convolutional neural networks (CNNs) for image classification as a critical component of the computer vision systems for driverless cars (Bojarski et al. 2016) and for facial recognition on social media platforms (Taigman et al. 2014). Recently, deep neural network models have been applied for medical image classification. As millions of medical images are captured and analyzed by radiologists each year, integrating AI and machine learning into the medical system has the potential to greatly improve diagnosis efficiency. Organizations such as the Canadian Radiology

Association are pushing to incorporate increased utilization of machine learning for medical imaging (Tang et al. 2018). Machine learning systems are capable of radiologist level diagnostics as demonstrated by Rajpurkar et al. (2017) where a deep neural network was shown to achieve radiologist level performance for pneumonia detection.

Despite the success of image classification using deep neural networks, several questions remain:

- **Data Privacy**: One of the most pertinent questions in present day machine learning is how to deal with data privacy. In medicine, patient privacy and confidentiality have always been of utmost importance. Therefore, a critical question is how to handle medical data for machine learning in a way which preserves the privacy of the individual in question but provides the necessary information to successfully train diagnostic models. This question extends beyond medicine and into the general private sector. On May 25, 2018 the General Data Protection Regulation (GDPR) was implemented within the European Union which mandates several restrictions over how personal data must be treated within organizations to comply with privacy regulations (Kingston, 2017). While the practical implications of this policy as still being assessed, the global push towards data privacy may require machine learning algorithms to be modified to achieve compliance with such regulation.
- **Small Datasets**: Empirically it has been observed that training deep neural networks requires a significant amount of data to avoid overfitting (Caruana et al. 2001). The mantra of the deep learning community has been to use as much data as possible for training neural networks. During the 2012 ILSVRC, there were roughly 1.2 million images used for training (Krizhevsky et al. 2012). Mahajan et al. (2018) demonstrated how a training set of roughly 3.5 billion images from Instagram could be used to achieve state-of-the-art

classification performance. To train neural networks with datasets at this scale requires a significant computational infrastructure likely beyond the capabilities of a standard academic institution. As the volume of global data continues to grow at a staggering rate, a pressing question is how to best organize this data into suitable datasets for machine learning training. Furthermore, application domains such as medical imaging have dataset sizes which are orders of magnitude smaller than the data captured by social media platforms. Therefore, image classification algorithms must be adapted to provide the necessary performance when only small datasets are available.

- **Unbalanced Datasets**: A common issue when training image classifiers on medical dataset is class imbalance. Most medical datasets have a large ratio of benign to malignant training examples. To achieve the required classification performance various oversampling and statistical weighting techniques have been applied (Rahman et. al 2013). This is especially important in the medical domain since false negatives are significantly more detrimental than false positives.

1.2 Purpose

In this thesis a classification framework is proposed and implemented that simultaneously addresses the questions of dataset privacy, small-scale, and imbalance when training a classification network. The core ideas behind this framework are 1) we train a GAN on the dataset to generate synthetic images that can augment the training dataset and 2) we quantify the classifier prediction uncertainty to sample the most informative GAN generated images for augmentation. These ideas form the basis of a feedback loop that cycles between training the classification network on the current training set and using the network prediction uncertainty to augment the

training set with new images. This feedback loop, and specifically the ability to select GAN samples which maximize classification performance, are the key innovative contributions provided by this thesis.

Although the dataset used to train a GAN may be private, the samples generated by the GAN are largely anonymized. Beaulieu-Jones et al. (2017) demonstrated how GANs could be used as a privacy preserving mechanism for clinical data sharing. The level of anonymity within a dataset can be quantified through the notion of differential privacy (Abadi et al. 2016). Intuitively the definition of differential privacy states that if any data point from a dataset is removed, then the resulting statistics computed based off this dataset do not change significantly. This constraint insures that the private data of each individual data point in the dataset is sufficiently anonymized. As an application, consider a hospital working in collaboration with a research institute, where the research institute is training a diagnostic image classifier that requires the hospital data. Instead of the hospital sending the raw patient data, the hospital can deliver a GAN trained on the private data and capable of generating samples similar to the patient data but fully anonymized. In addition to the privacy benefits, the GAN architecture provides a natural way to augment a dataset by sampling synthetic images to either increase the dataset size or balance the amount of data between classes. As a point of notation, we shall refer to data from the underlying non-synthetic dataset as *raw* data.

The experimental work in this thesis aims to address the following research questions:

1) What is the difference in classifier performance if we train using purely GAN synthesized data vs. raw data?
2) What is the difference in classifier performance if we train using randomly chosen samples vs. samples chosen based off classifier prediction uncertainty?
3) How is the capacity of the GAN used to generate training images correlated with the final classification performance?
4) What overall performance gain can we achieve from using GAN augmentation?

1.3 Qualitative Assessment of GAN Augmentation

Naturally an important question is whether using a GAN to generate samples to augment a dataset has any benefit, or if it is simply insidiously self-referential. To address this question let us consider what data augmentation entails. If we have a data point $\mathbf{x}$ and are given the ground truth conditional distribution for the class label $P(y|\mathbf{x})$, then a proper augmentation of the data is an augmentation function of the data $f(\mathbf{x})$, such that $P(y|\mathbf{x}) = P(y|f(\mathbf{x}))$. In other words, a proper augmentation does not change the underlying label of the data. Common augmentation functions include geometric transformations such as rotations, shifts, and flips as well as color transformations. It has been shown that using data augmentation during training can improve the performance of the classifier (Wang et al. 2017). We can explain the increase in classifier performance through recognizing that implicit constraints are imposed upon the data when we define an augmentation function as label preserving. For example, consider building a classifier to detect circles. We can rotate and translate the circle training images while preserving the 'circle' label, however if we

scale the image nonuniformly we will no longer have a circle, but an ellipse. Therefore, providing the classifier with augmented training data that implicitly emphasizes which transformations are label preserving helps the classifier to learn the structure of the data, resulting in improved classification performance. To prove that a GAN can provide valuable information for data augmentation, we must demonstrate that it is capable of learning transformations of the data which are label preserving. As a qualitative justification, consider Figure 1-1, where two images of the digit '1' generated from a GAN trained on the MNIST handwritten digits dataset are shown. Each of these images was generated by passing a vector sampled from the GAN latent space through the generator network. We can interpolate between these vectors and generate the intermediate images shown in Figure 1-2. Notice how these images appear to be progressively rotating. This implies that the latent space for the GAN has learned to encode rotation. Therefore, we can qualitatively infer that the GAN is capable of generating images which are augmented by rotation. It is important to note that no other information was provided to the GAN during training other than the MNIST images. Hence, the rotation encoding in the latent space was learned directly from the structure of the data. This result provides a qualitative justification that a GAN is capable of implicitly learning appropriate augmentation functions such as rotation which have potential to benefit data augmentation strategies.

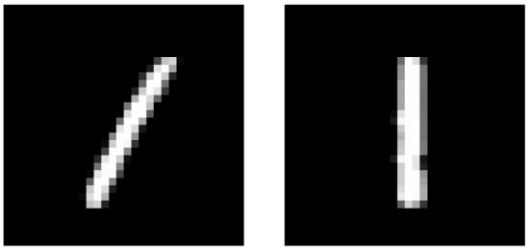

Figure 1-1 Samples generated from a GAN trained on the MNIST dataset.

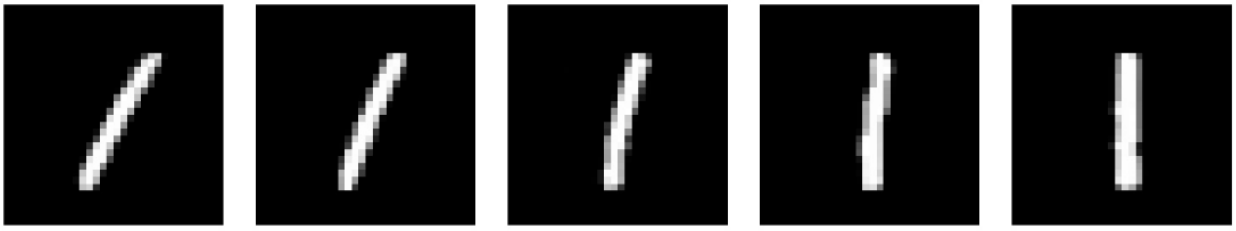

Figure 1-2 GAN samples generated from latent space interpolation.

1.4 Objectives

The main objectives for this thesis are the following:

- Develop a classification framework using the TensorFlow (Abadi et al. 2016) and Keras (Chollet et al. 2015) Python libraries to experimentally address the research questions described in Section 1.2 using the following datasets:
    - **MNIST**: The Modified National Institute of Standards and Technology (MNIST) dataset consists of 60,000 training images and 10,000 testing images of handwritten digits (LeCun et al. 1998). The MNIST dataset has long served as a classic benchmark for computer vision and machine learning algorithms.
    - **LSUN**: The Large-scale Scene UNderstanding (LSUN) dataset consists of thousands of images from 10 different physical environments including *dining rooms, living rooms, bedrooms, bridges, kitchens, classrooms, restaurants, church outdoors, towers,* and *conference rooms* (Yu et al. 2015).
    - **ISIC 2018**: The International Skin Imaging Collaboration (ISIC) 2018 Challenge dataset consists of 10015 dermoscopic lesion images from seven different lesion categories consisting of *melanoma, melanocytic nevus, basal cell carcinoma, actinic keratosis, benign keratosis, dermatofibroma,* and *vascular lesion* (Codella

et al. 2017). The images in this dataset were originally taken from the "Human Against Machine with 10000 training images" (HAM10000) dataset (Tschandl et al. 2018). The ISIC 2018 Challenge was designed to encourage researchers to develop high performance classification algorithms on this dataset.

- Demonstrate how the Progressive Growing of GAN (PGGAN) and the Deep Convolution GAN (DCGAN) architectures can be trained on the MNIST, LSUN, and ISIC 2018 datasets. Investigate how the loss of the generator and discriminator networks converge. Show how the quality and diversity of the samples can be measured using the Inception Score (IS) and the Fréchet Inception Distance (FID) metric. Demonstrate how the IS and FID scores change as the capacity of the GAN is increased.
- Develop the Convolutional Neural Network (CNN) architectures used for image classification. For the MNIST dataset, design a CNN with suitable capacity that can be trained efficiently from random weight initialization. For the LSUN and ISIC 2018 datasets, utilize the MobileNet architecture (Howard et al. 2017) pretrained on the ImageNet dataset as a base model. Perform transfer learning to finetune the weights of the network and demonstrate the convergence of the network. Quantify the performance of the CNN architectures using balanced accuracy as well as the multiclass Receiver Operating Characteristic (ROC) metric.
- Demonstrate how dropout layers can be used in the CNN architectures to model Bernoulli prior distributions over the CNN weights. Show how Monte Carlo (MC) samples can be acquired from the CNN to estimate the posterior uncertainty. Analyze the output of the GAN discriminator networks to determine appropriate thresholds to filter samples from the GAN based on the likelihood of being a representative sample from the underlying dataset.

Develop acquisition functions using random sampling, Bayesian Active Learning by Disagreement (BALD), and maximum entropy to sample the images which provide the greatest information gain for CNN training.

- Perform experiments using the developed classification framework to address each of the research questions proposed in Section 1.2.

1.5 Contributions

The following describes the main contributions made by this thesis:

- Designed an importance sampling mechanism to prioritize GAN samples based on the classification network uncertainty to maximize the final classification performance.
- Demonstrated how a PGGAN architecture could be trained to synthesize high resolution medical images representing the ISIC 2018 dataset.
- Developed an iterative feedback training loop to incrementally build up the training set from GAN generated images to maximize the final performance of the classifier.
- Thesis work has been accepted for publication in the CVPR 2019 Workshop on Uncertainty and Robustness in Deep Visual Learning (Nielsen et al. 2019).

1.6 Thesis Overview

The remainder of the thesis is structured as follows. Chapter 2 provides a literature review and an overview of the machine learning technologies used for this thesis. Chapter 3 describes the classification framework developed in this thesis, provides an outline of the preprocessing operations applied to each dataset, and defines the architecture of the neural networks used for the

GAN and classification models. Chapter 4 presents the results of the experimental work conducted for this thesis. Specifically, the generated image quality is assessed for the trained GANs, a qualitative interpretation of the classification network uncertainty is reported, and the results from the classification experiments are used to address the thesis research questions. Chapter 5 investigates possible applications for the developed technology. Chapter 6 provides concluding remarks, contributions made by the thesis, and a discussion of potential future developments.

Chapter Two: Background

The purpose of this chapter is to provide a literature review of related work and an introductory overview of the machine learning techniques applied for this thesis. Section 2.1 will present a literature review of related work and describe the historical development of the applied machine learning models. Section 2.2 will discuss the algorithmic techniques behind each of the machine learning models used in this thesis. Section 2.3 will summarize the presented material and motivate the work proposed by this thesis.

2.1 Related Work

2.1.1 Image Classification

Over the last 50 years, and specifically in the last decade, the development of deep neural networks for image classification has progressed from being an academic niche, to becoming a mainstream industrial technology. In 1957, Frank Rosenblatt of the Cornell Aeronautical Laboratory developed one of the first machine learning classifiers, called the perceptron (Rosenblatt 1957). The perceptron was a machine designed for image recognition and consisted of 400 photocells randomly connected to neurons whose weights were encoded in potentiometers. During the training procedure, the weights were updated by electric motors. At the time of release, the perceptron generated a large amount of public interest, however the expectations of its capabilities were highly exaggerated. In a 1958 press conference organized by the US Navy, it was reported that the perceptron was "the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence" (Olazaran 1996). As the perceptron was incapable of learning an XOR function, let alone being self aware, the hype

from this development eventually gave way into a downturn in AI research. It was not until 1986 when the modern interpretation of neural networks was entrenched with the development of backpropagation (Rumelhart et al. 1986). In 1998, the MNIST dataset was released and the first CNN for image classification was developed (LeCun et al. 1998). A CNN is a neural network architecture that contains convolutional layers. Further details on the CNN architecture are presented in Section 2.2.1. A critical development, which catalyzed the development of machine learning over the last decade, was the release of the ImageNet dataset (Deng et al. 2009). The ImageNet dataset contains more than 14 million images from more than 20,000 categories that have been hand-annotated by humans using the Amazon Mechanical Turk crowdsourcing platform. Starting in 2010, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was run where competitors could compete to build classifiers capable of the best performance on the dataset (Russakovsky et al. 2014). It was in 2012 that the first deep convolutional neural network called AlexNet competed in the ILSVRC (Krizhevsky et al. 2012). AlexNet not only won the competition but was able to achieve a 10.8 percentage point improvement over the next runner up for top-5 error performance. This unprecedented achievement catalyzed the development and research of deep convolutional neural networks for classification tasks. In 2014, the winner of the ILSVRC was the Inception network (Szegedy et al. 2015). The main innovation was the development of Inception modules which allowed for convolutions with different size kernels to be processed in parallel. In 2015, the winner of the ILSVRC was the ResNet network (He et al. 2016). The main innovation provided by ResNet was the use of residual blocks which allowed for the networks to extend to unprecedented depths. The winning network had 152 layers. Huang et al. (2016) proposed the DenseNet architecture which uses dense blocks similar to the ResNet residual blocks to achieve trainable networks with over 200 layers. As most network architectures

consist of millions of parameters, they are difficult to use on low power devices such as mobile phones. Howard et al. (2017) released a network architecture called MobileNet which has far fewer parameters than AlexNet and achieves better performance on the ImageNet dataset.

2.1.2 Generative Models

Unsupervised learning is the process of extracting meaningful patterns from data that do not have given labels. An application of unsupervised learning is for estimating and generating samples from an underlying dataset distribution. Models of this type are called generative models. Salakhutdinov et al. (2007) proposed the restricted Boltzmann machine as a neural network architecture capable of learning a probability distribution over its set of inputs. Restricted Boltzmann machines can be trained using gradient descent and backpropagation. Another model variety capable of learning a representation of the underlying data distribution are variational autoencoders (VAEs). VAEs use an architecture consisting of an encoder and decoder neural network (Doersch 2016). The encoder network takes the input image and transforms it into a set of parameters describing an underlying latent space distribution. This latent distribution is than sampled and the resulting latent vector is passed through the decoder network which attempts to reconstruct the original image. VAEs are *autoencoders* since they learn to reconstruct the original image and are *variational* since they learn to approximate the underlying data distribution as a parameterized variational latent space. A VAE is trained by maximizing a lower bound on the log likelihood of the data called the Evidence Lower Bound (ELBO) (Kingma et al. 2013). The primary advantage of VAEs over other generative models is that the relationship between the data and the underlying latent space is directly modelled using the encoder and decoder networks. One

limitation of VAEs is that the ELBO loss function does not produce images which have the highest visual quality due to the averaging effects of maximizing the log likelihood.

Another class of generative models are autoregressive models. The essence of an autoregressive model is to learn the conditional distribution of every pixel in an image conditioned on all previously sampled pixels. Synthetic images can be generated one pixel at a time using the likelihood function learnt by the network. Models such as the PixelRNN have had great success in generating high quality samples (van den Oord et al. 2016). However, the sampling process is inefficient and does not directly model a low dimension latent space for the data.

A further variety of generative models are GANs. The develop of GAN architectures has been extremely rapid since their original conception in 2014. The initial paper on GANs was written by Goodfellow et al. (2014) and the focus of this original work was to describe the minimax competition between the discriminator and generator. Experimental verification of the technique was provided by qualitatively showing the quality of the generated images after training on the MNIST and the CIFAR-10 datasets. An extension was made by Mirza et al. (2014) to condition both the generator and discriminator model on the label of the training data, enabling samples to be generated from specific class labels. This work was expanded by Radford et al. (2015) when the DCGAN architecture was developed which used deep convolutional neural networks for both the generator and discriminator models. Additionally, it was shown how generated samples from the trained GANs could be used for semi-supervised learning, where the initial layers of the discriminator are used as a feature extractor to train a classification model. Denton et al. (2015) proposed the Laplacian GAN (LAPGAN) model where a cascade of discriminator and generator neural network models were trained at each level of a Laplacian pyramid to generate images in a

coarse to fine process. Odena et al. (2016) developed the auxiliary classifier GAN framework where in addition to predicting the validity of the given data, the discriminator was trained to classify the label of the real data. This addition to the discriminator was shown to provide better performance by teaching the discriminator to disentangle the features specific for different classes. Due to the instability of training GANs using original loss function presented by Goodfellow et al. (2014), Gulrajani et al. (2017) released an improved loss function based on the Wasserstein distance. Theoretically the Wasserstein loss has smoother gradients and greater stability over the loss function proposed by Goodfellow et al. (2014). The PGGAN architecture was released in 2017 and provided an approach to train a GAN architecture by training the discriminator and generator models on lower resolution samples before progressively growing toward high resolution samples (Karras et al. 2017). Samples generated from the PGGAN architecture after being trained on a celebrity face dataset were the first photorealistic generated images of humans by a GAN at the resolution 1024x1024 pixels. In late 2018, the style-based generator architecture for GANs was released, demonstrating how the generator network can be improved through the use of synthesis networks to customize the style being generated by the GAN (Karras et al. 2018b). The experimental analysis of GAN behaviour led to a number of different techniques described by Salimans et al. (2016) to improve the stability and measure the quality of the samples produced by a GAN. It was in this paper that the Inception Score was proposed as a benchmark to measure the quality and diversity of the sample generated by a GAN. Heusel et al. (2017) proposed the FID metric as an improved benchmark over the Inception Score for assessing GAN quality. Wang et al. (2017) demonstrated how GAN samples can be used for data augmentation. However, the images were sampled randomly from the GAN latent space and provided minimal improvement for the final classification performance.

2.1.3 Uncertainty Analysis

The question of how to quantify uncertainty has its roots in probability and estimation theory and has a rich history. During the 18th century Thomas Bayes proposed a mechanism now referred to as Bayes' rule which describes the probability of an event given prior knowledge of factors that might influence the event. Although the original concept was defined by Bayes, the effort to develop the idea was performed by Laplace where he used the Bayesian approach to estimate the mass of Saturn with a high degree of accuracy (Sivia et al. 2011). Interestingly during much of the 20th century, Bayesian statistical methods were much less popular compared to frequentist statistical methods due to the philosophical and practical concerns associated with choosing appropriate prior distributions and computation of the posterior. Bayesian methods gained significant popularity with the discovery of Markov Chain Monte Carlo (MCMC) methods which enabled sampling from complex posterior distributions (Andrieu et al. 2003). Development effort was directed toward transforming neural networks into Bayesian networks by placing prior distributions over the weight parameters in the network (Tishby et al. 1989). Radford (1996) demonstrated that a neural network with prior distributions over the weights and infinitely wide hidden layers corresponds to a Gaussian Process model. Recently an approximation approach was developed using dropout to learn the prior distribution over the weights of the network through a process called Bayes by Backprop (Blundell et al. 2015). The advantage of this method is that the prior distribution of each parameter is learned simultaneously as the network is trained. Additionally, computing samples of the posterior using this method is highly efficient.

2.2 Machine Learning Algorithms

The purpose of this section is to present the machine learning techniques used to build the classification framework for this thesis.

2.2.1 Image Classification

The classification task can be defined in the following way: we have a dataset consisting of input data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ and corresponding labels $\mathbf{Y} = \{y_1, y_2, \cdots, y_N\}$ and our goal is to model the conditional distribution $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ such that we can make inferences on this distribution to find the optimal label y to assign to a new data point $\mathbf{x}$. One method to solve this problem is to approximate the discriminative distribution using a parameterized function. In this case we have a function $\mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}) = P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ where $\boldsymbol{\omega}$ is the set of parameters describing the function. Deep neural networks are one such functional form that have had tremendous success at approximating the discriminative distribution. The remainder of this section will describe the structure of deep neural networks and how they can be trained for the classification task.

2.2.1.1 Neural Network Architecture

The inspiration for neural networks initiated from attempting to model the biological neural structure in the human brain. At a high level, a biological neuron senses stimulus from dendritic connections, combines these signals, and if the combined signal surpasses an activation threshold, an output signal propagates down the axon towards other neurons. In a similar way, an artificial neuron receives input from the neurons in the previous layer, combines this input and passes the

combined signal through an activation function whose output it propagated to the next layer. A neural network is composed of layers of neurons typically connected in a directed acyclic fashion, known as a feedforward network. The inputs to a neuron are combined as an affine transformation. The activation function is a nonlinear function that takes the result of this affine transformation and passes forward the output to the next layer of neurons. The critical component that enables the functional approximation power of neural networks is the nonlinearity of the activation function. This gives the neural network the ability to model highly complex transformation. This is described formally by the universal approximation theorem (Csáji 2001). If the activation function was linear, then the entire neural network would collapse down into a single affine transformation.

Let us consider a simple single layer neural network which maps an input vector $\mathbf{x} \in \mathbb{R}^4$ into a scalar output $y \in \mathbb{R}^1$. We can write the expression for this mapping as $y = f(\mathbf{W}\mathbf{x} + \mathbf{b})$ where $f(\ )$ is a nonlinear activation function, $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is the bias vector. We can represent this network visually as seen in Figure 2-1.

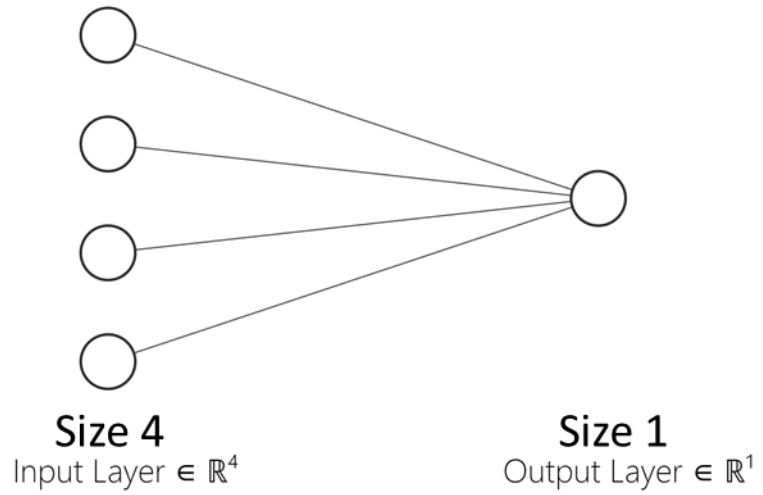

Figure 2-1 Simple single layer feedforward neural network.

We can extend this concept to networks with greater number of layers by simply taking the functional composition of neuron outputs. For example, consider modifying the neural network from Figure 2-1, by adding two hidden layers with sizes 10 and 3. We can now write the total composition of the neural network as follows $y = f_3\left(\mathbf{W}_3 f_2\left(\mathbf{W}_2 f_1\left(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1\right)+\mathbf{b}_2\right)+\mathbf{b}_3\right)$, where $\left\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\right\}$ are the weight matrices, $\left\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\right\}$ are the bias vectors, and $\left\{f_1(\ ), f_2(\ ), f_3(\ )\right\}$ are the activation functions for the layers of the neural network. A graphical representation of this neural network is shown in Figure 2-2.

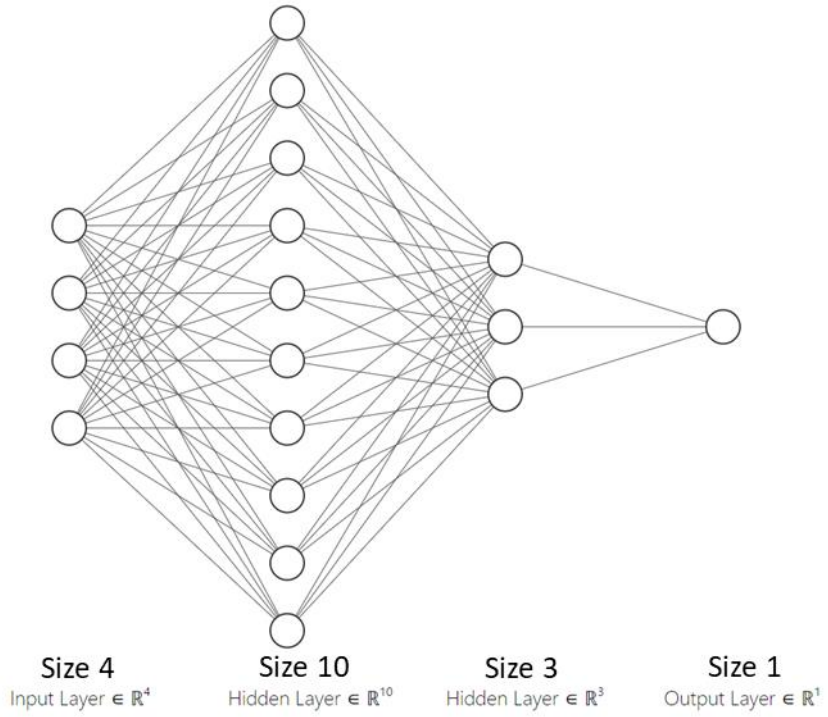

Figure 2-2 Multilayer feedforward neural network.

A neural network layer of the form $y = f\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$ is called densely connected since every neuron from the previous layer contributes to the stimulus of each neuron in the next layer. As image data is very high dimensional, using densely connected layers quickly becomes computationally intractable due to the large number of required parameters. Since most images exhibit local spatial structure it is possible build a network layer which uses a relatively small spatial neighborhood of neurons to compute the stimulus for the next layer. Computationally this local neighborhood combination can be performed by convolving the image with a set of learned kernels. Similar to the densely connected layers, a nonlinear activation function is applied after the stimulus has been computed. To reduce the dimensionality of the data, pooling layers are used which reduce the output from a spatial neighborhood of neurons to a single value. Max-pooling is commonly used and is performed by outputting the maximum value within a spatial neighborhood of neurons. The standard structure of a CNN is to have a series of convolutional layer at the beginning followed by densely connected layers at the end. An example of a 5-layer CNN is shown in Figure 2-3. The input image size is 128x128 pixels with 3 color channels. The first convolutional layer consists of a stack of 8 kernels each with size 7x7 pixels and max-pooling is used to reduce the image to 64x64 pixels. The second convolutional layer consists of a stack of 16 kernels each with size 5x5 pixels and max-pooling is used to reduce the image size to 32x32 pixels. The third convolutional layer consists of a stack of 32 kernels each with size of 3x3 and max-pooling is used to reduce the image size to 16x16 pixels. The image is then flattened into a vector and sent through the fourth layer which is a densely connected layer with size of 256. The final fifth layer is a densely connected layer of size 128.

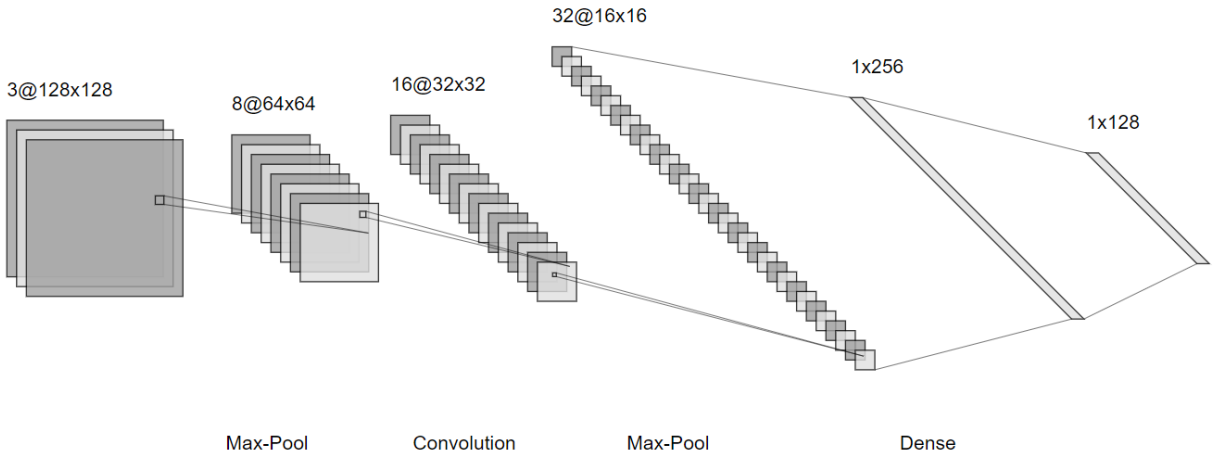

Figure 2-3 Convolutional neural network with 5 layers.

A wide range of activation functions are used in practice. Five of the activation functions used for this thesis are described below:

- **Sigmoid**: The sigmoid function is defined as $f(x)=\frac{1}{1+e^{-x}}$ and is a monotonically increasing function that maps a real value input into an output between 0 and 1. In neural network design, the sigmoid activation is commonly used for logistic regression to output a valid probability value.
- **Tanh**: The tanh function is defined as $f(x)=\frac{\left(e^{x}-e^{-x}\right)}{\left(e^{x}+e^{-x}\right)}$ and is a monotonically increasing function that maps a real value input into an output between -1 and 1.
- **ReLU:** The Rectified Linear Unit (ReLU) function is defined as $f(x)=\max(0,x)$ and is a piecewise linear function that is differentiable everywhere except at 0.
- **Leaky ReLU:** The Leaky ReLU function is defined as $f(x)=\begin{cases}\alpha x & for\ x<0\\ x & for\ x\geq 0\end{cases}$ and is a piecewise linear function similar to the ReLU with the difference being that when $x<0$

the function 'leaks' by having outputting the value αx where α is typically a small number such as 0.01.

- **Softmax:** The softmax activation function is slightly different in purpose to the activation functions described previously as it is a vector function rather than a scalar function. For a given vector of length N, the softmax function is defined as $f_i(\mathbf{x}) = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}}$. The softmax activation is used to normalize the output of a given layer into a valid probability distribution.

A key feature of the activation functions described above is the computational simplicity. Figure 2-4 displays a plot of these activation functions.

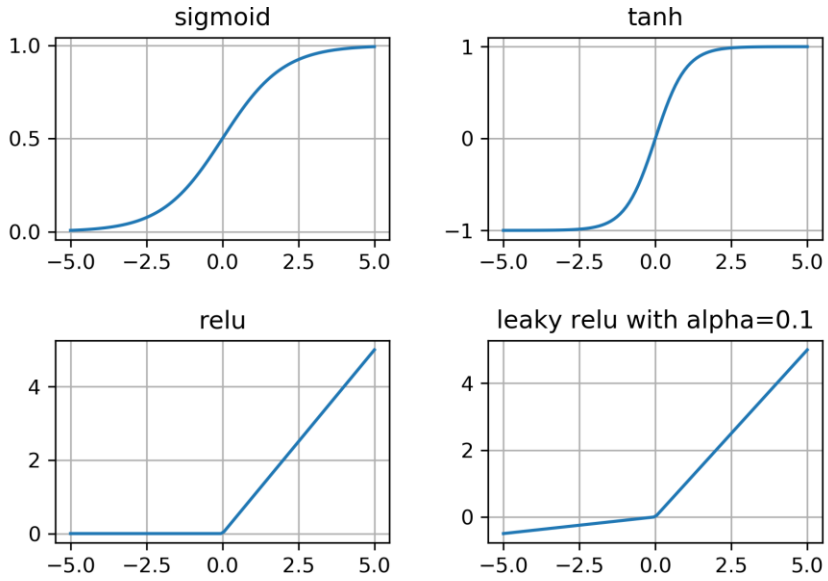

Figure 2-4 Activation functions.

Since the goal of a classification network is to produce an estimate of the conditional label distribution $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$, we must ensure that the output of the neural network is in fact a valid

probability function. A probability distribution is valid if all probabilities are between 0 and 1 inclusive, and the sum of the probabilities of all possible outcomes is equal to 1. One commonly used method to achieve this is to have the final activation of the network be a softmax function. This ensures that the output is a valid distribution. After computing the output probability distribution, the next step is to have some metric which can assess the quality of the predictions such that we can train the network to improve its performance. The most common way to do this is to compare the network label predictions and ground truth label distribution using cross-entropy loss (Janocha et al. 2017). For the sake of brevity in our notation, let us define $\hat{\mathbf{y}}$ as the neural network output estimate of $P\left(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}\right)$, and $\mathbf{y}$ as the ground truth label distribution for the sample. Suppose that we have N possible labels. The cross-entropy loss is defined as the cross-entropy between the ground truth distribution $\mathbf{y}$ and the estimated label distribution $\hat{\mathbf{y}}$. We can write the cross-entropy loss as

$$H\left(\mathbf{y}, \hat{\mathbf{y}}\right) = -\sum_{n=0}^{N-1} \mathbf{y}_n \log\left(\hat{\mathbf{y}}_n\right) \tag{2.1}$$

Since the ground truth distribution will be 0 everywhere except at the actual label index $\mathbf{y}_c$ where it will be 1, we can simplify the cross-entropy loss as follows

$$H\left(\mathbf{y}, \hat{\mathbf{y}}\right) = -\sum_{n=0}^{N-1} \mathbf{y}_n \log\left(\hat{\mathbf{y}}_n\right) = -\mathbf{y}_c \log\left(\hat{\mathbf{y}}_c\right) \tag{2.2}$$

To understand the intuition behind the cross-entropy loss, we can write the expression in terms of the Kullback-Leibler (KL) divergence (Shlens 2014) as

$$H\left(\mathbf{y}, \hat{\mathbf{y}}\right) = H\left(\mathbf{y}\right) + D_{\mathrm{KL}}\left(\mathbf{y} \| \hat{\mathbf{y}}\right) \tag{2.3}$$

Therefore, minimizing the cross-entropy loss with respect to $\hat{\mathbf{y}}$ is equivalent to minimizing the KL divergence between the ground truth distribution and the estimated distribution. During training it is common practice to process multiple samples simultaneously in a batch, hence the total loss for the neural network is the average cross-entropy loss over all of the input samples in the batch.

Before a neural network is trained, each parameter is typically assigned a starting value based on an initialization strategy (Hanin et al. 2018). One possible strategy is to use what is known as Xavier uniform initialization (Glorot et al. 2010). This technique initializes the parameters by randomly selecting values from the uniform distribution using a range that is inversely proportional to the number of neurons in the previous layer. Therefore, the larger the previous layer, the smaller the initialized values will be. The purpose of this approach is to assign initial values to parameters which would not cause the gradients to either explode or vanish at the start of training.

Training a neural network boils down to an effective use of the chain rule from rudimentary calculus. The individual functions which make up a neural network each have analytical gradients that can be computed in an efficient manner. Computing the gradient of the loss function with respect to the parameters in the network is accomplished through preceding backwards through the network using the chain rule to compute the gradients at each layer of the network. This procedure of calculating the gradients in a neural network is called backpropagation. As feedforward networks are directed acyclic graphs, computing the gradients in very large networks can be done efficiently. Software packages such as TensorFlow have been released to automate the gradient calculation (Abadi et al. 2016). After the gradients have been computed, a numerical optimizer is used to determine how each weight in the network must be adjusted to reduce the

overall loss value. The simplest optimization method is known as Stochastic Gradient Descent (SGD) where each parameter is updated proportional to its gradient. The constant of proportionality is called the learning rate and is a hyperparameter that must be specified before the network is trained (Ruder 2016). More sophisticated optimization methods such as AdaDelta and Adaptive Moment Estimation (Adam) have been developed to provide benefits over basic SGD by using momentum and adaptive learning rates in the gradient update equation (Zeiler 2012). When a dataset is highly unbalanced, it may be necessary to train the classifier using a weighted loss function (Janocha et al. 2017). A weighted loss function is computed by weighing the contribution of each individual training image inversely proportional to the number of occurrences that the training image label category has in the training set. This has the effect of placing more importance on underrepresented label categories. During training, the loss function can be monitored to analyze how well the neural network is learning. An epoch is defined as the number of training iterations required to process each image in the training set. A network will typically be trained for a fixed number of epochs. Knowing how many epochs is necessary to achieve convergence depends on the data and the network design.

Many techniques have been developed to improve the convergence of deep neural networks during training (Gu et al. 2015). A strategy known as batch normalization is often applied between the layers of a neural network to learn the statistics of the activation outputs (Ioffe et al. 2015). The activation statistics are then used to normalize the data such that the activations of the layer are uniformly scaled which improves the stability of the gradients during training. Another technique that is widely used is transfer learning. Training a large image classification network from scratch typically requires a lot of data. The goal of transfer learning is to combat this problem by

pretraining a network on a larger dataset and then finetuning the parameters on a smaller dataset. The reason why this is successful is that deep neural networks tend to learn a hierarchical representation of the data (Krizhevsky et al. 2012). As low-level geometric features such as lines and corners are common across a wide range of images, the features extractors learnt on larger datasets can be transferred for training classifiers on smaller datasets.

There are several metrics commonly used to evaluate the performance of a classification network (Hossin et al. 2015). One of the simplest and most commonly used methods is to calculate the prediction accuracy. Prediction accuracy is defined as the ratio of correct predictions to the total number of predictions. For unbalanced datasets, the concept of prediction accuracy can be extended to form the balanced accuracy metric. Balanced accuracy is defined as the average of the prediction accuracies for each individual label category. Another approach to measure classification performance is to use the ROC curve. The ROC curve is defined for a binary classifier as curve describing the true positive rate as a function of the false positive rate. The area under the curve (AUC) of the ROC can be used as a metric to assess the performance of a binary classifier (Hajian-Tilaki 2013). This methodology can be extended to a multiclass classifier by computing the average ROC AUC for each individual label.

2.2.2 Generative Networks

A generative network is a neural network that can be trained using images from a dataset to learn a representation of the underlying dataset distribution $p_{data}(\mathbf{x})$. Once trained, a generative network can be used to generate synthetic images which closely resemble images from the underlying

dataset. By defining the representation of $p_{data}(\mathbf{x})$ in terms of a lower dimensional space called a latent space, it is possible to disentangle abstract visual features, allowing specialized samples to be generated by manipulating the latent vector space. Due to sampling efficiency and the high visual fidelity of the generated images, GANs will be used in this thesis as the underlying generative model. There are two GAN architectures used in this thesis, the DCGAN and the PGGAN. The remainder of this section will examine the details of these particular GAN architectures as well as the metrics used to evaluate the diversity and quality of the generated samples.

2.2.2.1 GAN Architecture

As described in Section 2.1.2, a GAN consists of two networks called a generator and a discriminator which are trained competitively against each other. The generator attempts to generate samples indistinguishable from an underlying dataset, and the discriminator attempts to infer whether an image is synthesized by the generator or from the underlying dataset. A relevant analogy to the GAN architecture is to consider the competition between a counterfeiter and a bank. The counterfeiter attempts to generate fake currency which closely resembles real currency, while the bank develops the means to discriminate between real and fake currency. As this game is played, both the counterfeiter and the bank continue to improve their ability to generate and discriminate currency respectively. In game theory this is known as a zero-sum non-cooperative game, where the optimal convergence point is the Nash equilibrium (Mescheder et al. 2018).

To describe this formally, let us denote the generator and discriminator networks as G and D respectively. The goal of the generator is to take an input vector $\mathbf{z}$, which has been sampled from the generator's latent space $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ and to produce an image $G(\mathbf{z})$ which closely resembles a sample from the data distribution $p_{data}(\mathbf{x})$. Let the distribution of samples from $G(\mathbf{z})$ be described as the generator distribution $p_{generator}(\mathbf{x})$. The goal of the discriminator is to differentiate for a given sample $\mathbf{x}$ whether it was more likely to have been produced by $p_{data}(\mathbf{x})$ or $p_{generator}(\mathbf{x})$. If the discriminator believes that $\mathbf{x} \sim p_{data}(\mathbf{x})$, then it will output a value close to 1, likewise if the discriminator believes that $\mathbf{x} \sim p_{generator}(\mathbf{x})$, then it will output a value close to 0. We can now define the minimax loss function for a GAN as follows:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right] \tag{2.4}$$

Where $V(D,G)$ is called the value function and $\mathbb{E}$ represents the expectation operator. Therefore, the term $\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right]$ can be interpreted as the expected value of $\log D(\mathbf{x})$ where $\mathbf{x} \sim p_{data}(\mathbf{x})$. The intuition behind why the loss function uses the log function is to heavily penalize the discriminator for being incorrect. To understand what this loss function is trying to achieve we can look at each component separately.

- $\mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right]$: Intuitively this quantity describes how much the discriminator believes that samples from the dataset distribution are real. This quantity has no dependence on G.
- $\mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1 - D\left(G(\mathbf{z})\right)\right)\right]$: This quantity describes how much the discriminator believes that the samples from the generator are fake. During the minimax game, D will attempt to

maximize this quantity by classifying the generator samples as fake, while G will attempt to minimize this quantity by generating samples which fool the discriminator.

2.2.2.2 DCGAN

When Goodfellow et al. (2014) released the original GAN architecture, only dense feedforward neural networks were used for the discriminator and generator models. The key innovation in the DCGAN architecture was to use convolutional layers in both the discriminator and generator models (Radford et al. 2015). To generate the final image dimensions, fractionally-strided convolutions were used in the generator. A fractionally-strided convolution is performed by first inserting zero padding between the pixels in the image and then performing the convolution operation. The purpose of using a fractionally-strided convolution is to increase the output image size relative to the input image size. As the loss function for the DCGAN is the same as the original GAN, the training procedure is similar. The original architecture for the DCGAN generator can be seen in Figure 2-5. Starting from the left in Figure 2-5, we see that a latent vector with 100 dimensions is processed by a dense layer and then reshaped to form an image of size 4x4x1024. Afterwards, 4 fractionally-strided convolutional layers are used to gradually increase the image size up to the final output size of 64x64x3.

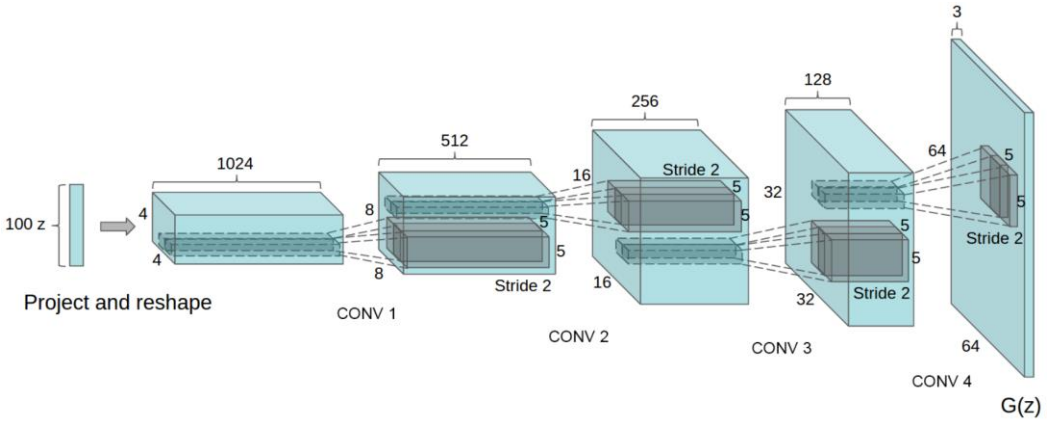

Figure 2-5 Original architecture for DCGAN generator network (Radford et al. 2015).

2.2.2.3 PGGAN

The key innovation in the PGGAN architecture is that the generator and discriminator networks are trained starting from low image resolution and progressively growing to the final large image resolution (Karras et al. 2017). This process is demonstrated in Figure 2-6. As the networks transition to train at higher resolution, linear interpolation between layer outputs is used to smooth the transition. Each time the PGGAN transitions to a higher resolution, the width and height of the images are doubled. Therefore, an important consideration when training a PGGAN is that width and height of the training images must be a power of 2. The PGGAN loss function is largely based off the improved Wasserstein metric (Arjovsky et al. 2017). Intuitively, the Wasserstein distance can be thought of as the minimum cost of transporting mass in the generator image distribution to form the dataset distribution. In addition to the Wasserstein metric, the PGGAN also utilizes labels on the dataset to incorporate an auxiliary classifier in the discriminator (Odena et al. 2016). In addition to predicting whether the data is real or fake, the discriminator is trained to maximize the

log likelihood of the correct class label. The auxiliary classifier loss encourages the discriminator to learn the image features which distinguish different classes.

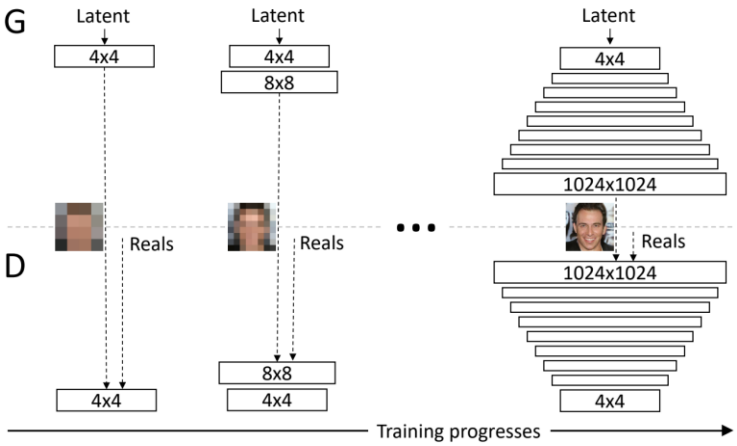

Figure 2-6 Visualization of the PGGAN training procedure, progressively growing the GAN from low resolution up to the final image resolution of 1024x1024 (Karras et al. 2017). On the upper half of the figure we see the generator network taking in a latent vector and producing an image output. On the lower half of the figure we see the discriminator processing the generated images together with raw images from the dataset (denoted in the figure as Reals).

2.2.2.4 Evaluation

After a GAN has finished training, it is important to have some metric which can measure the quality and diversity of the samples produced. Two commonly used metrics for this purpose are the Inception Score (IS) and the Fréchet Inception Distance (FID). Both the Inception Score and the FID will be used to evaluate the quality and diversity of GANs trained for this thesis.

The IS is based on using the output of the Inception classification network that has been pretrained on the ImageNet dataset. If the GAN produces good quality samples, then the conditional distribution $p(y|\mathbf{x})$ of the output label y given a generated sample $\mathbf{x}$ would be expected to have high predictability and hence low entropy. On the other hand, if the generated distribution is

diverse, then the marginal distribution $p(y) = \mathbb{E}_{\mathbf{x}\sim p_{generator}(\mathbf{x})}\left[p(y|\mathbf{x})\right]$ should have high entropy. Using these intuitions, the IS forms the scoring metric based on the KL divergence between the conditional distribution $p(y|\mathbf{x})$ and the marginal distribution $p(y)$ as follows

$$\mathrm{IS}\left(p_{generator}(\mathbf{x})\right) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_{generator}(\mathbf{x})}\left[D_{KL}\left(p(y|\mathbf{x})\|p(y)\right)\right]\right) \tag{2.5}$$

A larger IS indicates a better generator distribution. One of the limitations of the IS metric is that it does not compare the generated images against the real images from the dataset. Therefore, the IS metric does not provide any information regarding how well the generator distribution matches the dataset distribution.

The FID metric improves upon the IS metric by comparing the distribution of Inception layer activations for both the synthetic generated data and the raw dataset. A multivariate Gaussian is used to model the distribution for the Inception layer activations. The FID is calculated by the following expression

$$\begin{aligned}&\mathrm{FID}\left(p_{dataset}(\mathbf{x}), p_{generator}(\mathbf{x})\right)\\ &= \left\|\mu_{dataset} - \mu_{generator}\right\|_2^2 + \mathrm{Tr}\left(\Sigma_{dataset} + \Sigma_{generator} - 2\left(\Sigma_{dataset}\Sigma_{generator}\right)^{\frac{1}{2}}\right)\end{aligned} \tag{2.6}$$

Where $\mu_{generator}$ and $\mu_{dataset}$ denote the mean vectors of the Gaussian model for the generator and dataset Inception layer activations respectively, while $\Sigma_{generator}$ and $\Sigma_{dataset}$ denote the covariance matrices for the generator and dataset Inception layer activations respectively. Smaller FID values indicate better quality and diversity of the generator distribution.

2.2.3 Uncertainty Analysis

Suppose that we have a dataset consisting of input data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ and corresponding outputs $\mathbf{Y} = \{y_1, y_2, \cdots, y_N\}$ and we have trained a neural network classifier to estimate the discriminative distribution $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ such that we can make inferences on this distribution to find the optimal label to assign to a new data point $\mathbf{x}$. A pertinent question is how we can assess the uncertainty that the classifier has about the estimates it makes so that we can assign a confidence level to the assigned classification label. One metric commonly used to measure the amount of uncertainty in a probability distribution is entropy. Entropy measures the average information content in a distribution and is defined as:

$$\mathrm{H}(X) = \mathbb{E}\left[-\log\left(\mathrm{P}(X)\right)\right] \tag{2.7}$$

Where $\mathrm{P}(X)$ is the probability mass function. A possible approach to measure the uncertainty that a classifier has about a new data point $\mathbf{x}$ is to calculate the entropy of the discriminative distribution:

$$\mathrm{H}(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) = \mathbb{E}\left[-\log\left(P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})\right)\right] \tag{2.8}$$

Where $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ is the predictive probability mass function of the neural network conditioned on the new data point $\mathbf{x}$ and the dataset used to train the network $\mathbf{X}, \mathbf{Y}$. To compute the predictive probability mass function $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$, the classification network is first trained using $\mathbf{X}, \mathbf{Y}$ and then the new data point $\mathbf{x}$ is passed through the network as input. The resulting output of the neural network is a vector describing $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$. $\mathrm{H}(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ can be calculated by computing the entropy of this output vector. Unfortunately, Gal (2016) demonstrated that simply

using a point estimate of the classifier output distribution is not sufficient to properly model the uncertainty. This is due to the softmax activation function exaggerating the prediction confidence. Hence, while the output probabilities of the softmax activation are good for prediction, they must be calibrated to represent the true probabilities (Guo et. al 2017). A solution to this problem is to form a better estimate of the uncertainty by placing prior distributions on the parameters of the network such that a larger number of samples can be used to estimate the uncertainty. We can write the predictive distribution as a function of the neural network parameters $\boldsymbol{\omega}$ such that $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) = \mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x})$. Now using Bayes' Rule, we can expand this distribution as follows:

$$\begin{aligned} & P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) \\ &= \int_{\boldsymbol{\omega}} P(y, \boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})\, d\boldsymbol{\omega} \\ &= \int_{\boldsymbol{\omega}} P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}, \boldsymbol{\omega})\, P(\boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})\, d\boldsymbol{\omega} \end{aligned} \tag{2.9}$$

The first step follows from expanding the marginal distribution in terms of the joint distribution, and the second step follows directly from the definition of conditional probability. In the final expression, we have two terms in the integral. Let us examine what these terms represent.

The first term $P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}, \boldsymbol{\omega})$ is the output of the neural network and describes the probability of each classification label. The second term $P(\boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ describes the distribution over all possible network parameters given the training data. As the network parameters $\boldsymbol{\omega}$ are initialized randomly and the neural network is trained using stochastic gradient descent, the final parameters values of the trained network will also vary stochastically, and this is described by the distribution $P(\boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$. As a large deep neural network can easily have millions of parameters, calculating $P(\boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ directly is computationally intractable. To mitigate this problem, we can use the methods of variational inference to develop a distribution which closely approximates

$P(\boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$. Kendall et al. (2017) and Gal et al. (2015) show how dropout can be used as technique of sampling from a distribution which approximates $P(\boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$ by assuming Bernoulli prior distributions for the weights. Dropout is a technique which was proposed originally to regularize a neural network for prevention of overfitting (Srivastava et al. 2014). The basic premise is that during training, a Bernoulli random variable is sampled for each network parameter where dropout is used. This sampled value acts as a multiplicative mask for the parameter. In other words, when the sampled value is 1, the parameter keeps its value, otherwise the parameter is dropped for the training iteration (assigned a value of 0). The motivation behind this technique is to stochastically create subnetworks within the larger network, such that the network must learn redundancy which combats overfitting. For Bayesian uncertainty analysis, when we use dropout, we can consider each parameter to be sampled from a scaled Bernoulli distribution. Kendall et al. (2017) show how by using this formulation, we can develop a Monte Carlo method using dropout for sampling from the desired distribution $P(\boldsymbol{\omega} \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})$. The final calculation is given as follows

$$P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y}) \approx \frac{1}{N} \sum_{n=1}^{N} P(y \mid \hat{\boldsymbol{\omega}}_n, \mathbf{x}, \mathbf{X}, \mathbf{Y}) \tag{2.10}$$

Where $\hat{\boldsymbol{\omega}}_n$ are the parameters of the network sampled in the nth Monte Carlo dropout sample. We will refer to this sampling technique as MC dropout. We can use this approximation of the predictive distribution to better estimate the network uncertainty. To rank samples by their uncertainty we use a scoring metric called an acquisition function. The Bayesian Active Learning by Disagreement (BALD) acquisition function proposed by Houlsby et al. (2011) is defined as follows

$$U(\mathbf{x}) = H\left[P(y \mid \mathbf{x}, \mathbf{X}, \mathbf{Y})\right] - \mathbb{E}_{P(\boldsymbol{\omega} \mid \mathbf{X}, \mathbf{Y})}\left[H\left[P(y \mid \mathbf{x}, \boldsymbol{\omega})\right]\right] \tag{2.11}$$

Computationally, this can be approximated using the MC dropout samples as

$$U(\mathbf{x}) \approx H\left[\frac{1}{N}\sum_{n=1}^{N} P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right] - \frac{1}{N}\sum_{n=1}^{N} H\left[P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right] \tag{2.12}$$

where N is the number of MC samples, and $\boldsymbol{\omega}_n$ are the parameters of the network sampled for the nth MC dropout sample. Data points with high entropy for the average predictive distribution of the MC dropout samples, but low average entropy for the entropy of each of the sampled predictive distribution will have a high BALD score indicating that the network is uncertainty about the prediction. The intuition behind this metric is that if the dropout sampling of the weights causes the network to change its prediction, then the network is considered uncertain about the sample prediction.

2.3 Summary

The purpose of this chapter was to present a background summary regarding prior related work and machine learning techniques relevant for this thesis. A thorough description of image classification, generative networks, and uncertainty analysis was provided. Traditional GAN augmentation methods do not take the classification uncertainty into account when sampling the synthetic images, limiting the resulting classification performance (Wang et al. 2017). The key innovation in this thesis was to combat this limitation through the development of an image classification system that is capable of augmenting the training set using samples selected from a GAN based on analysis of the classification network uncertainty. The following chapter will describe the overall design for this system.

Chapter Three: System Design

The purpose of this chapter is to present the overall system design for the classification framework developed in this thesis. Section 3.1 presents a high-level overview of the system design. Section 3.2 describes the operations performed for dataset preprocessing. Section 3.3 defines the architecture of each GAN, the training procedure, and the metrics used for performance evaluation. Section 3.4 presents the classifier architecture and describes the processing for each iteration of the training feedback loop. Section 3.5 summarizes the details presented in this chapter.

3.1 System Overview

The overall structure for the classification framework developed in this thesis is shown in Figure 3-1 and will be described in the subsequent sections. The high-level operation of each component is as follows:

- **Dataset**: For the purposes of this thesis, all raw data points are acquired from the MNIST, LSUN, and ISIC 2018 datasets. Each raw data point is an image with a corresponding class label. To improve the stability of the algorithms, preprocessing is applied to each of the images before training (Tabik et al. 2017). Further preprocessing details are provided in Section 3.2.
- **GAN**: The GAN is trained on the raw data in the dataset and consists of two networks, the generator and the discriminator. The generator network is used to synthesize image samples for data augmentation. The discriminator network is used by the importance sampling mechanism to determine which samples from the GAN have a high probability of being

realistic. The architecture design for each GAN and a description of the training procedure and metrics used for performance evaluation is provided in Section 3.3.

- **Importance Sampling**: The importance sampling mechanism developed in this thesis determines which data samples are to be acquired for the next iteration of the training loop. The samples could come from the GAN or from the raw dataset. Details of the importance sampling mechanism are given in Section 3.4.2.
- **Training Loop**: During each iteration of the training loop, samples selected by the importance sampling mechanism are used to train the image classifier. After the training iteration has completed, the trained classifier is then used by the importance sampling mechanism to pick the best samples for the next iteration. This process continues until the final dataset size is reached, upon which the resulting network is outputted as the *Final Classification Network*. Further details regarding the training loop operation are provided in Section 3.4.2.

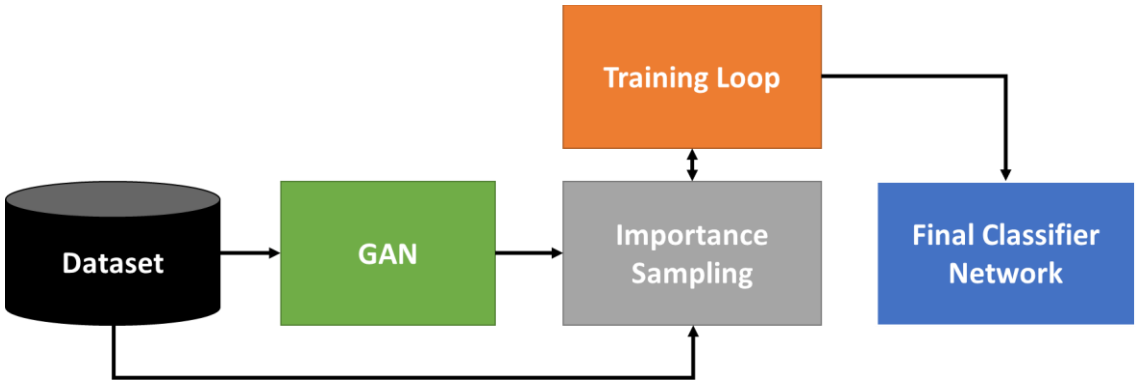

Figure 3-1 Overall design for the classification framework developed in this thesis. The direction of each line indicates the flow of data.

3.2 Dataset Preprocessing

A description of how each dataset was preprocessed is given below:

- **MNIST**: Each image in the original MNIST dataset has a resolution of 28x28 pixels. The preprocessed MNIST dataset for this thesis was formed by resizing the images to a resolution of 32x32 pixels using cubic interpolation. The motivation for resizing each image to 32x32 pixels was to enforce that the width and height of the images are powers of 2 which is required for training a PGGAN. The images were then normalized such that each pixel had a value range between -1 and 1. The resulting dataset consisted of 60,000 training images with ~6,000 images per category and 10,000 test images with ~1,000 per category. Example images from the MNIST dataset after preprocessing are shown in Figure 3-2.
- **LSUN**: Each image in the original LSUN dataset has a resolution where the image short edge had 256 pixels (Yu et al. 2015). The preprocessed LSUN dataset for this thesis was formed by first randomly sampling 10,000 images from each category to form the training set. Then each image was cropped such that the resulting resolution was 256x256 pixels. The motivation for resizing each image to 256x256 pixels was to enforce that the width and height of the images are powers of 2 which is required for training a PGGAN. For GAN training the images were normalized such that each pixel had a value range between -1 and 1. For classifier training the preprocessing method proposed by Howard et al. (2017) was used. The resulting dataset consisted of 100,000 images, with 10,000 images from each category and 3,000 test images, comprising 300 images from each category. Example images from the LSUN dataset after preprocessing are shown in Figure 3-3.

- **ISIC 2018**: Each of the images in the original ISIC 2018 dataset has a resolution of 600x450 pixels. The preprocessed ISIC 2018 dataset developed for this thesis was formed by first randomly sampling 500 images from the 10,015 training images to form the test set. All images were scaled to a resolution of 256x256 pixels using cubic interpolation. The motivation for resizing each image to 256x256 pixels was to enforce that the width and height of the images are powers of 2 which is required for training a PGGAN. As the images were not cropped before scaling, the aspect ratio of the images was not preserved. The motivation behind this preprocessing decision was that the skin lesions in the images were not centered, therefore cropping the image might have removed important details relating to the classification of the lesion. For GAN training the images were normalized such that each pixel had a value range between -1 and 1. For classifier training the preprocessing method proposed by Howard et al. (2017) was used. The resulting dataset consisted of 9,515 training images and 500 test images. The label distribution for the ISIC 2018 dataset is highly nonuniform as shown in Figure 3-5. Example images from the ISIC 2018 dataset after preprocessing are shown in Figure 3-4.

Figure 3-2 Example images from the preprocessed MNIST dataset.

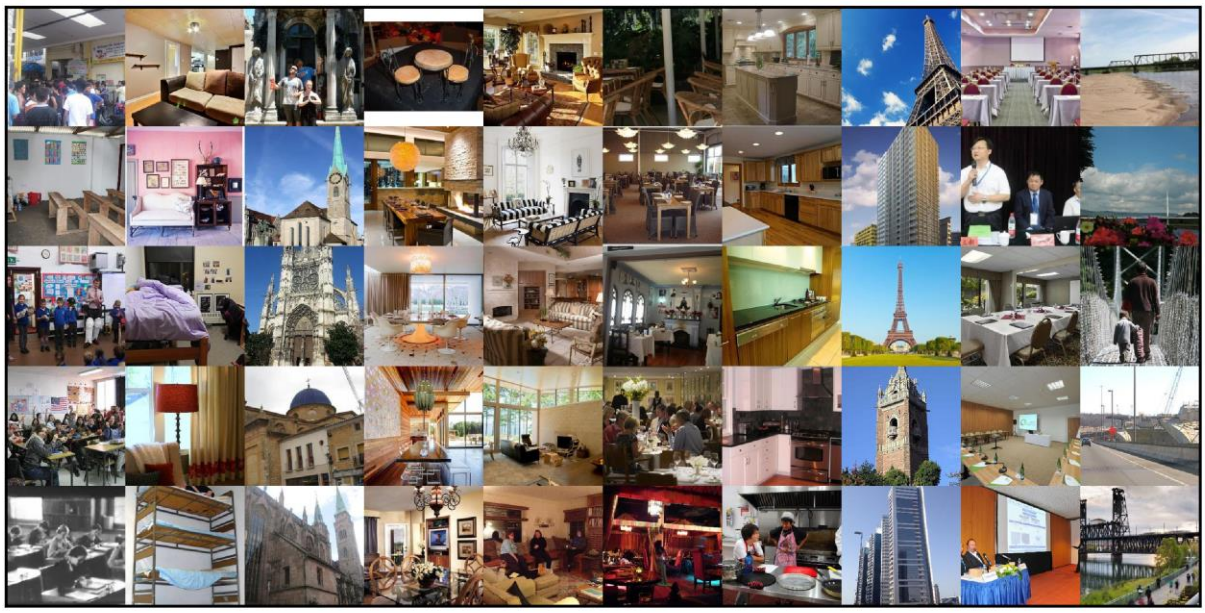

Figure 3-3 Example images from the preprocessed LSUN dataset.

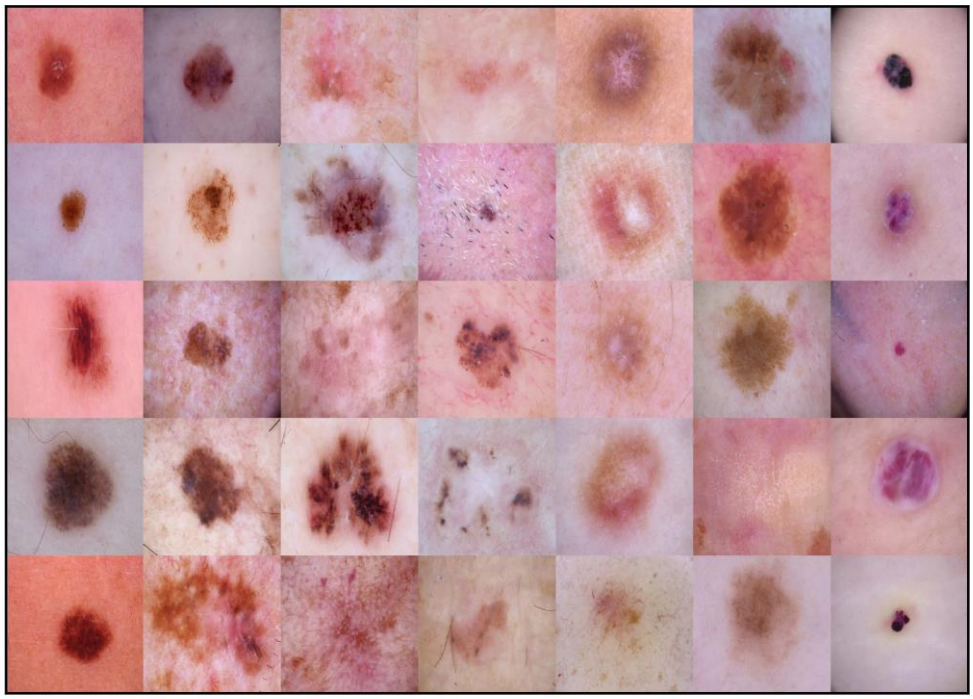

Figure 3-4 Example images from the preprocessed ISIC 2018 dataset.

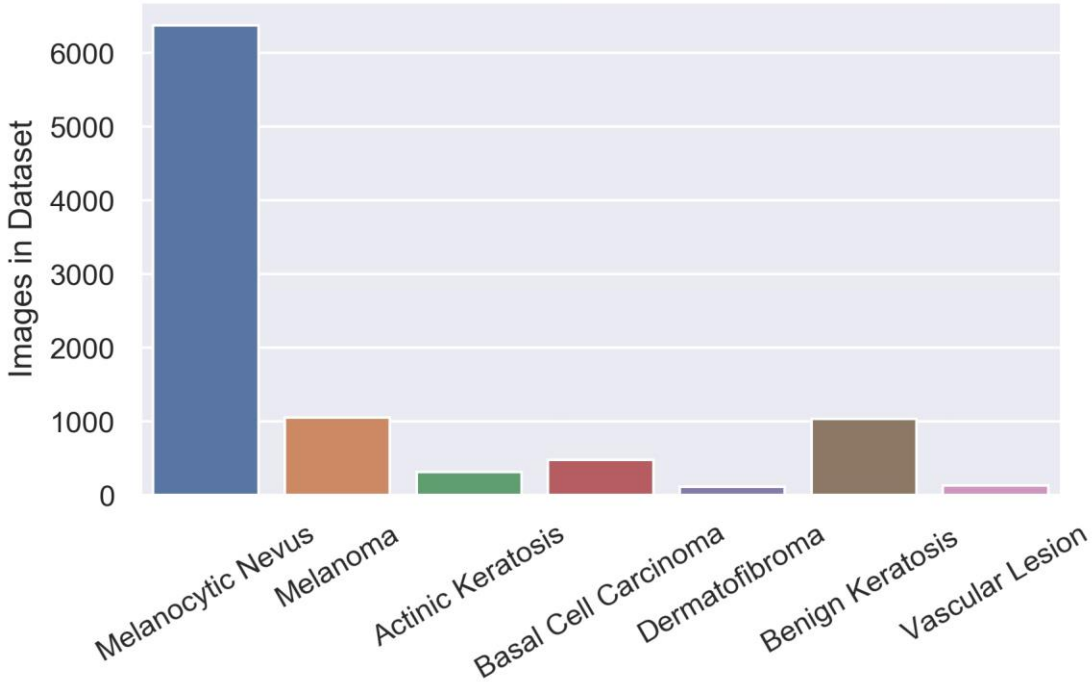

Figure 3-5 Distribution of class labels for the ISIC 2018 dataset.

3.3 GAN Training

3.3.1 GAN Architectures

There are three GAN architectures used for this thesis: Small-DCGAN, Large-DCGAN, and PGGAN. The Small-DCGAN and Large-DCGAN architectures are adaptations of the original DCGAN model (Radford et al. 2015) with small and large capacities respectively. Due to the lower capacity relative to the PGGAN, the Small-DCGAN and Large-DCGAN were only trained using the MNIST dataset. Let us describe the terminology used in the following tables:

- **Layer Number:** As the architectures for the networks developed in this thesis are feedforward, all layers are processed sequentially and the number referencing the given layer describes the processing order.
- **Operation:** There are different layer types used for the construction of the network architectures. *Linear* refers to a densely connected layer. *Convolution* refers to a convolutional layer. *Upsample + Convolution* consists of an upsampling operation followed by a convolutional layer. The upsampling operation uses interpolation to scale the width and height of the image by 2. The *Max Pool* layer performs downsampling on the activation outputs by only outputting the maximum value activation for a small neighborhood of surrounding pixels.
- **Kernel**: The width and height of the convolution kernel.
- **Strides**: For convolution layers, *Strides* refers to the step size taken by the convolution operation. For max pool layers, *Strides* refers to the neighborhood size which is pooled.
- **Feature Maps:** For densely connected layers, *Feature Maps* describes the number of neurons in the layer. For convolutional layers, *Feature Maps* describes the number of kernels in the layer.

- **Dropout:** The probability of dropping each output activation in the layer.
- **Use Batch Normalization:** Whether batch normalization was used after the layer activation.
- **Activation Function:** The type of activation function used by the layer.

The network architecture for the Small-DCGAN generator and discriminator are seen in Table 3-1 and Table 3-2 respectively. The total number of trainable parameters for the Small-DCGAN is 369,762 where the generator and discriminator each respectively contain 270,113 and 99,649 trainable parameters. The training parameters used for the Small-DCGAN are shown in Table 3-3.

Table 3-1 Generator layer architecture for Small-DCGAN
(100-dimensional latent space with 270,113 total trainable parameters)

Layer Number	Operation	Kernel	Strides	Feature Maps	Dropout	Use Batch Normalization	Activation Function
L1	Linear	N/A	N/A	1568	0.0	Yes	ReLU
L2	Upsample + Convolution	3x3	1x1	128	0.0	Yes	ReLU
L3	Upsample + Convolution	3x3	1x1	64	0.0	Yes	ReLU
L4	Convolution	3x3	1x1	1	0.0	No	Tanh

Table 3-2 Discriminator architecture for Small-DCGAN (99,649 total trainable parameters)

Layer Number	Operation	Kernel	Strides	Feature Maps	Dropout	Use Batch Normalization	Activation Function
L1	Convolution	3x3	2x2	16	0.25	No	Leaky ReLU
L2	Convolution	3x3	2x2	32	0.25	Yes	Leaky ReLU
L3	Convolution	3x3	2x2	64	0.25	Yes	Leaky ReLU
L4	Convolution	3x3	1x1	128	0.25	No	Leaky ReLU
L5	Linear from L4	N/A	N/A	1	0.0	No	Sigmoid

Table 3-3 Training hyperparameters for Small-DCGAN and Large-DCGAN

Training Parameter	Value
Generator Optimizer	Adam
Discriminator Optimizer	Adam
Batch Size	32
Iterations	50000
Leaky ReLU Slope	Alpha = 0.2
Weight, Bias Initialization	Xavier Uniform Initializer

The network architecture for the Large-DCGAN generator and discriminator are seen in Table 3-4 and Table 3-5 respectively. The structure of the Large-DCGAN has more layers than the Small-DCGAN and the total number of trainable parameters for the Large -DCGAN is 1,627,682 where the generator and discriminator each respectively contain 1,040,705 and 586,977 trainable parameters. The training parameters used for the Large-DCGAN are identical to those used for the Small-DCGAN and are shown in Table 3-3.

Table 3-4 Generator layer architecture for Large-DCGAN
(100-dimensional latent space with 1,040,705 total trainable parameters)

Layer Number	Operation	Kernel	Strides	Feature Maps	Dropout	Use Batch Normalization	Activation Function
L1	Linear	N/A	N/A	6272	0.0	Yes	ReLU
L2	Convolution	3x3	1x1	128	0.0	Yes	ReLU
L3	Upsample + Convolution	3x3	1x1	128	0.0	Yes	ReLU
L4	Convolution	3x3	1x1	64	0.0	Yes	ReLU
L5	Upsample + Convolution	3x3	1x1	64	0.0	Yes	ReLU
L6	Convolution	3x3	1x1	1	0.0	No	Tanh

Table 3-5 Discriminator architecture for Large-DCGAN (586,977 total trainable parameters)

Layer Number	Operation	Kernel	Strides	Feature Maps	Dropout	Use Batch Normalization	Activation Function
L1	Convolution	3x3	1x1	32	0.25	No	Leaky ReLU
L2	Convolution	3x3	2x2	32	0.25	No	Leaky ReLU
L3	Convolution	3x3	1x1	64	0.25	Yes	Leaky ReLU
L4	Convolution	3x3	2x2	64	0.25	Yes	Leaky ReLU
L5	Convolution	3x3	1x1	128	0.25	Yes	Leaky ReLU
L6	Convolution	3x3	2x2	128	0.25	Yes	Leaky ReLU
L7	Convolution	3x3	1x1	256	0.25	Yes	Leaky ReLU
L8	Linear	N/A	N/A	1	0.0	No	Sigmoid

The PGGAN model used for this thesis was based off the implementation provided by Karras et al. (2017). The PGGAN architecture is significantly larger than the Small-DCGAN and Large-DCGAN with over 20 million parameters used in each of the generator and discriminator networks. One key advantage of the PGGAN architecture is the ability to progressively scale the model to

train on images of various sizes. The ISIC 2018 and LSUN dataset images were each 256x256 pixels and the MNIST dataset images were 32x32 pixels, yet the PGGAN could train on both resolutions due to the progressively growing nature of the model.

3.3.2 Training and Performance Evaluation

During training, multiple iterations of gradient descent were taken to decrease the loss function of the generator and discriminator. The details for each of the trained GAN networks are shown in Table 3-6. Label conditioning refers to the GAN being trained to model the conditional distribution of the data for a given label. This is beneficial as it allows the entire dataset to be represented using a single GAN. However, it was experimentally determined that the Small-DCGAN and Large-DCGAN suffered from mode collapse when modelling the conditional distribution. Mode collapse is a common issue when training GANs where the generator network collapses onto a single mode of the data distribution (Che et al. 2016). To compensate for this problem, an ensemble of GANs was used, each trained to generate images representing a specific label from the dataset.

The GANs used to generate the LSUN samples were pretrained by Karras et al. (2017). All other models were trained on a local machine that contained two Nvidia 1080 Ti GPUs. The time required to train the GANs varied with the capacity of the architectures. The simplest model (MNIST Small-DCGAN) took 30 minutes to train, while the most complex model (ISIC 2018 PGGAN) took 10 days to train.

Table 3-6 Overview of trained GAN architectures

Dataset Type	GAN Architecture	Use Label Conditioning	Training Location	Training Time
MNIST	Small-DCGAN	No	Local Machine	30 Minutes
MNIST	Large-DCGAN	No	Local Machine	1 Hour
MNIST	PGGAN	Yes	Local Machine	1 Day
LSUN	PGGAN	No	Pretrained (Karras et al. 2017)	N/A
ISIC 2018	PGGAN	Yes	Local Machine	~10 Days

To evaluate the performance of the trained GANs, the IS and FID scores were computed for each GAN. The IS is computed using samples from the GAN, while the FID is computed using samples from both the GAN and the dataset. 5000 images were sampled from each class label for the metric computation.

3.4 Classification Training Loop:

3.4.1 Classifier Architecture

The classifier architecture used for the MNIST dataset is described in Table 3-7. The developed CNN has 5 layers and has 1,199,882 trainable parameters. The specifications of the training environment are described in Table 3-8.

Table 3-7 Classifier Architecture for MNIST (1,199,882 total trainable parameters)

Layer Number	Operation	Kernel	Strides	Feature Maps	Dropout	Use Batch Normalization	Activation Function
L1	Convolution	3x3	1x1	32	0.0	No	ReLU
L2	Convolution	3x3	1x1	64	0.0	No	ReLU
L3	Max Pool	N/A	2x2	N/A	0.25	No	None
L4	Linear from L3	N/A	N/A	128	0.5	No	ReLU
L5	Linear from L4	N/A	N/A	10	0.0	No	Softmax

Table 3-8 Training hyperparameters for MNIST Classifier

Training Parameter	Value
Optimizer	Adadelta
Batch Size	32
Epochs	100
Weight, Bias Initialization	Xavier Uniform Initializer
Loss Function	Cross-Entropy

As the images in the LSUN and ISIC 2018 datasets are much higher resolution than the MNIST data, a classification network with greater capacity must be used. Training high resolution classification models from scratch typically requires a large amount of data. To mitigate this problem, we used a classification network that has been pretrained on the ImageNet dataset. The pretrained network used for the transfer learning is based on the MobileNet architecture developed by Howard et al. (2017). The top layers of the network were stripped off and replaced by a 128-dimension dense layer with dropout of 0.5. A final dense layer was placed on the network with the

dimensions equal to the number of classes in the dataset (7 for ISIC 2018 and 10 for LSUN). Furthermore, a weighted loss function was used to compensate for the imbalance in the ISIC 2018 dataset. The hyperparameters used for training the classification network are shown in Table 3-9.

Table 3-9 Training hyperparameters for LSUN and ISIC 2018 Classifier

Training Parameter	**Value**
Optimizer	SGD
Batch Size	32
Epochs	30
Weight, Bias Initialization	Xavier Uniform Initializer
Loss Function	Cross-Entropy

3.4.2 Training Feedback Loop

The algorithm for the processing performed during each iteration of the training loop is shown in Figure 3-6. We start iteration step N in possession of the current trained classifier network and the current training set. To perform an iteration of the training loop, samples from the data source are used to compute the classifier network posterior estimates through MC dropout. Next, an acquisition function is used to process the posterior estimates and assign each image sample a score. The samples with the highest scores are added to the training set for iteration step $N+1$ and used to train the resulting classifier for iteration step $N+1$. This process repeats until desired convergence is met or the predefined number of iterations are completed. For the base case when $N=0$, the classifier network is initialized with random parameter values. The data sources used for this thesis consist of images from the raw dataset, and images sampled from the Small-

DCGAN, Large-DCGAN, and PGGAN. When a GAN data source is used, the discriminator output is computed to filter the data such that only generated images with a discriminator output greater than one standard deviation above the mean output value will be considered for acquisition function scoring. This thresholding has the effect of filtering out images that the discriminator believes are not representative of the dataset. The acquisition functions used for this thesis are random sampling, BALD, and max entropy. Random sampling simply involves selecting random images from the data source to become part of the training set for the next iteration. BALD acquisition involves computing the following score for each of the assets in the data source

$$S_{BALD}(\mathbf{x}) = H\left[\frac{1}{N}\sum_{n=1}^{N} P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right] - \frac{1}{N}\sum_{n=1}^{N} H\left[P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right] \tag{3.1}$$

After the scores are computed, they are sorted and the images with the highest scores are sampled and added to the training set for the next iteration. Max entropy acquisition involves computing the following score for each of the assets in the data source

$$S_{Max\,Entropy}(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} H\left[P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right] \tag{3.2}$$

Similar to BALD acquisition, after the scores are computed, they are sorted and the images with the highest scores are sampled and added to the training set for the next iteration.

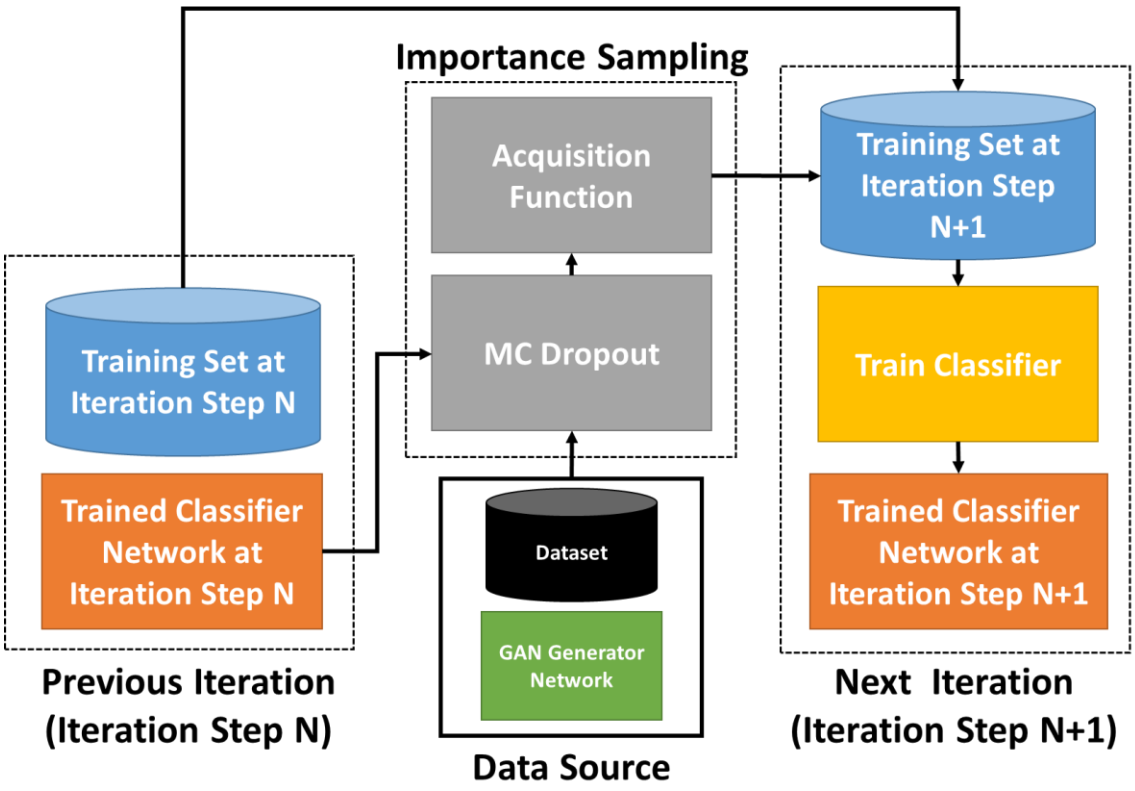

Figure 3-6 Overview of the classification training loop.

3.5 Summary

The purpose of this chapter was to describe the overall system structure for the classification training loop. A detailed description was provided regarding the preprocessing operations applied for each of the datasets. The neural network architecture and training hyperparameters were presented for each of the GAN and classifier models. Finally, the algorithmic processing operations performed at each iteration of the training loop were presented.

Chapter Four: Experimental Results

4.1 Introduction

The purpose of this chapter is to present the experimental results for the developed classification system. In Section 4.2, the outcome of the GAN training will be discussed. The output of the GAN loss functions will be analyzed, and a set of sample images generated by each GAN will be visually inspected for a qualitative assessment of image quality. Furthermore, the statistics of the output range for each discriminator will be presented, and the IS and FID metrics will be used to quantitatively measure the diversity and quality of images synthesized by each GAN. Section 4.3 will present a qualitative interpretation of the classifier network uncertainty by comparing the difference between images with low and high BALD scores. Section 4.4 will describe the experiments that were conducted for the classification framework using the MNIST, LSUN, and ISIC 2018 datasets. Section 4.5 addresses each of the research questions proposed in Section 1.2 through analysis of the experimental results. Finally, Section 4.6 will present a summary of the findings in this chapter.

4.2 GAN Training

4.2.1 Loss Function Analysis

During the training of the GAN architectures, the classification accuracy of the discriminator and the loss functions for the discriminator and generator networks were recorded. Figure 4-1 shows the discriminator training accuracy for the Small-DCGAN and Large-DCGAN models while being trained on the MNIST dataset. A key point to illustrate is that a larger prediction accuracy does not imply better GAN samples, it simply describes the relative strength of the discriminator to the

generator. Notice how the prediction accuracy seems to stabilize for the Small-DCGAN around 0.70. This is due to the balanced competition between the capacities of the generator and discriminator networks. If the discriminator network was much stronger than the generator, the prediction accuracy would be closer to 1, and the generator would be unable to compete with the discriminator, preventing the GAN from converging. Therefore, by observing the stability of the prediction accuracy, we are made aware that the capacity of the generator and discriminator networks are well matched. On the other hand, we notice that the prediction accuracy for the Large-DCGAN seems to be decreasing gradually over time. To explain this result, we must examine the loss functions for the generator and discriminator networks.

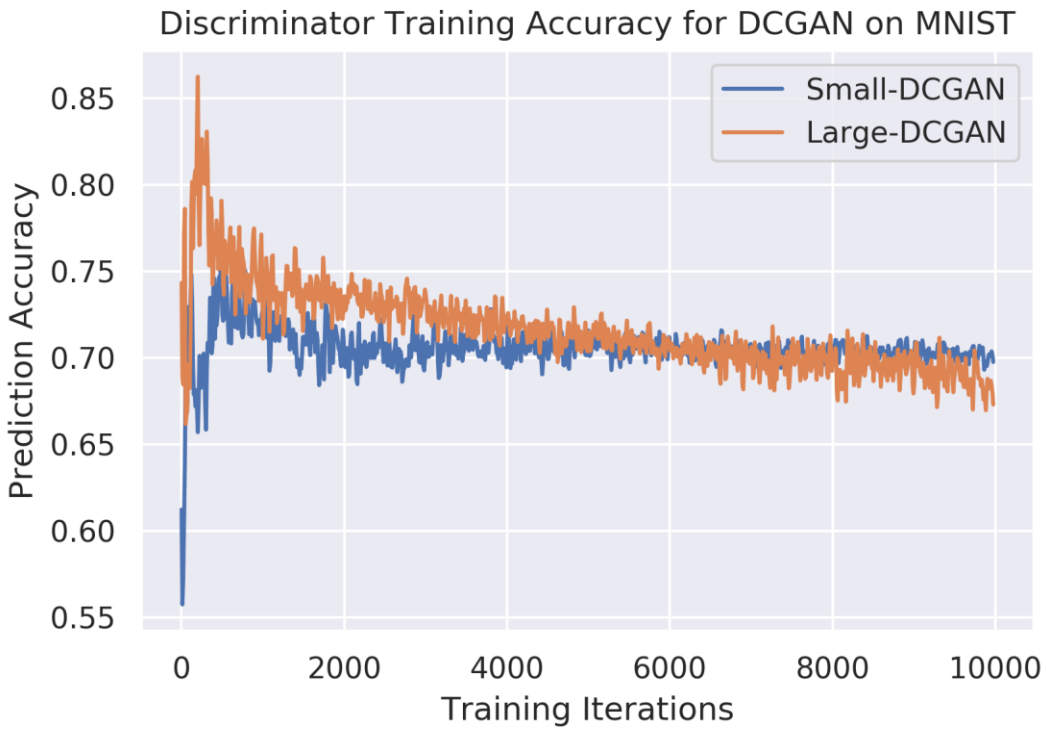

Figure 4-1 DCGAN discriminator training accuracy on the MNIST dataset.

Figure 4-2 shows the generator and discriminator loss functions for both the Small-DCGAN and Large-DCGAN models while being trained on the MNIST dataset. Examining the loss for the Large-DCGAN we see that the loss for both the discriminator and generator is increasing at the end of the training iterations. This result can be explained by looking back at the GAN optimization problem

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}\left[\log\left(1-D\left(G(\mathbf{z})\right)\right)\right] \tag{4.1}$$

If the generator and discriminator loss are simultaneously increasing then it follows that discriminator is getting better at discriminating images produced by the generator such that $\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}\left[\log\left(1-D\left(G(\mathbf{z})\right)\right)\right]$ is increasing, but worse at discriminator real images such that $\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x})\right]$ is decreasing. This explains why the classification accuracy in Figure 4-1 was decreasing over time.

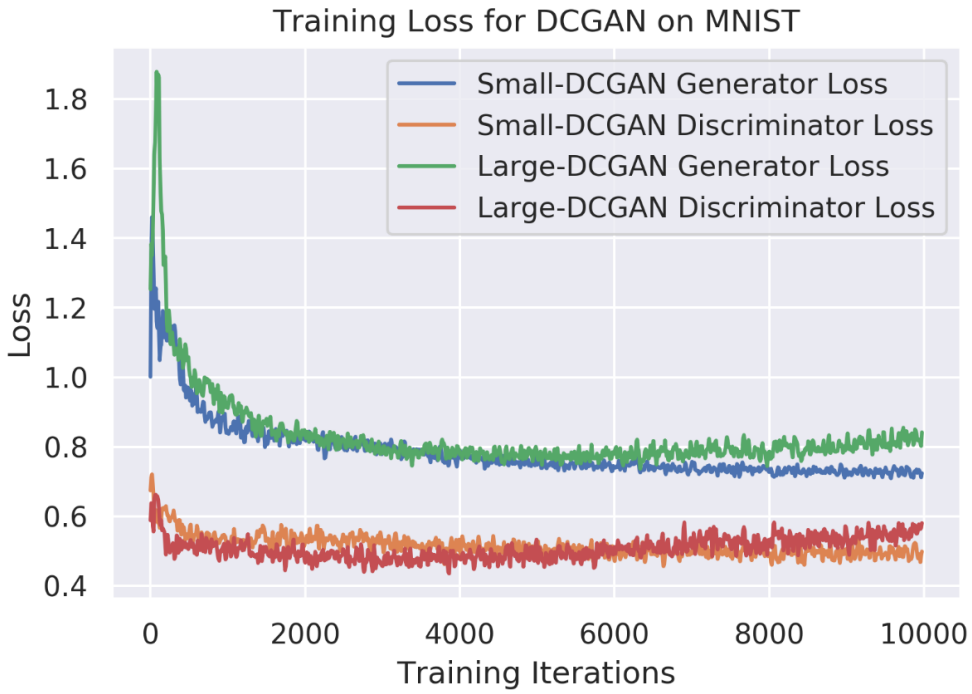

Figure 4-2 DCGAN training loss for discriminator and generator on the MNIST dataset.

4.2.2 Qualitative Visual Assessment of Samples

To qualitatively assess the generator quality of the trained GANs, images from each GAN architecture were sampled and visually inspected.

4.2.2.1 GAN Image Quality During Training

Figure 4-3 shows samples taken during the training of the MNIST PGGAN architecture. Each column displays the generated images for a fixed latent vector over the course of the training iterations. Notice how during the early iterations the images are quite pixelated, this is due to the progressive growing nature of the PGGAN.

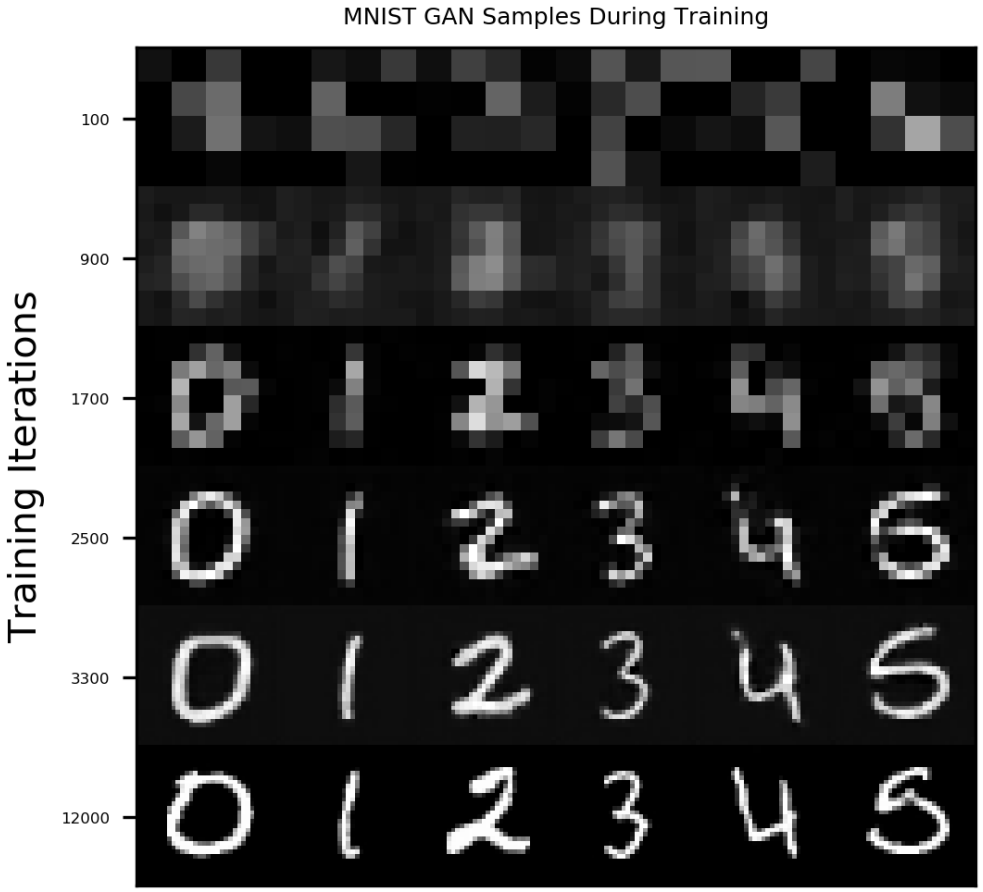

Figure 4-3 PGGAN samples during training on the MNIST dataset.

Figure 4-4 shows samples taken during the training of the ISIC 2018 PGGAN architecture. Each column displays the generated image for a fixed latent vector over the course of the training iterations. Notice how some columns exhibit a significant amount of variation between training iterations. The relative variation seems to be correlated with class imbalance between the training labels. Impressively, the GAN architecture is capable of learning to represent specialized details such as hair in the images.

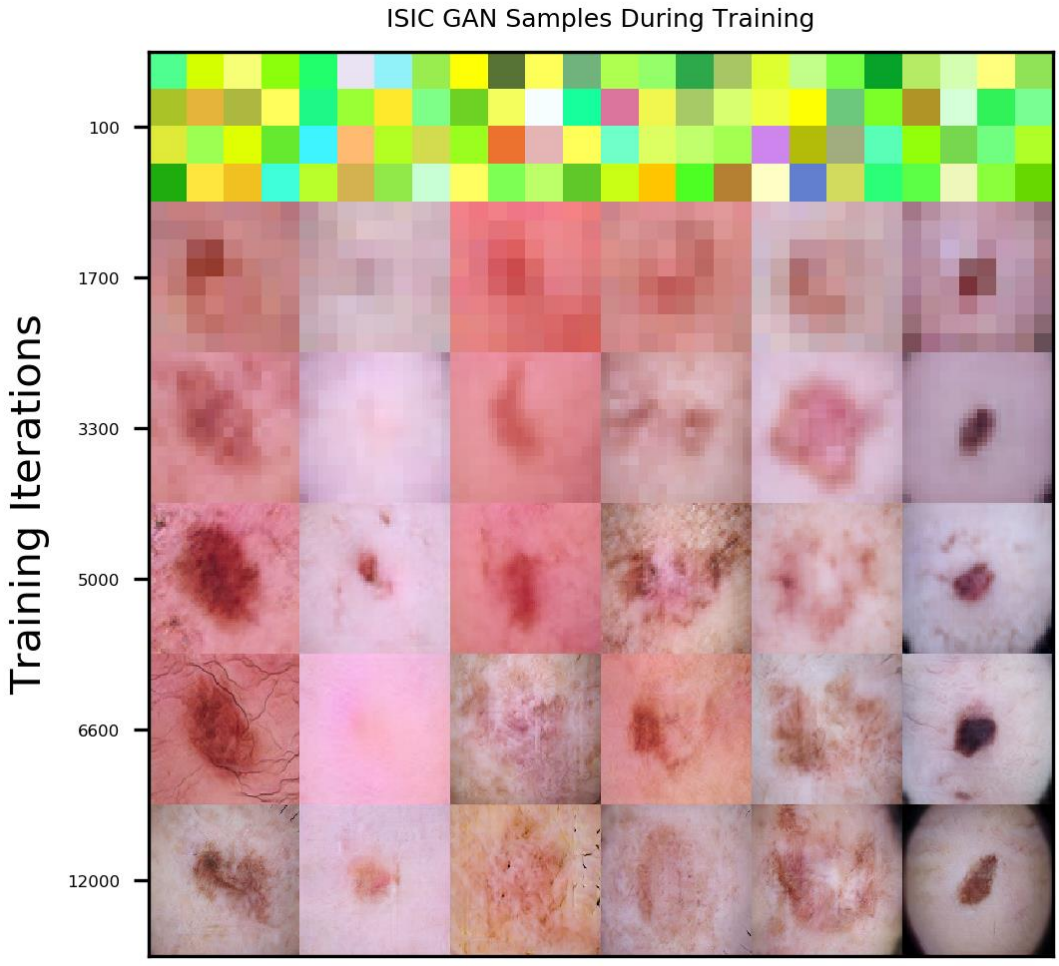

Figure 4-4 PGGAN samples during training on the ISIC 2018 dataset.

4.2.2.2 Final GAN Image Quality

Figure 4-5 shows samples generated from the trained PGGAN for each category of the LSUN dataset. The GAN generates near photorealistic images of environments composed of static objects such as chairs and beds. Specifically, the images from the *dining room* category exhibit a high degree of photorealism. Examining the images from the *classroom* category, we see that the faces generated by the GAN are distorted. This is due in part to the high variance in facial structure that is difficult for the GAN to capture, but there is also a psychological explanation of this observation. The visual cortex of a human observer has been evolutionarily trained to recognize human faces, making it is easier for the observer to detect facial distortion in GAN generated images than the distortion of inanimate objects such as couches (Tsao et al. 2008). Therefore, although the distortion of the human faces in the classroom category is perceived to be worse than the distortion of the desks, this perception is heavily biased by evolutionary preconditioning.

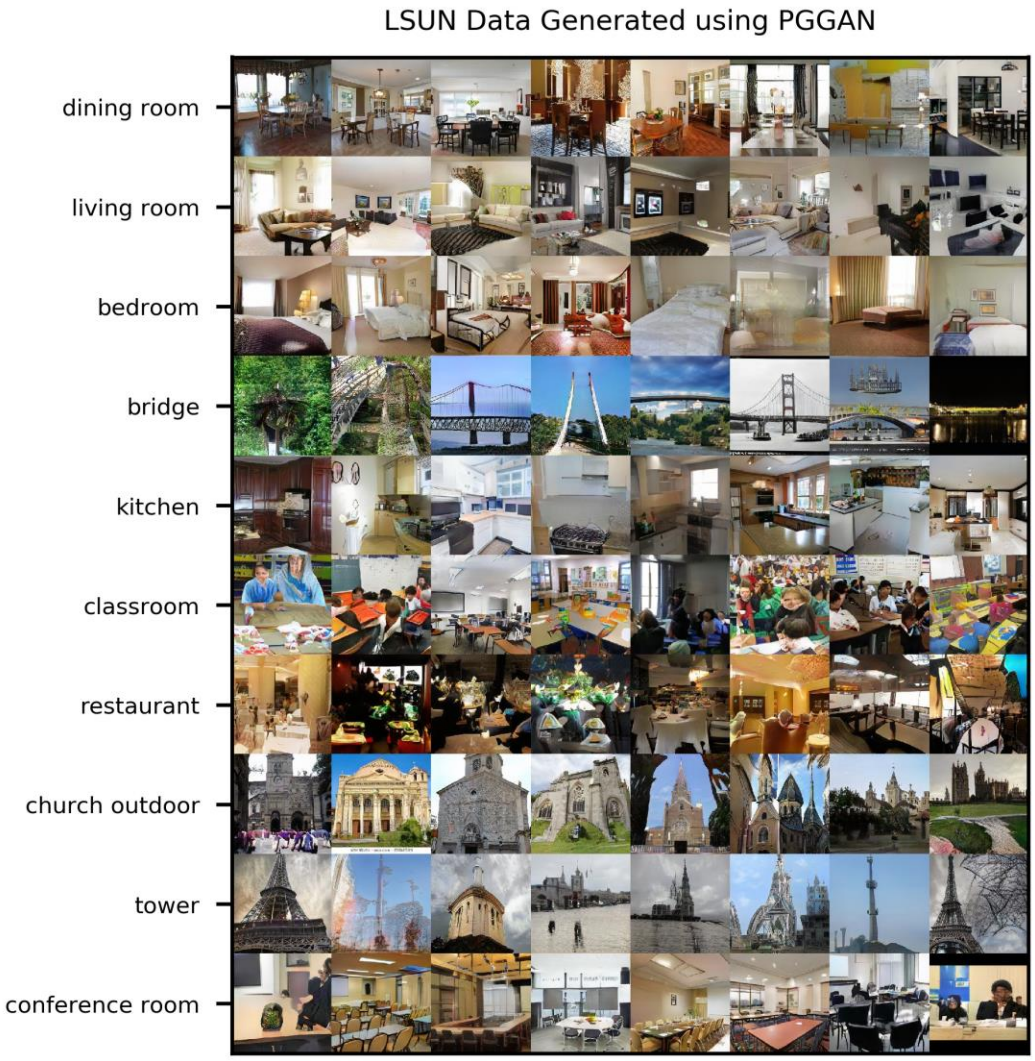

Figure 4-5 PGGAN samples from LSUN dataset categories.

Figure 4-6 shows samples generated from the trained PGGAN for each category of the ISIC 2018 dataset. Examining the generated images, we see that the GAN was capable of modelling specific details of the skin such as wrinkles and hair. We also notice that the GAN models the artifacts of the original image capturing device such as the black border around the perimeter of some of the generated images. To quantitatively describe the realism of these images it would be necessary to have them examined by a trained radiologist. However, by comparing the high-level characteristics

of the generated images to the samples from raw data set (shown in Figure 3-4) we can infer by visual inspection that the images are highly similar.

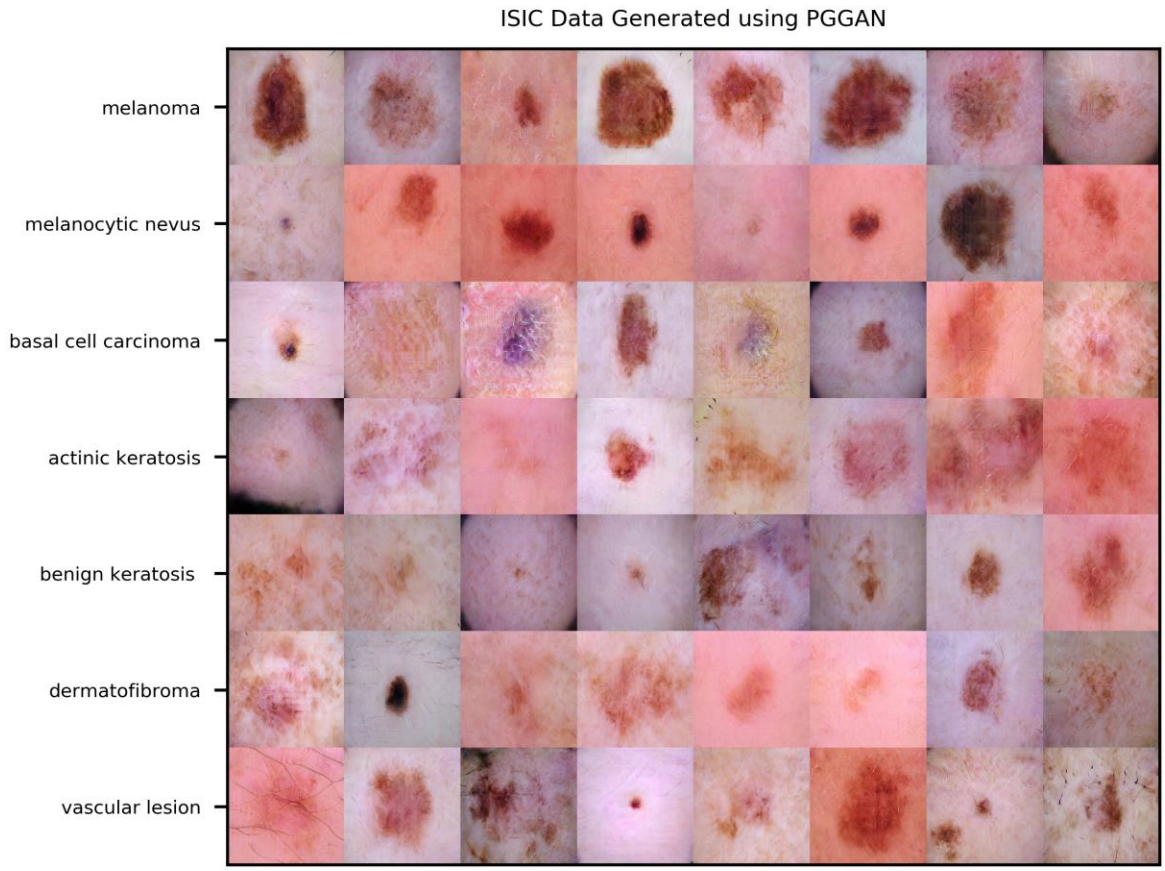

Figure 4-6 PGGAN samples from ISIC 2018 dataset categories.

Figure 4-7 shows samples generated from the GAN architectures trained on the MNIST dataset. By examining the images, it is clear that the sample quality increases as we move from the Small-DCGAN to the Large-DCGAN and finally to the PGGAN. This is understandable since the capacity of the Small-DCGAN is significantly smaller than that of the PGGAN. Notice how the GANs are able to generate digits with the same label, for example '2', but with unique styles.

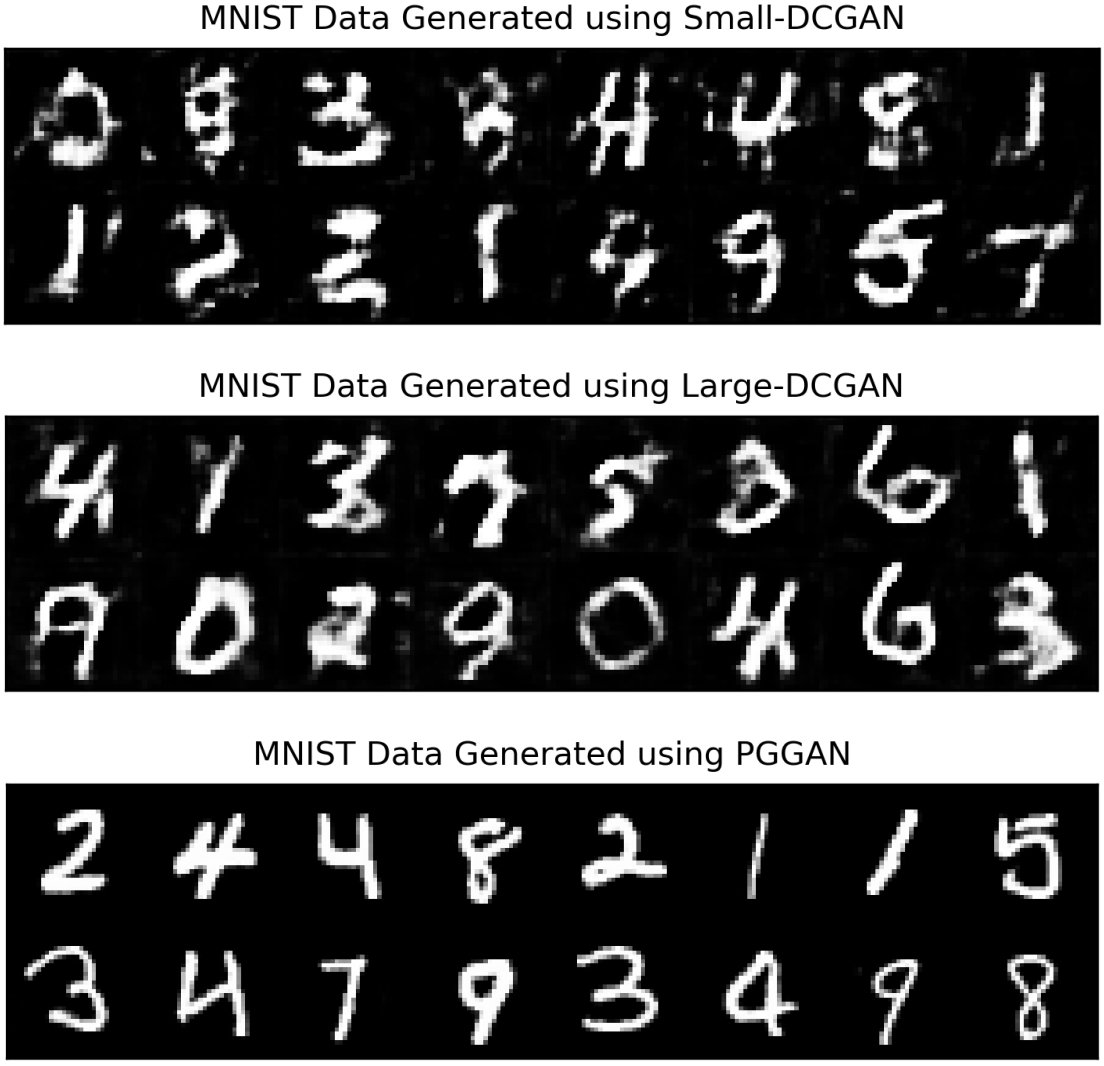

Figure 4-7 Samples from the GAN architectures trained on the MNIST dataset.

4.2.3 Modeling the Discriminator Output

To perform the importance sampling it was necessary to evaluate the statistical properties of the discriminator output for each GAN architecture to determine the cutoff thresholds. 3200 random samples were generated for each category modelled by the GAN architecture. These samples were fed into the discriminator and used to compute the output response. Note that for the Small-DCGAN and Large-DCGAN architectures the output response is the raw probability, while for the

PGGAN the output response is a score value. The larger the value of the discriminator output, the more confidence it has that the sample is a good representative of the raw dataset. The mean and variance of the discriminator outputs was calculated and the average values across the image categories are presented in Table 4-1. We observe that for the Small-DCGAN and Large-DCGAN, that the average output values are less than 0.5, and with very small variance. This indicates that for a random GAN sample, the discriminator is consistently more likely to believe that the samples are fake. Considering the relative scores of the PGGANs, we see that the ISIC 2018 GAN has the greatest uncertainty regarding the discriminator output. This can be explained by recognizing that the ISIC 2018 dataset is highly unbalanced, and some categories had less than 200 images. Therefore, the resulting variance in the discriminator output is larger.

Table 4-1 Statistical description of discriminator output

Model Architecture	**Dataset**	**Average Discriminator Output**	**Average Discriminator Variance**
Small-DCGAN	MNIST	0.47	2.64e-05
Large-DCGAN	MNIST	0.46	4.47e-05
PGGAN	MNIST	1.26	1.60
PGGAN	LSUN	-14.42	190.10
PGGAN	ISIC 2018	-59.69	357.14

4.2.4 Assessing the GAN Sample Quality

To evaluate the quality and diversity of the samples produced by the GANs, the IS and the FID were calculated for each of the GANs. Additionally, the IS score for the raw dataset images was calculated to provide a baseline. For the MNIST and LSUN datasets, 1000 images were sampled from each image category to evaluate the metrics. As the ISIC 2018 dataset is highly imbalanced, an equal number of GAN and raw samples were drawn from the different categories. The result is shown in Table 4-2. Examining the results in this table, we see that the IS for the MNIST dataset provided very little information, in fact the metric stated that the score for the GAN generated data was actually better than the score for the raw data. This demonstrates a limitation of the IS. We are only able to get relevant inception scores for images which resemble the data from the ImageNet dataset that the Inception network was trained using. Therefore, as the MNIST dataset, and to some extent, the ISIC 2018 dataset are far removed from the content in the ImageNet dataset, the validity of the metric diminishes. However, for the LSUN dataset, the resulting IS values are close to results stated in prior literature (Karras et al. 2017). Notice how the FID score for the MNIST dataset decreases as we progress from the Small-DCGAN model to the PGGAN. This can be explained as the number of parameters of the Small-DCGAN is less than that of the Large-DCGAN which in turn is less than the PGGAN. Therefore, we see that as the capacity of the GAN increases, the FID score decreases as would be expected.

Table 4-2 IS and FID calculated for different GAN architectures.
Note that larger Inception score values indicate better generated results
and smaller FID scores indicate better generated results.

Model Architecture	**Dataset**	**Inception Score (Raw Data)**	**Inception Score (GAN)**	**FID Score**
Small-DCGAN	MNIST	2.07	2.10	135.06
Large-DCGAN	MNIST	2.07	2.27	92.27
PGGAN	MNIST	2.07	2.10	14.09
PGGAN	LSUN	9.60	8.17	10.28
PGGAN	ISIC 2018	3.51	3.12	78.83

4.3 Interpreting Network Uncertainty

To qualitatively understand how uncertainty manifests itself in the network, the following experiment was performed. A classifier was trained on the MNIST dataset. Then the BALD acquisition function was evaluated for all the real training images as well as 10,000 images sampled from the PGGAN. The real images with high and low BALD scores are shown in Figure 4-8. Likewise, the PGGAN generated images with high and low BALD scores are shown in Figure 4-9. A high BALD score represents that the network is uncertain about the label of the sample. Notice how many of the high BALD score samples are visually challenging to interpret and there are some which are ambiguous. Likewise, the images with low BALD scores are very discernable. Therefore, we can qualitatively infer that the network uncertainty described by the BALD score relates to our human interpretation of visual uncertainty.

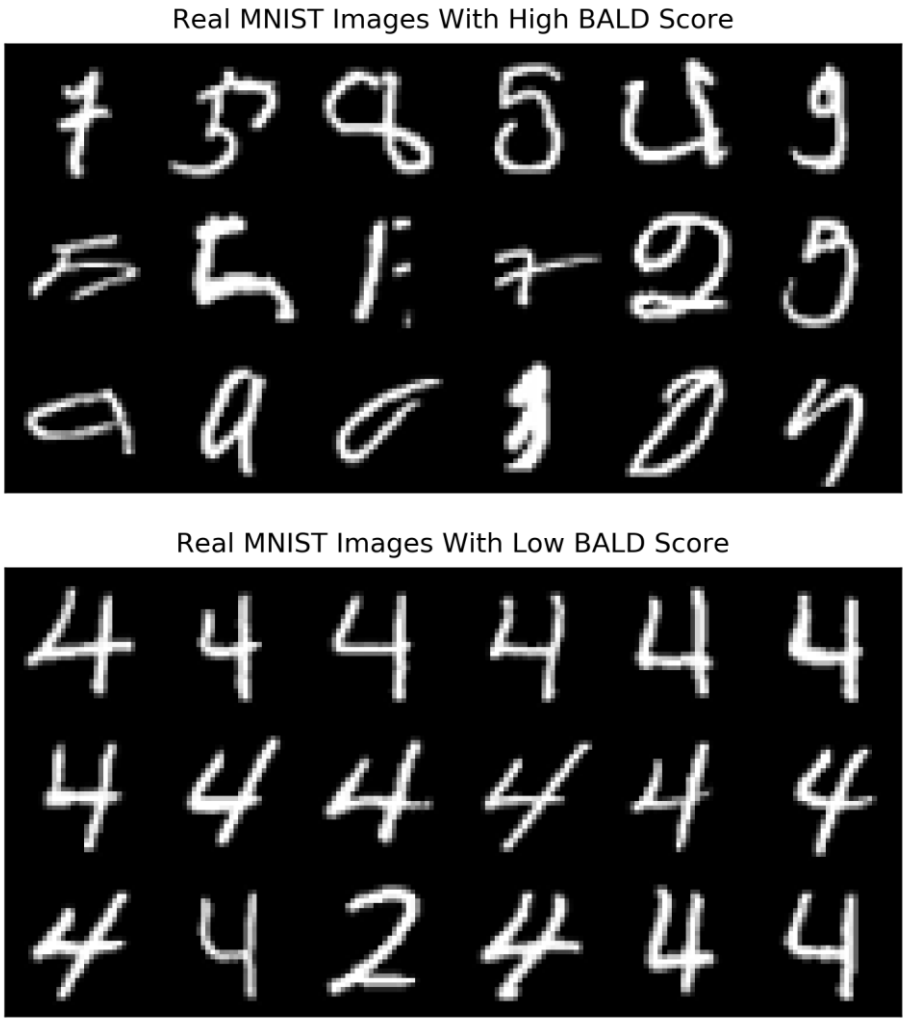

Figure 4-8 Raw MNIST images with high BALD scores (top) and low BALD scores (bottom).

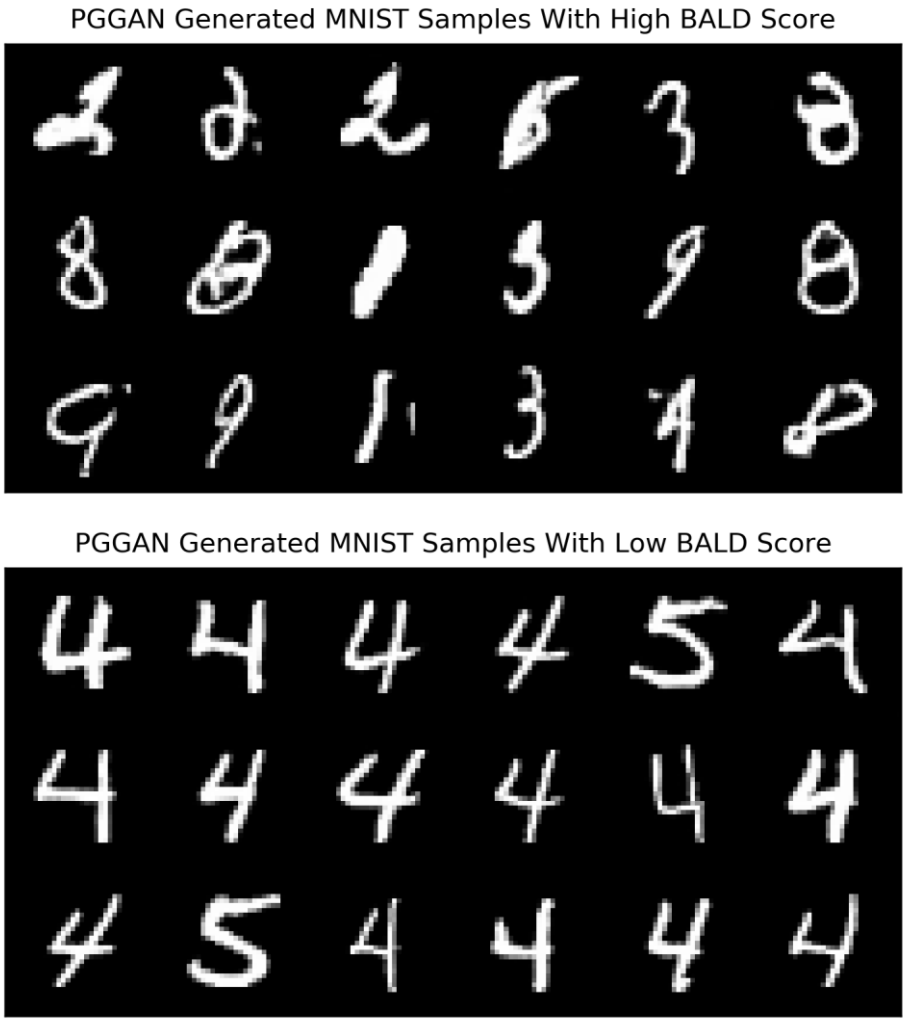

Figure 4-9 PGGAN generated MNIST images with high BALD scores (top) and low BALD scores (bottom).

4.4 Classification Experiments

The fundamental goal of the classification experiments was to address the core research questions stated in Section 1.2. As the time required to train a classifier for the LSUN and ISIC 2018 datasets was roughly two orders of magnitude larger than the time required to train a classifier for MNIST, the majority of experiments were performed using the MNIST dataset. To maximize the cohesiveness in the presentation of the experimental results, all analysis and discussion of the results will be reported in Section 4.5.

4.4.1 Experiments with MNIST

The purpose of the MNIST dataset experiments was to determine the relative performance when training a classifier using different data sources and acquisition functions. The Small-DCGAN, Large-DCGAN, and PGGAN training datasets each had 10,000 images where 1,000 images were sampled from each of the 10 categories. The raw MNIST training dataset consisted of 60,000 images with roughly 6,000 in each category. To measure classification performance, a test set was built using 1,000 raw images with 100 from each category. The experiments on the MNIST dataset were performed in the following way:

- At the start of each iteration, the classification network was initialized with a set of new random weights. During each iteration the chosen acquisition function was used to sample 10 images from the given data source. These new images were added to the training dataset. The classification network was then trained using this dataset for 100 epochs. The resulting final balanced accuracy was then computed and stored for each iteration. This process continued for 50 iterations. At this point the training set size had reached 500 images.

- The first set of experiments examined the performance when each data source was used independently i.e. (the classifier was only trained on data from one of the four possible data sources: raw MNIST dataset, Small-DCGAN, Large-DCGAN, and PGGAN).
- The second set of experiments examined the performance when the GAN data sources were used to augment the raw dataset i.e. (the classifier was trained on a dataset consisting of the raw MNIST dataset combined with data from one of the three possible GAN data sources: Small-DCGAN, Large-DCGAN, and PGGAN).
- For all experiments, classification performance was measured using each combination of data source and acquisition function. Additionally, each experiment was repeated 4 times to establish a confidence interval for the accuracy estimate.

4.4.2 Experiments with LSUN and ISIC 2018

The goal of the experiments performed using the LSUN and ISIC 2018 datasets was to compare the classification performance when training using raw data and data generated from the PGGAN. The classifiers were trained under random sampling or BALD acquisition. The experimental procedure can be described as follows:

- Each experiment used either random sampling or BALD acquisition as the chosen acquisition function.
- During each experiment, 100 images were randomly selected from the raw dataset to form the initial training set. Every 5 epochs as the classification network was trained, 10 new images for each category were selected either using the chosen acquisition function and added to the training set.

- Each experiment was repeated 10 times to establish a confidence interval for the accuracy estimate.

4.5 Assessment of Research Questions using Experimental Observations

This section addresses each of the research questions proposed in Section 1.2 through analysis of the experimental results.

4.5.1 What Is the Difference in Classifier Performance If We Train Using Purely GAN Synthesized Data vs. Raw Data?

Examining the results for the first set of MNIST experiments shown in Figure 4-10 we observe that under random acquisition the performance of the raw data source is optimal, however under BALD or max entropy acquisition, the PGGAN data source is optimal. The fact that a classifier trained on purely synthetic data can outperform a classifier trained on raw data is quite fascinating. This implies that the images sampled from the GAN are more informative for training the classifier than the images from the raw dataset. A possible explanation for this observation is that while the distribution of samples from the GAN has less diversity than the raw dataset, the GAN is capable of generating samples which are highly informative for classification. Therefore, under an appropriate acquisition function these informative samples can be selected by the importance sampling mechanism and added to the training set, resulting in increased classification performance.

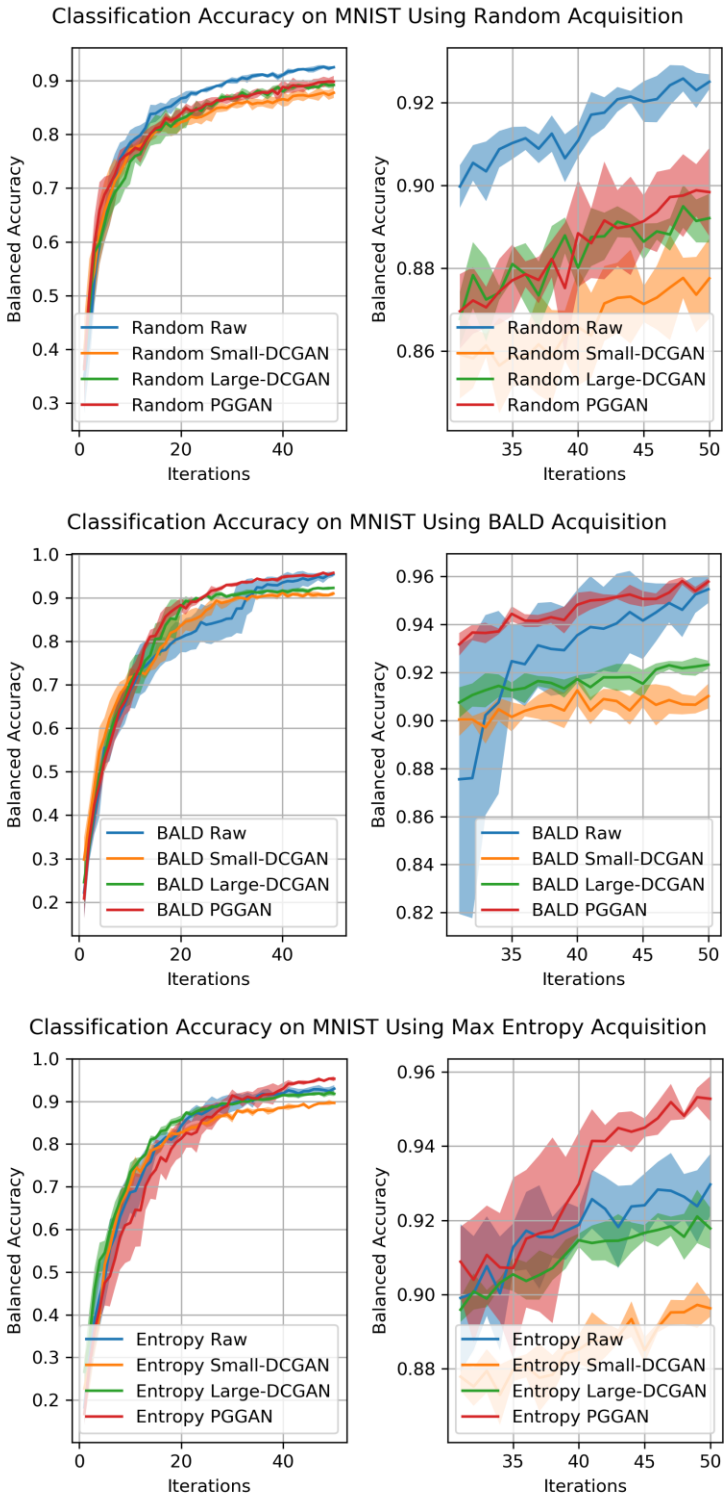

Figure 4-10 MNIST classification performance under various acquisition functions. The plots on the left show the balanced accuracy for all 50 iterations, while the plots on the right show the balanced accuracy for the final 20 iterations. The shaded area around each line signifies a confidence interval of one standard deviation.

For the ISIC 2018 dataset we notice a significant difference in classification performance between the purely GAN synthesized data and the raw data as indicated in Figure 4-11 with the classifiers trained on raw data achieving a 10 percentage point improvement in balanced accuracy over the classifiers trained on purely synthetic data. With this said, the classifiers trained on purely synthetic GAN data still performed reasonably well given the complexity of the classification task, resulting in balanced accuracy across the 7 classes of 0.52 and a ROC AUC score of 0.7.

The results provided by the LSUN experiments in Figure 4-12 were rather intriguing as the performance of the classifiers trained on purely synthetic data was nearly identical to the performance of the classifiers trained using raw data. This implies that the images sampled from the GAN are equally informative for training the classifier as the images from the raw dataset.

In summary, the experimental results have demonstrated that training a classification network using synthetic data from a GAN is not only feasible, but under an appropriate acquisition function can surpass the performance of a classifier trained using raw data. It was observed that the final classification performance is dependent on the quality of the GAN used to synthesize the training images.

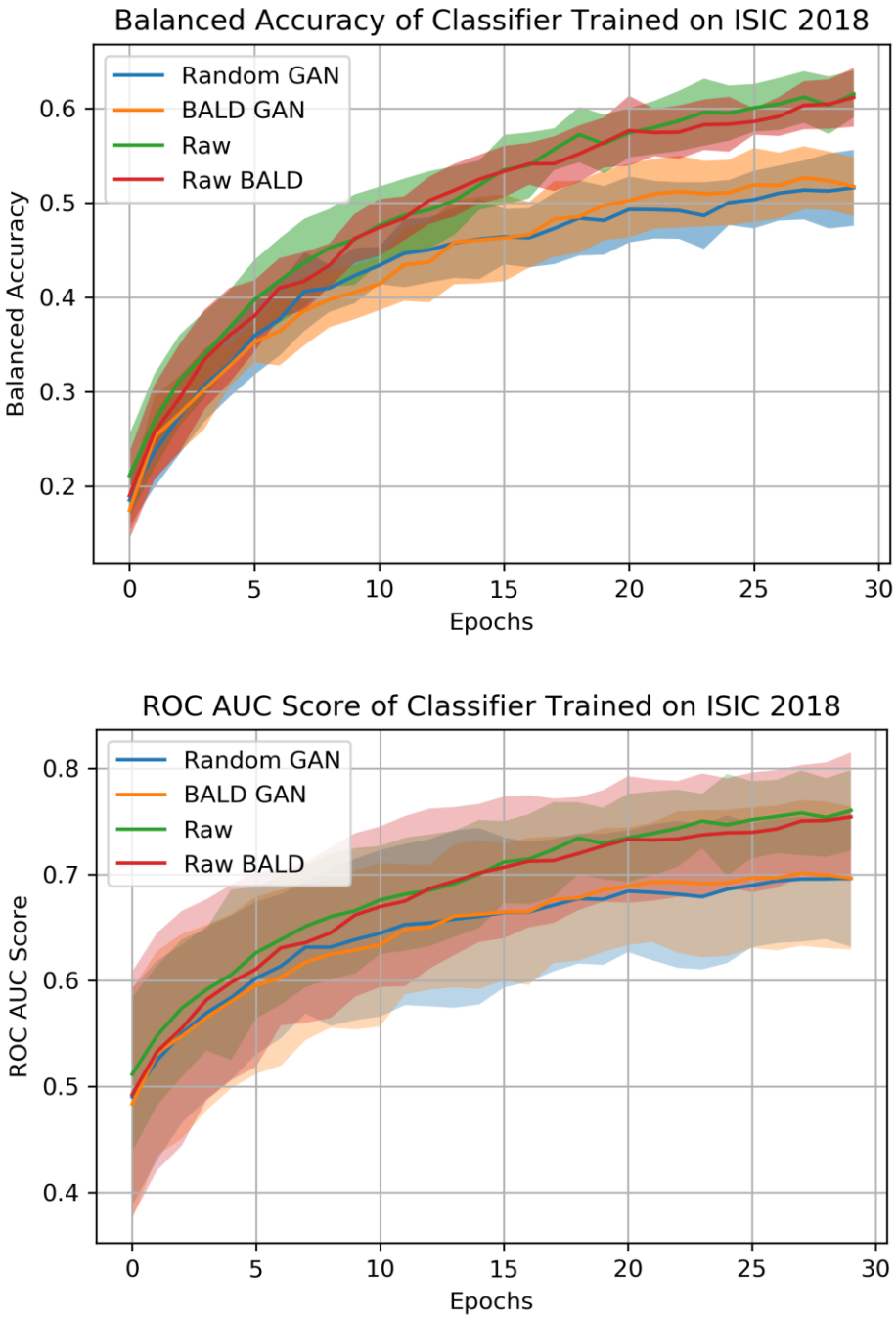

Figure 4-11 ISIC 2018 classification performance. The plot on the top shows the balanced accuracy and the plot on the bottom shows the ROC AUC score. The shaded area around each line signifies a confidence interval of one standard deviation.

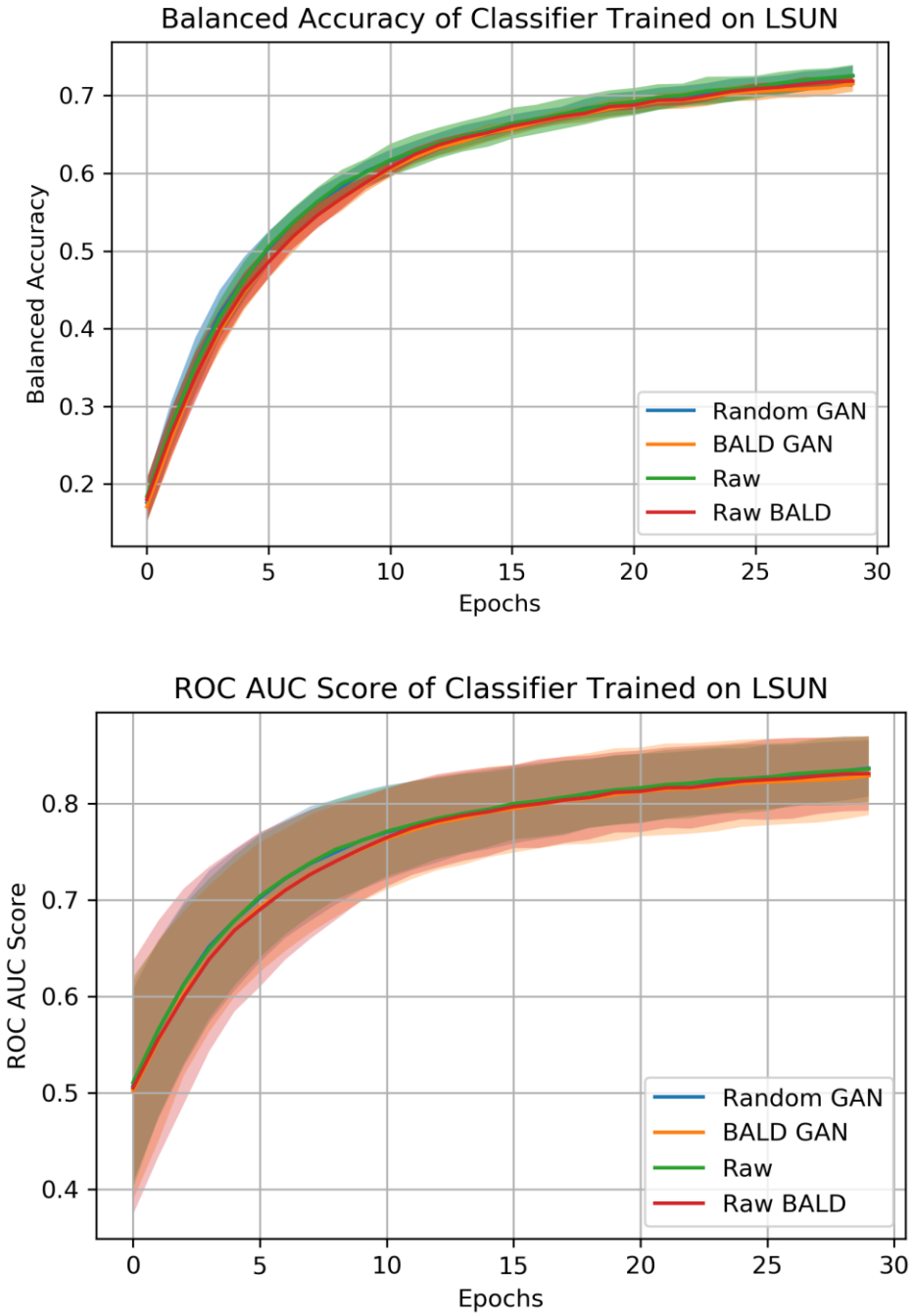

Figure 4-12 LSUN classification performance. The plot on the top shows the balanced accuracy and the plot on the bottom shows the ROC AUC score. The shaded area around each line signifies a confidence interval of one standard deviation.

4.5.2 What Is the Difference in Classifier Performance If We Train Using Random Chosen Samples vs. Samples Chosen Based off Classifier Prediction Uncertainty?

The results of the MNIST experiments displayed in Figure 4-13 show that classifiers trained under BALD acquisition performed consistently better than classifiers trained under random sampling or max entropy acquisition. Intuitively it is understandable that BALD acquisition should perform better than random sampling, however the consistent performance gain over max entropy acquisition is intriguing. To develop an explanation for this behavior, let us consider the definition of the BALD acquisition function shown below

$$U(\mathbf{x}) \approx H\left[\frac{1}{N}\sum_{n=1}^{N} P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right] - \frac{1}{N}\sum_{n=1}^{N} H\left[P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right] \tag{4.2}$$

The key difference between this expression and max entropy acquisition is the term $H\left[\frac{1}{N}\sum_{n=1}^{N} P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right]$. This term will be large if the network prediction fluctuates between the MC samples. A possible explanation for why BALD acquisition outperforms max entropy acquisition is due to the properties of the softmax activation function used in the classification network. As described by (Guo et. al 2017), the softmax activation has been shown to overestimate the network prediction certainty, causing the term $\frac{1}{N}\sum_{n=1}^{N} H\left[P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right]$ to be smaller than we would expect. Hence, by measuring how much the network predictions change between MC samples given by $H\left[\frac{1}{N}\sum_{n=1}^{N} P(y \mid \mathbf{x}, \boldsymbol{\omega}_n)\right]$ we get a better estimate of the uncertainty. The addition of this extra term is what makes the BALD acquisition function a better estimator for network uncertainty.

The classification results for the ISIC 2018 and LSUN datasets are shown in Figures 4-11 and 4-12 respectively. Notice how the classifiers trained under BALD acquisition are very similar in performance to the classifiers trained under random acquisition. This implies that BALD acquisition is not providing information gain during training. A possible explanation for this observation is that given the complexity of the classification network, too few MC samples are used to accurately estimate of the classifier uncertainty.

In summary, the results for the MNIST dataset have shown that the classifier performance improved significantly under the BALD and max entropy acquisition. However, for the ISIC 2018 and LSUN dataset, BALD acquisition had no significant effect. Further investigation will be needed to evaluate the effective use of BALD acquisition for higher dimensional classification tasks.

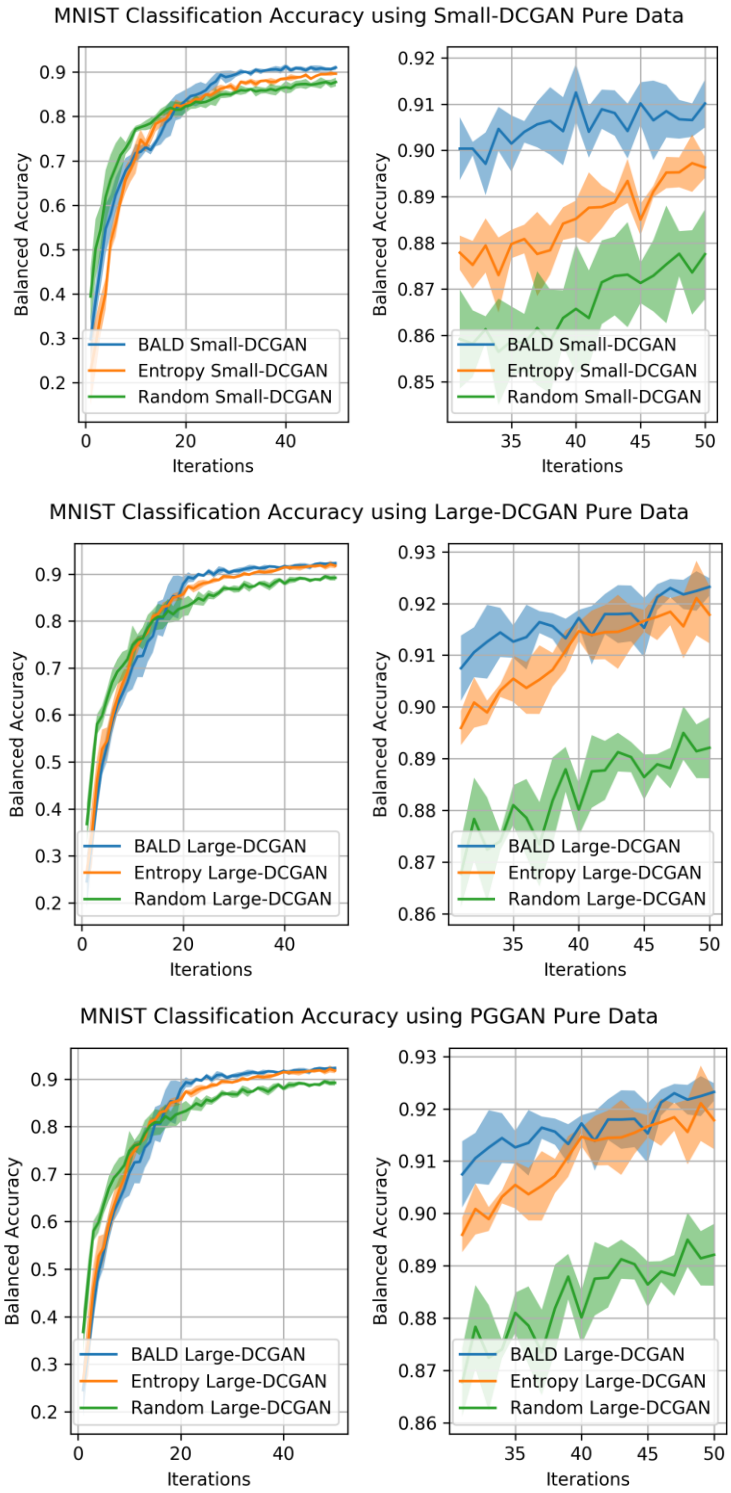

Figure 4-13 MNIST classification performance using pure GAN data. The plots on the left show the balanced accuracy for all 50 iterations, while the plots on the right show the balanced accuracy for the final 20 iterations. The shaded area around each line signifies a confidence interval of one standard deviation.

4.5.3 How Is the Capacity of the GAN Used to Generate Training Images Correlated with the Final Classification Performance?

To address this question, we can examine the results of the MNIST experiments in Figure 4-10. We see that the ordering of the GANs in terms of final classification performance is PGGAN > Large-DCGAN > Small-DCGAN. This is the same order as the capacity of the networks. Therefore, we have experimental evidence to infer that the classification performance is positively correlated with the capacity of the GAN. This makes intuitive sense since a higher capacity GAN will be able to generate results which are higher quality, more representative of the dataset, and hence better fit for training a classifier.

4.5.4 What Overall Performance Gain Can We Achieve from Using GAN Augmentation?

To determine the impact of GAN data augmentation on classifier performance, let us examine the results provided by the second set of MNIST experiments whose primary results are shown in Figure 4-14. Figure 4-15 shows the resulting classifier performance under random acquisition when data augmentation was performed. We see that augmentation using the PGGAN samples achieved the best overall accuracy by a small margin and surpassed the accuracy of the classifier trained on the raw dataset. We also notice that the Large-DCGAN augmented dataset performed on par with the raw dataset, and that the Small-DCGAN augmented dataset performed worse. This demonstrates that GAN augmentation has the potential to improve the classifier performance. However, if the GAN does not have sufficient capacity, then the classifier performance can decrease.

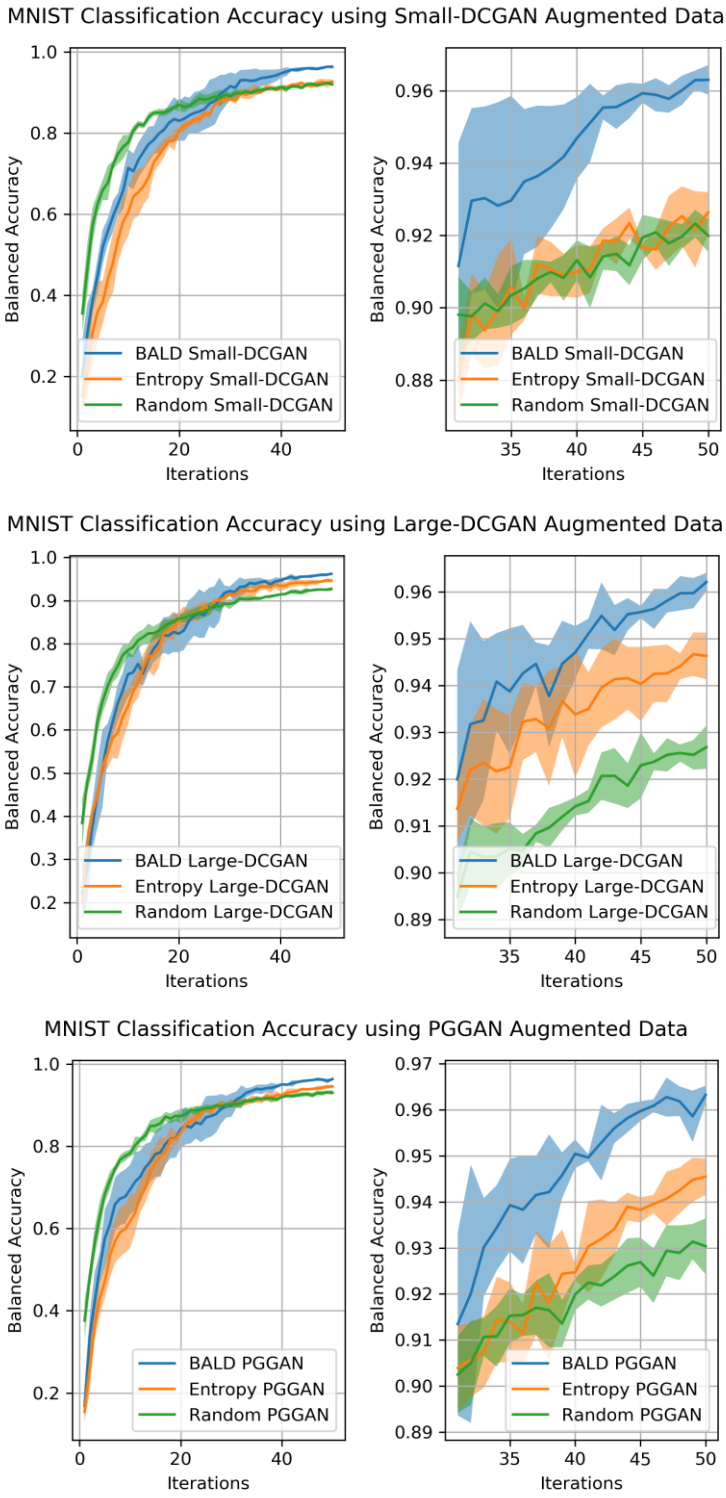

Figure 4-14 MNIST classification performance using augmented GAN data. The plots on the left show the balanced accuracy for all 50 iterations, while the plots on the right show the balanced accuracy for the final 20 iterations. The shaded area around each line signifies a confidence interval of one standard deviation.

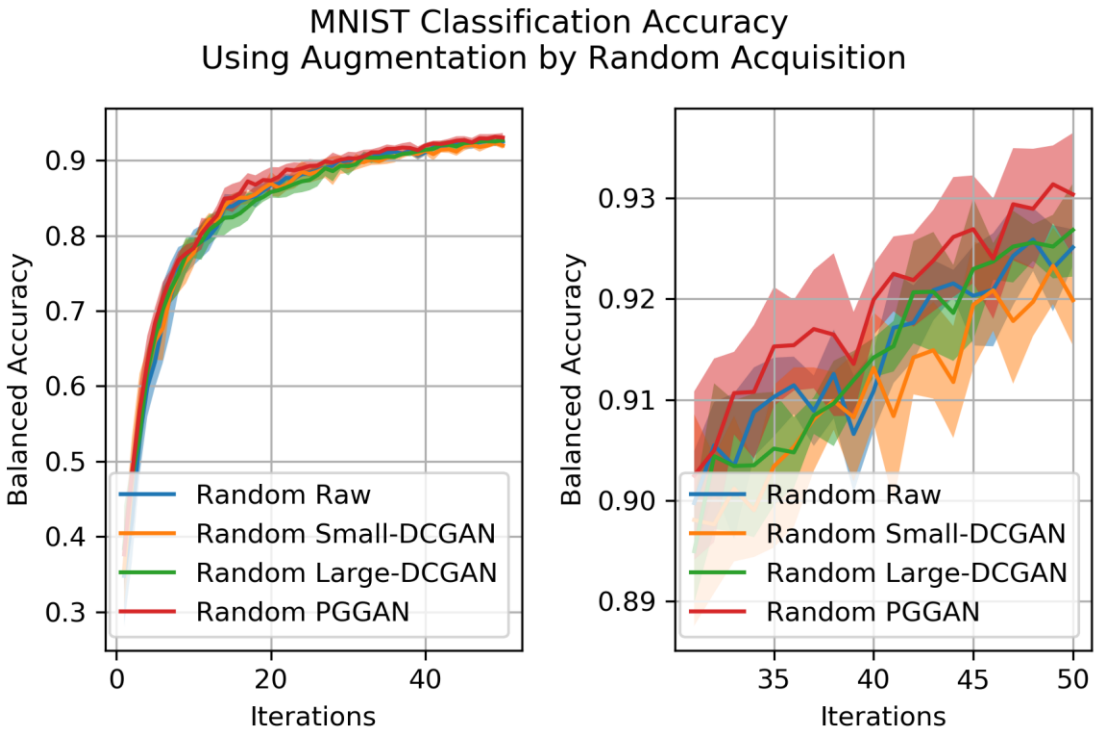

Figure 4-15 MNIST classification performance for GAN data augmentation under random acquisition. The plot on the left shows the balanced accuracy for all 50 iterations, while the plot on the right shows the balanced accuracy for the final 20 iterations. The shaded area around each line signifies a confidence interval of one standard deviation.

Examining the data from Figure 4-14 we see that BALD acquisition had the best performance for all classifiers. In Figure 4-16 the results of training the classifiers under BALD acquisition are shown together with the performance of the raw data classifier trained under random sampling added as a baseline. The final test accuracy for each of the trained classifiers are shown in Table 4-3. We see that the performance of the classifiers trained using augmentation by BALD acquisition outperformed the classifiers trained on the raw datasets. From the final test accuracies, we see that the BALD PGGAN augmented dataset had an increase of 3.82 percentage points over the Random Raw dataset, and an increase of 0.86 percentage points over the BALD Raw dataset.

These results indicate that GAN data augmentation using the developed classification framework can considerably improve the classification performance.

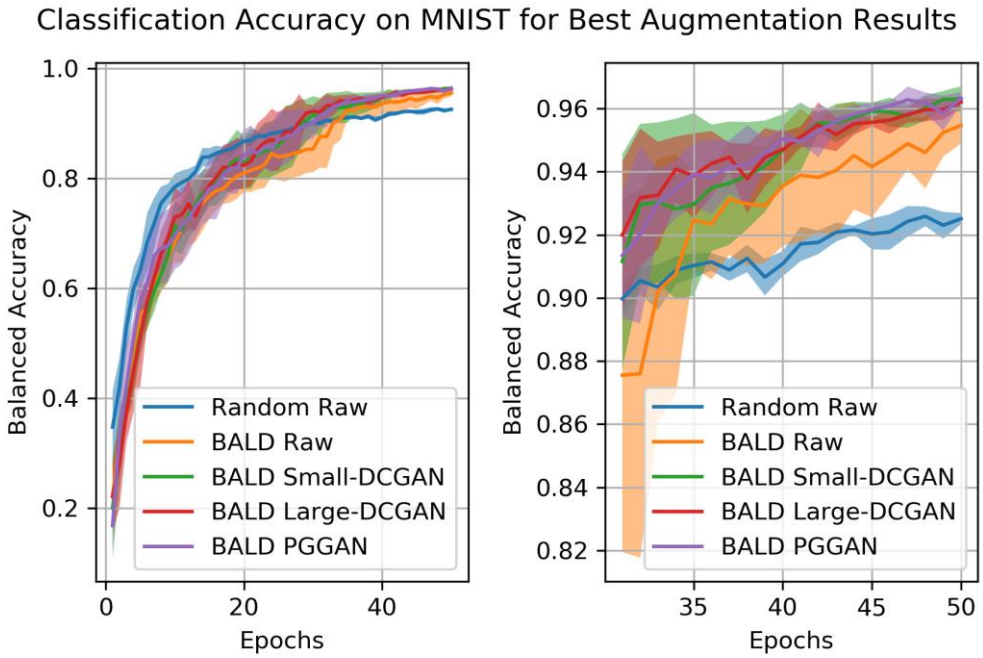

Figure 4-16 Plot of the best performing classifiers trained using GAN augmented MNIST data. The plot on the left shows the balanced accuracy for all 50 iterations, while the plot on the right shows the balanced accuracy for the final 20 iterations. The shaded area around each line signifies a confidence interval of one standard deviation.

Table 4-3 Final test accuracy for best performing classifiers on MNIST using GAN data augmentation.

MNIST Classifier Training Type	**Final Test Accuracy**
Random Raw	0.9251
BALD Raw	0.9547
BALD Small-DCGAN	0.9630
BALD Large-DCGAN	0.9620
BALD PGGAN	0.9633

Through the experimental results we have seen that training a classification network using a dataset augmented with synthetic GAN samples can improve the overall performance of the classifier. Additionally, the importance sampling mechanism was shown to the further improve the classifier performance, especially for the GANs with lower capacity.

4.6 Summary

In this chapter the results were presented for the experimental work conducted in this thesis. By means of visual inspection and computation of the IS and FID metrics, the quality and diversity of the generated GAN images were assessed. Qualitatively it was observed that the GAN samples closely resembled the raw images from the dataset. Furthermore, it was shown that the quality of the GAN samples was dependent on the capacity of the GAN. Through grouping images by BALD acquisition score, it was qualitatively demonstrated that the images with the largest prediction uncertainty were visually harder to identify due to ambiguity in the shape, or lack of geometric detail, while the images with the smallest prediction uncertainty were highly distinguishable and unambiguous. Two sets of classification experiments were conducted using the MNIST dataset. The first set examined the classification performance using training data sampled from a single data source, while the second set examined the classification performance using augmented training data consisting of samples from the original dataset and a GAN. Additional classification experiments were performed using the LSUN and ISIC 2018 to assess the performance when training on an augmented dataset. Thorough analysis of these results was provided and all of the research questions from Section 1.2 were addressed.

Chapter Five: Applications

The purpose of this chapter is to explore possible applications for the technology developed in this thesis. Section 5.1 discusses the potential of using the GAN trained on the ISIC 2018 dataset as a mechanism to investigate potential disease progressions. Section 5.2 describes how the developed classification framework could be used for active learning. Section 5.3 presents the possibility of using the importance sampling mechanism developed in this thesis to improve training efficiency for reinforcement learning. Section 5.4 summarizes the presented material.

5.1 Disease Progression Analysis

Monitoring disease progression is a critical component of optimal medical treatment planning (Manley 2007). A mechanism to generate images describing the potential progression of a disease could be useful for physicians to evaluate intervention strategies. One application of the GAN architecture is the ability to smoothly transition between two images by generating samples using latent space interpolation. Specifically, for GANs trained on medical images, such as the PGGAN trained for this thesis on the ISIC 2018 dataset, latent space interpolation can be used to generate images describing a possible disease progression. As an example, consider a latent space vector representing a melanoma skin lesion. By extrapolating along different latent dimensions, we can transform the image to visualize the progressive change. Figure 5-1 demonstrates how the melanoma latent vector can be transformed to generate images describing possible disease trajectories.

The progressions visualized in Figure 5-1 were randomly generated and have no specific medical significance. However, by providing actual disease progression data or including additional labels on the data describing features such as disease severity, we can train a classifier on the latent space to identify how the different latent space axis correspond to different disease progression trajectories. This would enable a physician to use the GAN latent space as an exploratory tool to investigate realistic disease progressions.

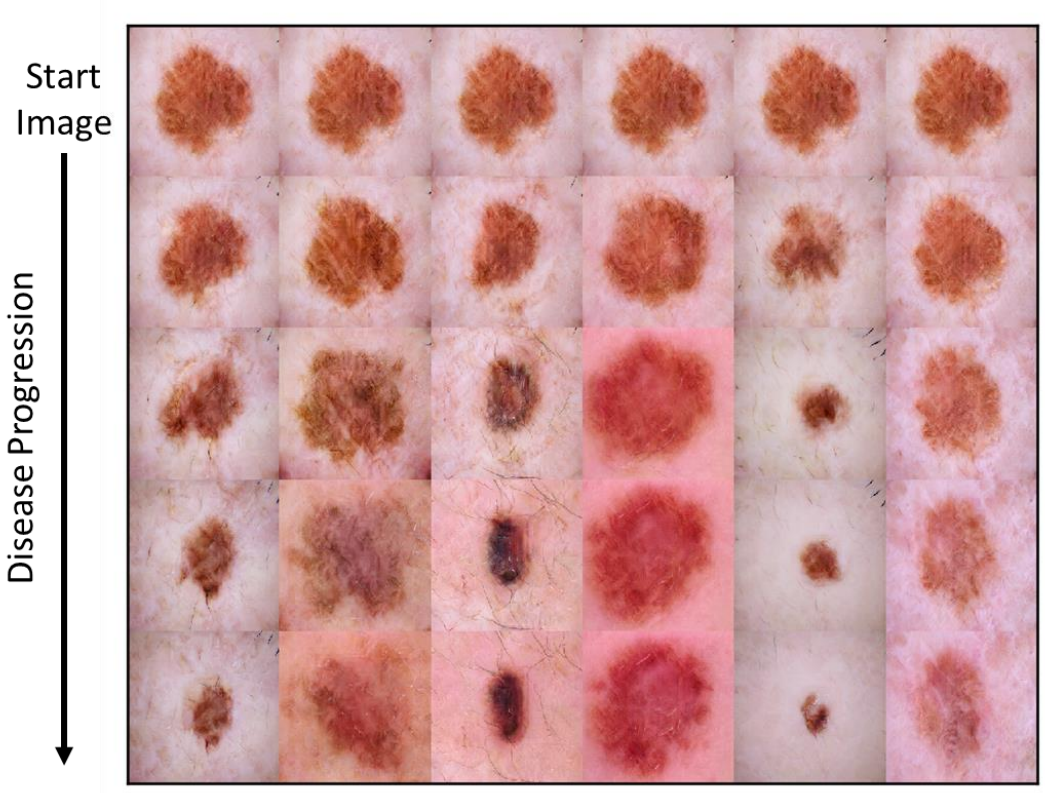

Figure 5-1 Using the PGGAN trained on the ISIC 2018 dataset to synthesize possible melanoma disease progressions. The top row shows the starting image. A possible disease progression is represented by each column. Note that the progressions in this figure were randomly generated and have no specific medical significance.

5.2 Active learning

Although augmentation strategies can be used to improve classification accuracy, there comes a point when additional raw data is necessary. Labelling data can be very expensive for specialized domains such as medical radiology. As such, it is critical that the labelling is prioritized for data which can significantly improve classification performance. The process of prioritizing the data for labelling and then using this newly labelled data to train a classifier is called active learning (Settles 2011). The work performed in this thesis demonstrated how the BALD and max entropy acquisition functions could assess which GAN images to sample for the biggest impact in classification performance. The developed techniques could be adapted to form an active learning framework where the acquisition functions were applied to unlabelled data. In fact, the BALD acquisition function was originally developed for active learning purposes (Houlsby et al. 2011). The benefit of this approach would be to minimize the amount of labelled data required to achieve the desired classification performance.

5.3 Reinforcement Learning

Reinforcement learning (RL) is the domain of machine learning where an agent is trained to make optimal decisions for a given environment (Gosavi 2009). RL is used heavily in robotics for tasks such as driverless cars or automated surgery (Wang et al. 2018). A key challenge with RL is the vast number of training samples typically needed for models to converge. This is due to the large number of uninformative samples typically encountered during training. For example, the vast majority of driving data consists of standard road conditions, however a driverless car must know how to respond to the critical rare occurrence incidents such as car accidents. One possible way to improve RL training is to use a GAN to generate training examples representing important

environmental edge cases. The mechanism proposed in this thesis for importance sampling from GANs has the potential to be applied for RL tasks to sample training examples which can improve the training efficiency.

5.4 Summary

This chapter presented several possible applications of the technology developed in this thesis. The ability to generate highly realistic synthetic medical images has great potential to assist physicians with treatment planning through providing tools capable of visualizing and predicting how various disease trajectories may progress. Additionally, the ability to model the uncertainty in a classification network has beneficial ramifications for the training of active learning and reinforcement learning systems.

Chapter Six: Conclusions

6.1 Thesis Summary

The work presented in this thesis demonstrated that it is possible to train an image classification network using purely synthetic images from a GAN and that the classifier performance is correlated with the capacity of the GAN. Furthermore, it was shown that using an appropriate acquisition function, the performance of the classification network could be improved. An interesting result was that the performance of a MNIST classifier trained under BALD acquisition using images synthesized from a PGGAN was superior to an equivalent classifier trained using raw data. The performance gain from augmenting a raw dataset with GAN synthesized images was shown to be measurable but dependent on the capacity of the GAN. In conclusion, the experimental results of this thesis demonstrate that GAN augmentation using importance sampling is advantageous for image classification. Overall, the developed technology has potential for application in medical image classification as well as other applications in the domains of active learning and RL.

6.2 Contributions

Through the process of addressing the research questions stated in Section 1.2, the following contributions were made:

- It was demonstrated how a GAN could be trained to synthesize high resolution medical images representing the ISIC 2018 dataset. The quality and diversity of the generated

samples was advocated for by the observation that the generated synthetic images could improve classification performance.

- An iterative training loop algorithm was developed to incrementally build up the training set from GAN generated images to maximize the final performance of the classifier. Custom neural network architectures were designed for the Small-DCGAN and Large-DCGAN, and the CNN classifiers.
- An importance sampling mechanism was developed to prioritize samples based on the impact they would have on classification performance. The mechanism was shown to provide substantial performance improvement for the MNIST dataset.
- The potential positive impact of using GAN synthesized data augmentation was validated by means of the experimental results. By thorough analysis, all research questions posed for this thesis were answered.
- Published developed work in the CVPR 2019 Workshop on Uncertainty and Robustness in Deep Visual Learning (Nielsen et al. 2019).

6.3 Future Work

- Recent work has been directed towards the challenge of insuring that a neural network complies with the definition of differential privacy (Abadi et al. 2016). A next step for the work developed in this thesis would be to incorporate differential privacy training mechanisms into the construction of the GAN and classifier networks. This would further enforce the anonymity of the GAN synthesized data.
- With the current GAN architecture, there is no direct way to infer the latent vector from a given image. This functionality would be beneficial to generate additional training images

by manipulating the latent vectors for specific images that the network is uncertain about. One GAN architecture that addresses this issue is called the Bidirectional GAN (BIGAN) (Donahue et al. 2016). The network modifies the discriminator by adding an encoder that is trained to predict the latent vector for a given image. Once the network is trained, this encoder network can be used to encode the image into its latent space representation. A next step for the work in this thesis would be to utilize a BIGAN architecture such that latent space manipulation could be used to improve the sample quality used for augmentation.

- In the experimental results it was shown that classifier performance is positively correlated with the capacity of the GAN used to augment the dataset. Currently the FID and IS are applied as metrics to assess the capacity of the GAN. In future work it would be valuable to quantify the relationship between the FID and IS output and the performance of the classifier trained using GAN augmented data, such that a metric could be developed to predict the potential benefit an arbitrary GAN would provide for data augmentation.

References

M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 308–318,* 2016.

M. Abadi, A. Agarwal et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv:1603.04467,* 2016.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine Learning, 50, 5–43,* 2003.

M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875,* 2017.

B. K. Beaulieu-Jones, Z. S. Wu, C. Williams, and C. S. Greene, "Privacy-preserving generative deep neural networks support clinical data sharing," *bioRxiv,* 2017.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *CoRR, abs/1505.05424,* 2015.

M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316,* 2016.

C. Callison-Burch, and M. Dredze, "Creating Speech and Language Data with Amazon's Mechanical Turk," *Workshop on Creating Speech And Language Data With Amazon's Mechanical Turk NAACL HLT, 1-12,* 2010.

R. Caruana, S. Lawrence, L. Giles, "Over?tting in neural nets: backpropagation, conjugate gradient, and early stopping," *Adv Neural Inf Processing Syst. 2001:402-408,* 2001.

T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv:1612.02136,* 2016.

F. Chollet, et al. "Keras," *https://keras.io,* 2015.

N. C. F. Codella et al. "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), Hosted by the International Skin Imaging Collaboration (ISIC)," *Preprint at https://arxiv.org/abs/1710.05006,* 2017.

B. Csáji BC, "Approximation with artificial neural networks," *Faculty of Sciences, Etvs Lornd University, Hungary,* 2001.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," *CVPR09*, 2009.

L. Denton, S. Chintala, R. Fergus, et al. "Deep generative image models using alaplacian pyramid of adversarial networks," *NIPS, pages 1486–1494*, 2015.

C. Doersch, "Tutorial on variational autoencoders," *arXiv:1606.05908*, 2016.

J. Donahue, P. Krahenbuhl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

Y. Gal, Z. Ghahramani, "Dropout as a bayesian approximation: representing model uncertainty in deep learning," *arXiv preprint arXiv:1506.02142*, 2015.

Y. Gal, "Uncertainty in Deep Learning," *PhD thesis, University of Cambridge*, 2016.

X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *AISTATS*, 2010.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, et al. "Generative adversarial nets," *NIPS*, 2014.

A. Gosavi, "Reinforcement Learning: A Tutorial Survey and Recent Advances," *INFORMS Journal on Computing, vol. 21, no. 2, pp. 178– 192*, 2009.

J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, et al. "Recent advances in convolutional neural networks," *arXiv 1512.07108*, 2015.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," *CoRR, abs/1704.00028*, 2017.

G. Guo, Y. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," *arXiv preprint arXiv:1706.04599*, 2017.

K. Hajian-Tilaki, "Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation," *Caspian J. Intern. Med. 2013, 4, 627–635*, 2013.

B. Hanin, Boris and D. Rolnick, "How to start training: The effect of initialization and architecture," *arXiv preprint arXiv:1803.01719*, 2018.

K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of CVPR, pages 770–778*, 2016.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in Neural Information Processing Systems (pp. 6626–6637)*, 2017.

M. B. Hossin, M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *Int. J. Data Min. Knowl. Manag. Process 2015, 5, 1–11,* 2015.

N. Houlsby, F. Huszar, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," *arXiv preprint arXiv:1112.5745*, 2011.

G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maate, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, 2016.

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

K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *CoRR*, 2017.

K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," *CoRR*, 2017.

T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *CoRR, abs/1812.04948*, 2018.

T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

A. Kendall, Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *ArXiv:1703.04977*, 2017.

D. P. Kingma, and M. Welling, "Auto-Encoding Variational Bayes," *The 2nd International Conference on Learning Representations (ICLR)*, 2013.

J. Kingston, "Using artificial intelligence to support compliance with the general data protection regulation," *Artificial Intelligence and Law 25, 429–443,* 2017.

A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *NIPS*, 2012.

J. Lawrence, J. Malmsten, A. Rybka, D. A. Sabol, and K. Triplin, "Comparing TensorFlow Deep Learning Performance Using CPUs, GPUs, Local PCs and Cloud," *proceedings of Student-Faculty Research Day, CSIS, Pace University, 1-7,* 2017.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE, 86, 2278–2324,* 1998.

W. Liu, Z. Wanga, X. Liua, N. Zengb, Y. Liuc and F. E. Alsaadid, "A Survey of Deep Neural Network Architectures and Their Applications," *network architectures and their applications, Neurocomputing 234 11–26,* 2017.

D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, "Exploring the limits of weakly supervised pretraining," *arXiv preprint arXiv:1805.00932,* 2018.

H. J. Manley, "Disease progression and the application of evidence-based treatment guidelines – Diagnose it early: A case for screening and appropriate management," *J Manag Care Pharm 2007: 13: S6–S12,* 2007.

L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?" *ICML,* 2018.

M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR, abs/1411.1784,* 2014.

C. Nielsen and M. Okoniewski, "GAN Data Augmentation Through Active Learning Inspired Sample Acquisition," *CVPR Workshop on Uncertainty and Robustness in Deep Visual Learning,* 2019.

A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," *arXiv preprint arXiv:1610.09585,* 2016.

M. Olazaran, "A Sociological Study of the Official History of the Perceptrons Controversy," *Social Studies of Science. 26 (3): 611–659,* 1996.

A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434,* 2015.

N. Radford, "Bayesian Learning for Neural Networks*," Springer-Verlag New York, Inc., Secaucus, NJ, USA, ISBN 0387947248,* 1996.

M. Rahman and D. Davis, "Addressing the class imbalance problem in medical datasets," *Int. J. Mach. Learn. Comput. 3 224–8,* 2013.

P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, et al. “Chexnet: Radiologist level pneumonia detection on chest x-rays with deep learning,” *arXiv preprint arXiv:1711.05225*, 2017.

F. Rosenblatt, “The Perceptron--a perceiving and recognizing automaton,” *report 85-460-1, Cornell Aeronautical Laboratory*, 1957.

S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1600.04747,* 2016.

D. Rumelhart, G. Hinton, R. Williams, "Learning representations by back-propagating errors," *Nature. 323 (6088): 533–536,*1986.

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision, 115(3): 211–252*, 2015.

R. Salakhutdinov, A. Mnih, and G. Hinton, “Restricted Boltzmann machines for collaborative filtering,” *Machine Learning, Proceedings of the Twenty-fourth International Conference, ACM*, 2007.

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training gans,” *arXiv preprint arXiv:1606.03498*, 2016.

B. Settles, “From theories to queries: Active learning in practice,” *Active Learning and Experimental Design Workshop, vol 16, JMLR Proceedings, Sardinia, pp 1–18,* 2010.

J. Shlens, “Notes on kullback-leibler divergence and likelihood,” *CoRR abs/1404.2000,* 2014.

D. Silver, A. Huang, C. J. Maddison, A Guez, et al. “Mastering the game of Go with deep neural networks and tree search,” *Nature, 529(7587):484–489*, 2016.

D. Sivia and J. Skilling, “Data Analysis: A Bayesian Tutorial,” *Oxford University Press; Second edition*, 2006.

N. Srivastava, G Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res., 15(1):1929–1958,* 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *CVPR*, 2015.

S. Tabik, D. Peralta, A. Herrera-Poyatos, F. Herrera, “A snapshot of image pre-processing for convolutional neural networks: Case study of MNIST,” *Int. J. Comput. Intell. Syst. 2017, 10, 555–568*, 2017.

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: closing the gap to human-level performance in face verification," *Proc. Conference on Computer Vision and Pattern Recognition 1701–1708,* 2014.

A. Tang, R. Tam, A. Cadrin-Chênevert, W. Guest, J. Chong, J. Barfett, et al. "Canadian Association of Radiologists White Paper on Artificial Intelligence in Radiology," *Can Assoc Radiol J; 69:120-35,* 2018.

N. Tishby, E. Levin, and S. A. Solla, "Consistent inference of probabilities in layered networks: Predictions and generalization," *In: Proceedings of the International Joint Conference on Neural Networks, Washington DC,* 1989.

D. Tsao, M. Livingstone, "Mechanisms of face perception," *Ann. Rev. Neurosci. 2008;31:411–437,* 2008.

P. Tschandl, C. Rosendahl, H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data,* 2018.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural network," *arXiv preprint arXiv:1601.06759,* 2016.

J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," *Stanford University research report,* 2017.

P. Wang, C. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," *arXiv preprint arXiv:1804.07871,* 2018.

F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365,* 2015.

M. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701,* 2012.