GENERATIVE ADVERSARIAL NETWORKS FOR LUPUS DIAGNOSTICS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Pradeep Periasamy

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL STATEMENT OF COMMITTEE APPROVAL

Dr. Vetria L. Byrd, Chair
Department of Computer Graphics Technology

Dr. Tim McGraw
Department of Computer Graphics Technology

Dr. Yingjie Chen
Department of Computer Graphics Technology

Dr. Paul Parsons
Department of Computer Graphics Technology

Approved by:

Dr. Nicoletta Adamo-Villani
Head of Graduate Program

ACKNOWLEDGMENTS

I would like to take this opportunity to thank Professor Dr.Vetria L.Byrd for graciously hiring me as a Research Assistant and extending the opportunity to work on Lupus Research as part of my thesis. I would like to thank Professors Dr.Tim McGraw, Dr.Yingjie Chen and Dr.Paul Parsons for agreeing to serve my committee and guiding me. I would like to thank American College of Rheumatology for providing access to their open source data-set of Lupus images, The Extreme Science Engineering Discovery Environment (XSEDE) for providing access to the Bridges Cluster, Tommy Sors - Assistant Director of the Purdue Institute of Inflammation, Immunology and Infectious Disease for providing feedback and ideas for the research. I would like to thank Professor Dr.Raghu Pasupathy, Associate Professor, and Professor Dr.Bruce A. Craig, Director of Statistical Consulting, from the Department of Statistics for guiding me with the statistical evaluation of the architecture and comparative studies. I would like to thank Tianyang Hu and Yueyun Zhang from the Department of Statistics for helping with the evaluation metrics. I would like to thank the HyperGAN community, Erik Linder-Norn from Signality, Junho Kim from NCSoft, and Taehoon Kim from Open AI for open-sourcing code.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

n	Number of samples used for training as true bucket
k	Number of samples used for training as false bucket
M	Number of samples generated from GAN
m	Number of samples randomly chosen from M GAN generated samples
t	Epochs for generating M samples
$E(r_i)$	Expected error rate
n_c	Number of observations for control group
n_e	Number of observations for experimental group
α	Ratio between training and testing dataset
δ_{real}	Vector values of the real data
$\delta_{artificial}$	Vector values of the artificially generated data

ABBREVIATIONS

AVI	Automated Visual Intelligence
BGAN	Boundary seeking Generative Adversarial Networks
BiGAN	Bi-Directional Generative Adversarial Networks
CBIR	Content Based Image Retrieval
CCGAN	Context Specific Conditional Generative Adversarial Networks
CGAN	Conditional Generative Adversarial Networks
CNN	Convolutional Neural Networks
COCO	Common Objects in Context
CPU	Central Processing Unit
CT	Computed Tomography
DAC	Digital to Analog Converter
DCGAN	Deep Convolution Generative Adversarial Networks
DCNN	Deep Convolutional Neural Networks
DPN	Dual Path Network
EBGAN	Energy-Based Generative Adversarial Networks
GAN	Generative Adversarial Networks
GMAN	Generative Multi-Adversarial Networks
GPU	Graphics Processing Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
Info-GAN	Interpretable Representation Learning by Information Maximizing Generative Adversarial Networks
JPEG	Joint Photographic Experts Group
LAPGAN	Laplacian Generative Adversarial Networks

LeNet	LeCun Network
LRGAN	Layered Recursive Generative Adversarial Networks
MedGAN	Medical Image Translation using GANs
MNIST	Modified National Institute of Standards and Technology
MRI	Magnetic Resonance Imaging
MTurk	Mechanial Turk
PET	Positron Emission Tomography
PGGAN	Progressive Growing Generative Adversarial Networks
ResNet	Residual Network
ReLu	Rectified Linear Units
SAGAN	Self Attention Generative Adversarial Networks
SLE	Systemic Lupus Erythematosus
VRAM	Video Random Access Memory
VGGNET	Visual Geometry Group Network
YOLO	You Only Look Once

ABSTRACT

Periasamy, Pradeep M.S., Purdue University, December 2019. Generative Adversarial Networks for Lupus Diagnostics. Major Professor: Vetria L. Byrd.

The recent boom of Machine Learning Network Architectures like Generative Adversarial Networks (GAN), Deep Convolution Generative Adversarial Networks (DCGAN), Self Attention Generative Adversarial Networks (SAGAN), Context Conditional Generative Adversarial Networks (CCGAN) and the development of high-performance computing for big data analysis has the potential to be highly beneficial in many domains and fittingly in the early detection of chronic diseases. The clinical heterogeneity of one such chronic auto-immune disease like Systemic Lupus Erythematosus (SLE), also known as Lupus, makes it difficult for medical diagnostics. One major concern is a limited dataset that is available for diagnostics. In this research, we demonstrate the application of Generative Adversarial Networks for data augmentation and improving the error rates of Convolution Neural Networks (CNN). Limited Lupus dataset of 30 typical 'butterfly rash' images is used as a model to decrease the error rates of a widely accepted CNN architecture like Le-Net. For the Lupus dataset, it can be seen that there is a 73.22% decrease in the error rates of Le-Net. Therefore such an approach can be extended to most recent Neural Network classifiers like ResNet. Additionally, a human perceptual study reveals that the artificial images generated from CCGAN are preferred to closely resemble real Lupus images over the artificial images generated from SAGAN and DCGAN by 45 Amazon MTurk participants. These participants are identified as 'healthcare professionals' in the Amazon MTurk platform. This research aims to help reduce the time in detection and treatment of Lupus which usually takes 6 to 9 months from its onset.

CHAPTER 1. INTRODUCTION

This chapter gives the synopsis to the current research and provides a preliminary overview of the importance and the need which resulted in the research question. It also delineates the scope of the study.

1.1 Background

The principal methods of this empirical research are based on Artificial Intelligence, Machine Learning, and Medical Imaging. They lie in the intersection of Generative Adversarial Networks and Medical Imaging where Machine Learning enables researchers to gain a sense of what the image data represents. Our method of using GAN in image classification could potentially impact critical functions across Health-care, Education, Defense and Entertainment industries where there is a need of making key decisions using Big Data but in a shorter time-frame. Systemic Lupus Erythematosus (aka Lupus) has been identified as a model for the proposed framework application of an automated visual intelligence approach to data associated with a chronic disease. Lupus is a chronic autoimmune disease characterized by clinical heterogeneity (Yu, Gershwin, & Chang, 2014), (Marion & Postlethwaite, 2014) which adds an additional level of complexity to the problem and a greater opportunity to advance Lupus research. One of the major concerns that exist in the medical fraternity is that limited availability of annotated images (Roth et al., 2016), (Litjens et al., 2017). The research community tries to overcome this gap by using the traditional approaches like translation, rotation, shearing, flipping and scaling that are commonly practiced within the deep learning community (Krizhevsky et al., 2012). However, the recent advancements in Deep Learning shows that Generative Adversarial Networks (GANs) (Goodfellow et al.,

2014) can be used an approach to increase accuracy in medical image segmentation (Frid-Adar et al., 2018). Drawing parallel from this prior research, GAN can be used to increase the accuracy of the Neural Network Classifiers, using Lupus as a model for Medical Diagnostics.

1.2 Significance

Applications of Generative Adversarial Networks (GANs)(Goodfellow et al., 2014) are evident in research (Emami, Dong, Nejad-Davarani, & Glide-Hurst, 2018) and in the industry (Karras, Aila, Laine, & Lehtinen, 2018). However, the literature shows minimal work has been reported in the use of GANs in image classification specific to Lupus diagnostics. One of the plausible potentials of GAN lies in its application of data augmentation where it can generate realistic looking data that looks authentic to the human eye. GAN has the potential to improve the state-of-the-art image classification neural network. This work focuses on the use of GAN in medical diagnostics. The idea is to use GAN to distinguish dermatological manifestations of Systemic Lupus Erythematosus (SLE) from other diseases that share similar symptoms. Prior knowledge shows that there are over 1 million people in the United States affected by Lupus, with symptoms ranging from mild to lethal. (Marshall, 2002). To summarize, an Automated Visual Intelligence (AVI) approach is introduced where GAN is adapted to generate synthetic images from a small sample dataset which is then used to train the classifier to distinguish cutaneous Lupus from other diseases. A new hybrid neural network architecture is demonstrated for image classification using a small training dataset.

1.3 Research Questions

In this study, simulated Butterfly rash images from normal photographs of patients who have the potential of developing Lupus at a later stage is generated and demonstrated. The goal of this research is to use simulated rash images to

anticipate and inform treatment for Lupus patients. The research questions are as follows.

1. Research Question 1 - Can the accuracy of the Neural Network classifier Le-Net (LeCun et al., 1998) be increased by including GAN generated synthetic images in the training dataset?

2. Research Question 2 - Can Conditional GAN (Mirza & Osindero, 2014) be used to generate simulated images that resemble the cutaneous manifestations of Lupus from normal images without Lupus?

    Research Question 2.1 - Which among the state-of-art GANs is preferred to generate artificial images including Context Conditional GAN (CCGAN)(E. Denton, Gross, & Fergus, 2016), Deep Convolutional GAN (DCGAN) (Radford, Metz, & Chintala, 2016), and Self-Attention GAN (Zhang, Goodfellow, Metaxas, & Odena, 2019), by the Amazon MTurk participants identified as 'healthcare professionals'?

The primary objectives of this research are

1. to test the accuracy of Le-Net classifier with the use of images generated from GAN in the training dataset.

2. to demonstrate the use of Conditional GAN (Mirza & Osindero, 2014) in generating the artificial images with cutaneous manifestations of Lupus from normal face dataset.

3. to study human evaluation of the artificial images with cutaneous manifestations of Lupus generated from the Context-Specific Conditional GAN (CCGAN).

1.4 Assumptions

Although GAN (Goodfellow et al., 2014) is a candidate for the application of data augmentation, it is highly debated in the research community. It can generate realistic looking data that looks authentic to the human eye, however it is not much different from the traditional data augmentations methods like translation, rotation, shearing, flipping and scaling as commonly practiced within the deep learning community (Krizhevsky et al., 2012). Given a training dataset, GAN generates the data, which have the same distribution of the training data with additional random noise.

It is noteworthy to point out the relationship between the various parameters including *'n'*, *'k* , α, *'M'*, *'t'*, $E(r_i)$ and *'i'*. *'n'* is the number of images used for training as true bucket, *'k'* is the number of false images for training, α is the ratio between training and testing dataset, *'M'* is the number of images generated over *'t'* epochs, and the expected error rate over *'i'* iterations is $E(r_i)$. The relationship between these parameters is highly researched and there is not enough clarity on the ideal setting for such context of classification. All the above parameters will be treated as hyper-parameters in the proposed experiment.

1.5 Limitations

The major limitation of the experiment involves using a limited number of training images for classification of Lupus. The dataset for this experiment consists of *'n'* = 30 images from different sources of publicly available images of Lupus related facial lesions, majorly from the image library of American College of Rheumatology (*Image Library*, n.d.). The authenticity of such images is not individually verifiable. Furthermore, for easier implementation, Le-Net classifier (LeCun et al., 1998) will be used for classification, though there are other more accurate classifiers available to experiment. The ideal relationship of the various

hyper-parameters is highly researched and there is no sufficient literature to know the ideal setting specifically catering to Lupus research.

1.6 Delimitations

To narrow down the scope of the study, a binary variant of the Le-Net classifier was adopted rather than the multi-class classifier. This is partly due to the unavailability of the data set pertaining to all the similar diseases. To test our hypothesis a sample Lupus dataset of 30 images was used to test the error rate of the Le-Net Classifier from different sources of publicly available images of Lupus related facial lesions. The classifier is further cross-validated by the MNIST dataset (LeCun, 1998). For easier implementation on CPU, a simpler GAN variant - HyperGAN (HyperGAN-Community, 2016) will be used for generating artificial Lupus images from the sample distribution.

1.7 Definitions

Generative Adversarial Network (GAN) -

> "A new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a mini-max two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 1/2 everywhere. In the case where G and D are defined by multi-layer perceptrons, the entire system can be trained with back propagation (Goodfellow et al., 2014).

Systemic Lupus Erythematosus(SLE) -

> "An autoimmune disease that can affect many organs, including the skin, joints, the central nervous system and the kidneys" (Kaul et al., 2016).

Convolutional Neural Network(CNN) -

> "Convolutional networks combine three architectural ideas to ensure some degree of shift and distortion invariance: local receptive fields, shared weights (or weight replication), and, sometimes, spatial or temporal subsampling. The input plane receives images of characters that are approximately size-normalized and centered. Each unit of a layer receives inputs from a set of units located in a small neighborhood in the previous layer" (LeCun et al., 1998).

Artificial Intelligence -

> "The automation of activities that we associate with human thinking, activities such as decision-making, problem solving, learning" (Bellman, 1978).

1.8 Summary

This section gave an in-depth explanation of the assumptions, limitations, delimitations, definitions, scope, significance, research question, and other background information for the research project. It also addressed the various boundaries that the study is limited by. It also gives a list of defined key terms that have been used in the proposal. The next chapter is aimed at giving a literature review that is relevant to "Generative Adversarial Networks for Lupus Diagnostics".

CHAPTER 2. REVIEW OF RELEVANT LITERATURE

This section provides the literature review of the prior work related to Lupus diagnostics, recent advancements in GAN and its variants, and a review of relevant neural network classifiers.

2.1 Lupus Diagnostics

SLE is a chronic autoimmune disease with heterogeneous presentation (Kaul et al., 2016), that can affect any part of the body. It affects persons of all ages, ethnic groups, and gender (Ferenkeh-Koroma, 2012). SLE symptoms vary widely and often mimic symptoms of other diseases making it difficult to diagnose and treat (Yu et al., 2014). Patients with SLE may present with various systemic manifestations (Hiepe, 2014). Literature shows that the skin is frequently affected among all the other organs affected by Lupus erythematosus (Hiepe, 2014). One visible manifestation includes the malar rash, which is characterized by a rash over the cheeks and nasal bridge on the face and is often referred to as a "butterfly" rash. The location of the rash in relation to facial landmarks and presence of these distinct characteristics in photographs of patients diagnosed with cutaneous erythematosus suggest an ideal starting point for the use of GAN as a diagnostic tool that could be visually verified. Machine learning techniques applied to medical imaging research include: noise reduction (Wolterink, Leiner, Viergever, & Išgum, 2017), evaluation of PET images (Y. Wang et al., 2018), and the generation of CT images from MRI images (Emami et al., 2018) to name a few. However, to the best of the author's knowledge, GAN techniques have not been utilized as a diagnostic tool for determining the presence of cutaneous manifestations of Lupus. The heterogeneous nature of the disease makes it an ideal candidate for GAN applications.

2.2 GAN: State of Art

Generative methods in Machine Learning started to gain attention after the invention of Generative Adversarial Networks by Ian Goodfellow (Goodfellow et al., 2014). Prominent GAN approaches include using Variation Auto-encoders (Pu et al., 2016), auto-regressive models (Akaike, 1969) and Generative Adversarial Networks (Goodfellow et al., 2014). Generative methods are concerned with distributions models where noise is introduced to create variations and at the same time tries to find dependencies between similar data points.

Recent variants of GAN include Deep Convolutional Generative Adversarial Networks (DCGAN) (Radford et al., 2016), Conditional GAN (Mirza & Osindero, 2014), BiGAN (Donahue & Darrell, 2017), Progressive Growing of GANs (PGGAN) (Karras et al., 2018) to name a few. DCGAN replaces deterministic pooling functions layers(such as max-pooling) with convolutions that are stridden (Radford et al., 2016).

PGGAN grows the generator and discriminator progressively starting from lower resolutions and modeling fine details with new layers while training progresses.

Another important variant of GAN is the Laplacian generative adversarial networks (LAPGAN) where the models are primarily targeted to generate up-sampled images in multiple steps (E. L. Denton, Chintala, Fergus, et al., 2015). It is worthy to mention the Generative Multi-adversarial networks (GMAN) (Durugkar, Gemp, & Mahadevan, 2017) which includes multiple discriminators where each generated image from the Generator is being examined by N randomly instantiated replicas of the discriminator. Layered Recursive Generative Adversarial networks (LR-GAN) (Yang, Kannan, Batra, & Parikh, 2017) in a sequential repetitive fashion simulates images where it first sequentially generates a background followed by the foreground which is conditioned on the background. LR-GAN uses a mask and affine transformation that collectively articulates the complete final image. Info-GAN (X. Chen et al., 2016) breaks down the noise

vector into a couple of parts: an incompressible noise z and a latent code c that emphasizes the most important features of the data distribution $G(z,c)$.

Energy-Based Generative Adversarial Networks (EBGAN) (Zhao, Mathieu, & LeCun, 2017) assigns energy magnitude as per the data manifold by using the discriminator as an energy function. The Boundary Seeking GAN (BGAN) (Hjelm et al., 2018) generates images according to the decision boundary in each update of the training process, therefore hoping to match a target distribution at the limit of the better discriminator. Least square GAN (Mao et al., 2017) uses the least-squared loss function for the discriminator.

The recent advancement in GAN involves using a conditional setting for the generation of the fake images also known as CGAN (Mirza & Osindero, 2014). In the common generative model which does not have any restriction, there is very little power over the data that is being generated. In the Conditional GAN (CGAN), the generator is learning to produce more artificial images with specific constraints by using labeled classifications forehand.

Conditional adversarial networks has shown applications in Splenomegaly Segmentation (Huo et al., 2018), generating phantom images that are used in the quantification of Ki67 breast cancer images (Senaras et al., 2018), as well as applications in image processing, graphics and computer vision (Isola, Zhu, Zhou, & Efros, 2017). It is noteworthy to mention about NiftyNet which tries to improve the collaborative efforts of researchers using a deep learning platform for medical image analysis (Gibson et al., 2018).

The conditional setting need not be limited only to the Discriminator but can also be extended to the Generator as well. Such a setting is demonstrated for the convoluted generation of faces by the Conditional generative adversarial nets (Gauthier, 2014). It is shown that "Identity-Preserving" optimization approach preserves the persons identities with very high facial recognition score of 82.9% by using Conditional GAN (Antipov, Baccouche, & Dugelay, 2017). Conditional GAN finds its application in the synthesis of high-resolution images from the semantic

label maps (T.-C. Wang et al., 2018). It can be seen that a novel stable adversarial learning goal along with multi-scale generator and discriminator architectures, can generate highly photo-realistic images. Such synthesized photo-realistic images are more appealing than those generated by previous methods like Image-to-Image translation (Isola et al., 2017) and Cascading refinement networks (Chen & Koltun, 2017). In the adverse scenario where there isn‘t enough labeled data-set, Image Translation problems are addressed in an unsupervised setting (Liu, Breuel, & Kautz, 2017). In such a setting, the marginal distributions in individual domains are used to determine the joint distribution of images in similar domains (Liu et al., 2017). In a similar setup, a mapping G: X $\rightarrow$ Y is made so that the data determined from the pool of images from G(X) is indiscernible from the other distribution Y using an adversarial loss. An inverse mapping F: Y $\rightarrow$ X is injected by a cycle consistency loss to enforce F(G(X)) = X (and vice-versa) (Zhu, Park, Isola, & Efros, 2017). Context-specific conditional GAN (CCGAN) is used in an in-painting method for filling up the pixel information in the holes within pictures which hinders labeling and the architecture. Such a contextual conditional GAN is used to augment the pixel information and then tested for classification accuracy (E. Denton et al., 2016).

Owing to the high accomplishment of Generative Adversarial Networks (GANs), Conditional GANs provide a more meaningful directive to the data generation process by limiting certain augmentative information. Generating image data from textual inputs has been a primary focus area of research for years. However, the images that are generated from the prior work does not reflect the semantic meaning of input given as it is typically vague. Stack-GAN tries to address this problem where photo-realistic images are generated conditioned on textual descriptions (Zhang et al., 2017). Using GAN in medical image analysis is explored in Liver Lesion Classification for data augmentation that improves the performance of Convolutional Neural Networks (Frid-Adar et al., 2018).

An important metric to measure the success of GAN is the Inception Score (Salimans et al., 2016) which is widely accepted in the GAN community. This is further improved by another metric proposed in the work done by Heusal et.al (Heusel, Ramsauer, Unterthiner, Nessler, & Hochreiter, 2017) known as Fréchet Inception Distance. According to these metrics, Big GAN (Brock, Donahue, & Simonyan, 2019), and Self-Attention GAN (SAGAN) (Zhang et al., 2019) seems to perform well on ImageNet Database (Deng et al., 2009). The next section provides details about the recent state of the art Convolutional Neural Networks (LeCun et al., 1998).

2.3 Convolutional Neural Network Classifiers - State of the Art

The first well-known Convolutional Neural Network classifier Le-Net (LeCun et al., 1998) was used for document recognition to identify characters and read documents.

It uses the MNIST database to classify handwritten digits. Convolutional Neural Networks have a significant difference in the architecture when it is compared to traditional Neural Networks. Regular Neural Networks consists of several hidden layers. Through these hidden layers, the inputs are transformed. Every layer is densely interconnected among every other layer. Finally, the output layer represents the predictions. Convolutional Neural Networks work in a different way. It has three major dimensions that include depth, width, and height. Alternatively, all the neurons in any layer do not connect with each and every neuron in the next layer but are only limited to a small segment. Finally, the outcome is reduced to a one-dimensional quantity for easy manipulation. The Convolution or the filter layers is an important aspect of any CNN architecture.

The ImageNet Database by Standford Vision Lab gave researchers access to millions of images and thousands of labels of the data-sets which is often used for training and testing. It is noteworthy to mention AlexNet (Krizhevsky et al., 2012)

which is similar to Le-Net but marked by a non-linear approach with Rectified Linear Units (ReLu) activation and max-pooling and is illustrated in Figure 2.1.

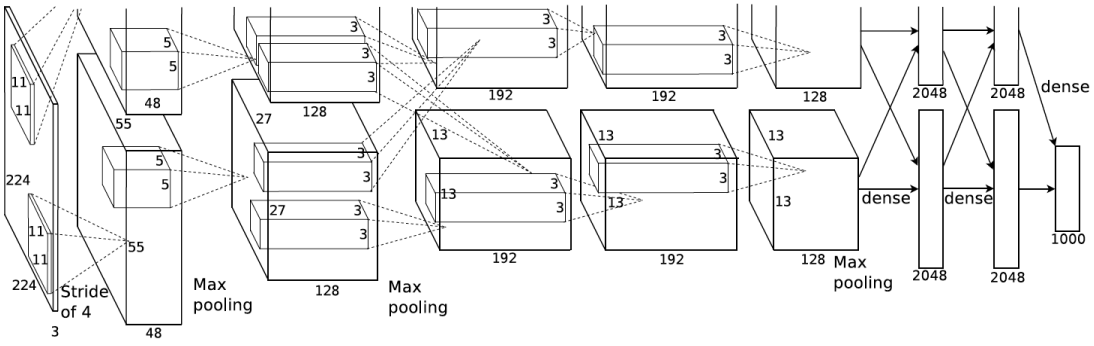


Figure 2.1. AlexNet Architecture, adopted from AlexNet (Krizhevsky et al., 2012). Reprinted with permission from"ImageNet Classification with Deep Convolutional Neural Networks" by Alex Krizhevsky et.al, 2012. Advances in Neural Information Processing Systems 25 (NIPS2012), 1097-1105. 2012 by Alex Krizhevsky.

AlexNet proved that using a GPU implementation rather than CPU will yield approximately 50 times faster results. The AlexNet is arguably one of the most influential implementations of Deep CNNs till today. It was the first Deep CNN that managed to beat more traditional object recognition approaches in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Moreover, the AlexNet proved the viability of DCNN approaches for object recognition tasks. AlexNet is not much different from the LeNet, as it also consists of only the input layer, a few convolutional layers with occasional pooling afterward, as well as some fully-connected layers right before the output layer. However, the AlexNet has more layers and neurons per layer and it also uses different hyperparameters.

VGGNet (Brock et al., 2019) used the concept of progressively training deeper networks in a sequence. VGGNet scored second place in the ILSVRC and influenced the deep learning scene in an important way, as they showed that using a

deeper architecture does generally lead to better results. The VGGNet that was submitted for the ILSVRC contained 19 parameterized hidden layers, which was much more than what previous architectures had used. Apart from its size, the VGGNet was very simple. It only consisted of convolutional layers with a 3 x 3 receptive field, which is the smallest size that can differentiate basic directions, as well as 2 x 2 max-pooling layers, and three fully-connected layers at the end.

GoogleNet (Szegedy et al., 2015), in order to have more distinctive features captured in the feature extraction maps, used parallel passages with receptive fields of various sizes. GoogleNet used 1x1 convolutions for dimension reductions before expensive convolutions and finally used filter concatenations to get back to its initial state and is illustrated in Figure 2.2.

The authors also implemented the so-called Inception Modules, which enable a network to recognize patterns of different sizes within the same layer. In order to do so, the inception module performs several convolutions with different receptive fields in parallel and combines the results by merging the depth slices of the different filters into one single layer. The final GoogLeNet consisted of several such inception modules stacked on top of each other with occasional pooling layers in between, a few additional convolutional layers at the beginning of the network and a few fully-connected layers right before the output layer. GoogLeNet also contained additional output layers closer to the middle of the network and their outputs were combined with the output of the final layer of the network to obtain the total prediction. This had some minor influence on the overall result but was mainly intended to accelerate the training of earlier layers. Inception-v4 is the fourth iteration of the GoogLeNet and consisted of many more layers than the original version. During the continuous improvements on the inception architectures, the inception modules, have been vastly improved as well and the Inception-v4 uses three different kinds of inception modules. In addition to the Inception-v4, the corresponding paper also introduced a new type of network, named Inception-Res-Net, which is a combination of an inception network and a ResNet,

by combining the inception module and residual connection This makes the network even more efficient, leading to much lower training times compared to a similarly complex inception network. Both of these network architectures are hundreds of layers deep and contain a wide variety of layers, inception modules, and residual blocks.

The most recent Neural Network classifier is ResNet (He, Zhang, Ren, & Sun, 2016) which has one of the lowest error rates at a minimal 3.57 percent trained on ImageNet database. ResNets, convolutional layers are divided into Residual Blocks and for each block, a Residual Connection is added, which is bypassing the corresponding block. The output of input was then forwarded by the residual connection. By adding these residual connections, the result of a training step can be back-propagated to the earlier layers directly, without any interference from subsequent layers. Therefore, residual connections enable the training of even deeper networks. He et al. won both the The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) localization and classification contests, and the COCO detection and segmentation contests in 2015. They also managed to improve on the previous error rates by a big margin. The residual block is merged by summation with the original. The Res-Net version that was submitted to these contests was the Res-Net101, which consists of 101 parametrized layers. These 101 layers consist of an initial 77 convolutional layer with a 22 stride, 33 residual building blocks with decreasing output size and increasing depth and a final 1000 neuron fully connected layer. The Res-Net101 also includes one max pooling layer apart from the average pooling layer before the final connected layer. The first convolutional layer of each group of blocks uses stride two in order to achieve the output size reduction.

Dense-Net is a generic successor to ResNet and achieves 3.46% error on CIFAR-10 and 17.18 on C-100. Dense-Nets (Huang, Liu, Van Der Maaten, & Weinberger, 2017) combines the two outputs by depth-wise filter concatenation, as performed in inception modules. Furthermore, Dense-Nets are adding one such connection from each layer to all subsequent ones with matching input sizes. By

doing so, the learned features of a layer can be reused by any of the following layers. Therefore, later layers need to produce much fewer feature maps, resulting in less complex architectures with fewer parameters. Since the width and height of layers in the CNNs are gradually decreasing, connecting all compatible layers is dividing the network into Dense Blocks. Between these blocks, pooling layers are used to alter the sizes accordingly. These layers are referred to as Transition Layers. Due to the high layer inter connectivity, Dense-Nets are easy to train and naturally scale well with increasing depth and amount of parameters. The design of Dense-Nets encourages the learning of new features, while Res-Net architectures are leading to increased feature re-use. Since both architectures have advantages over each other, (Chen et al.) combined the two architectures into a Dual Path Network (DPN), with which they won first place in the 2017 ILSVRC localization challenge and finished top three in both classification and detection. In order to combine the networks, the output of a layer is split and one part is combined with a residual connection, whereas the other is forwarded to all subsequent layers, as performed in DenseNets (Y. Chen et al., 2017). Shake-shake (Gastaldi, 2017), Shake-drop (Yamada, Iwamura, Akiba, & Kise, 2018) and possibly other variants are regularization techniques which can be used any ResNet-like architectures, and achieves 2.86/2.31% error on C-10 and 15.85/12.19% on C-100. These techniques work only on multi-branch architectures, even though they are not strictly architectures in themselves. Efficient Neural Architecture Search uses reinforcement learning to search for architectures and finds a network which achieves 2.89% error on C-10, using the cutout regularization technique (Pham, Guan, Zoph, Le, & Dean, 2018). One of the most relevant work in bio-medical image segmentation is U-Net architecture which uses elastic deformation for data augmentation (Ronneberger, Fischer, & Brox, 2015).

2.4 Summary

Though there has been significant progress in neural network classification for large data-sets, little has been achieved if the data-set is very limited in size, fittingly in the case of Lupus. Therefore, using data augmentation becomes a viable step. Therefore, the major focus was to test the efficiency of Le-Net classifier with the use of GAN generated images in the training data-set. Furthermore, the research included the perceptual study of the qualitative evaluation of the artificial images generated from the CCGAN.

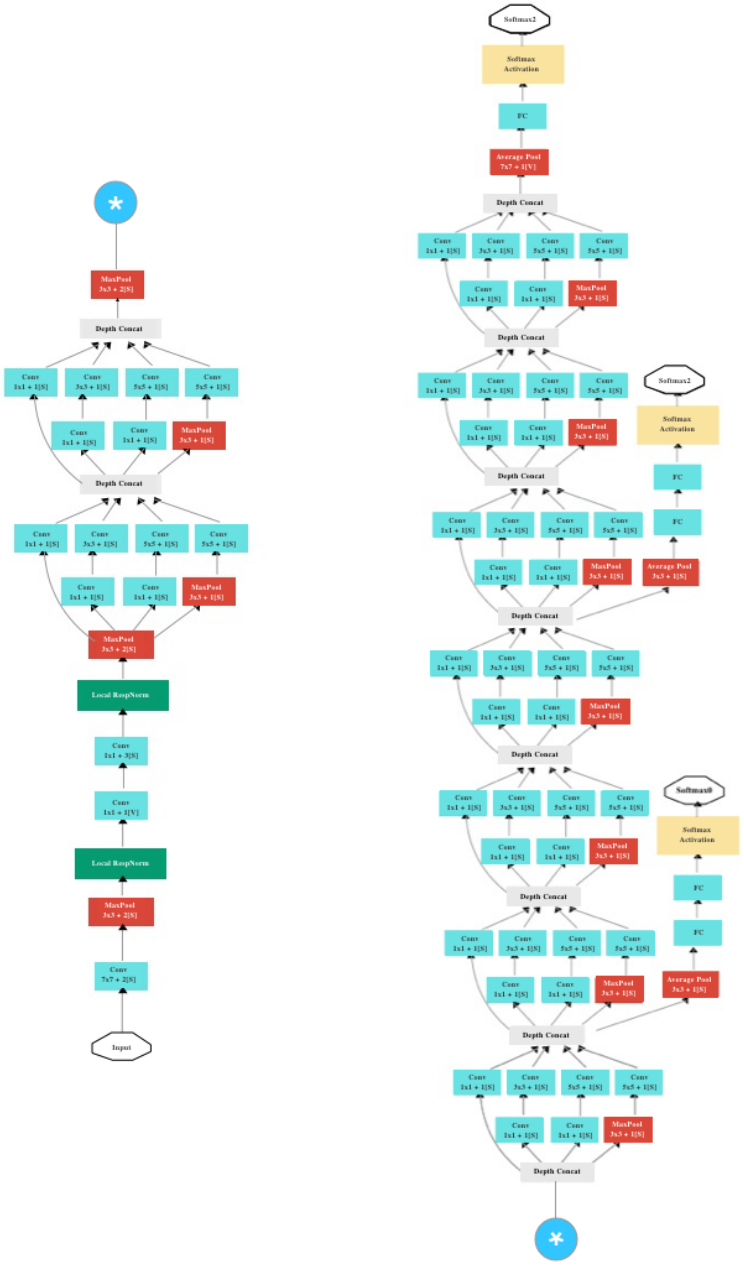


Figure 2.2. GoogleNet Architecture, adopted from GoogleNet (Szegedy et al., 2015). Architecture continues from left to right. Reprinted with permission from "Going Deeper with Convolutions" by Szegedy Christian et.al, 2014. IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015), 1-9. 2015 by Szegedy Christian.

CHAPTER 3. FRAMEWORK AND METHODOLOGY

This chapter provides details about the frameworks and methods that were used in the research study including the sample, the data collection, the variables that were tested, and the data analysis.

3.1 AVI Architecture

The architecture of the network for this research is illustrated in Figure 3.1. It consists of a combination of two deep neural network frameworks combined to form a neural network. Figure 3.1a demonstrates GAN architecture. The neural network consists of a Generator G and a Discriminator D which is trained from images of chosen resolution 32 x 32 (Goodfellow et al., 2014). Figure 3.1b is the Le-Net Classifier (LeCun et al., 1998) which takes the generated images from the GAN as input in a predetermined timed fashion and trains the Le-Net classifier over several epochs.

For easier implementation HyperGAN (HyperGAN-Community, 2016) was used for the GAN implementation. HyperGAN, like any other GAN, consists of the Generator, Discriminator, Encoder, Loss function, and the Trainer. The default configuration of HyperGAN was used for training where it uses Least Squares Loss function for both the discriminator and the generator given by the following equations where a,b,c are hyper-parameters and δ_{real}, $\delta_{artificial}$ are vector values of real and artificially generated data respectively.

$$Disc_{loss} = [\delta_{real} - b]^2 - [\delta_{artificial} - a]^2$$

$$Gen_{loss} = [\delta_{artificial} - c]^2$$

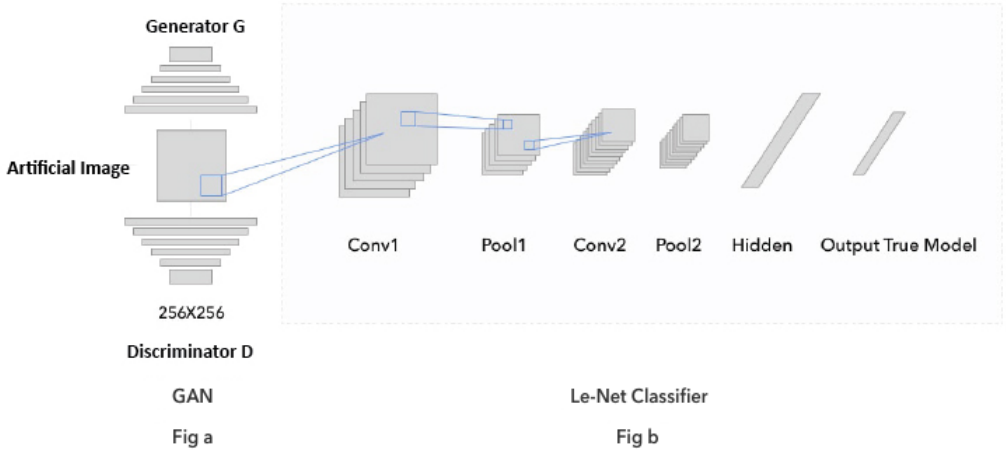


Figure 3.1. AVI Architecture, Adopted from GAN (Goodfellow et al., 2014) and Le-Net (LeCun et al., 1998). Reprinted with permission from Gradient based learning applied to document recognition by Yann Lecun et.al, 1998. Proceedings of the IEEE,86(11),2278-2324. doi: 10.1109/5.726791 by IEEE.

The Le-Net framework consists of convolution filers which are then followed by the ReLU activation. The ReLU activation is then succeeded by the 2x2 max pooling. The max-pooling layer has a stride of 2. This is again followed by a larger convolution filter that is succeeded by another ReLU activation and max-pooling layers. The output of the last max pooling layer is then standardized to a single vector. The single vector is used in a hidden layer. The last step involves using the soft-max or the normalized exponential function to classify and output the true model (p =1).

The Generator G constantly tends to deceive the Discriminator D whereas the Discriminator tries not to be deceived. After training over a few epochs the generated image which is classified as real by the Discriminator is used as an input to train the output true model. Similarly, another Output False Model is trained from non-Lupus sample images. The model, when tested with probability prediction function from Keras (HyperGAN-Community, 2016), gives a labeled classifier.

3.2 Overview of CCGAN Architecture

The primary objective of a Generative Adversarial Network is to generate synthetic data using the sample distribution provided to it. To accomplish this goal, it has two major components. Firstly, a Generator G that will be trained to produce synthetic images by introducing random noise in the distribution. Secondly, a Discriminator D that is trained to classify the generated images as real or fake. The loss functions serve as a two-player game where one player is competing against the other to win the game. More formally, let $X = x^1, ..., x^n$ be a dataset of images of d dimensions. Let D describe a discriminative function that will take an image x $\in R^{d}$ as an input. The Discriminator D then gives a probability output of whether the input x is real or not. The Generator G denotes a generative function that uses a random vector $z \in R^{z}$ as an input. This input is derived from a noisy distribution *pNoise*. The Generator gives a synthesized image $\hat{x} = G(z) \in R^{d}$ as output. Ideally,

D(x) = 1 when x ∈ X and D(x) = 0 when x was generated from G. The GAN objective is given by:

$$\min_G \max_D \; E_{x\sim X}[log(D(x))] + E_{z\sim pNoise}[log(1 - D(G(z)))]$$

The conditional generative adversarial network (Mirza & Osindero, 2014) is a derivation of the GAN. A vector y which could be the label of a dataset can be inputted as additional information to derive a conditional setting. The conditional GAN objective is given by:

$$\min_G \max_D \; E_{x,y\sim X}[log(D(x,y))] + E_{z\sim pNoise}[log(1 - D(G(z,y),x))]$$

The context-conditional generative adversarial networks (CCGANs) are conditional GANs where the generator is trained to fill in a missing image patch and the generator and discriminator are trained from the surrounding pixels. In particular, the generator G receives as input an image with a randomly masked out patch. The generator outputs an entire image. The missing patch is filled from the generated output and then the completed image is passed into D. The completed image is passed into D rather than the context and the patch as two separate inputs. This is done to prevent D from simply learning to identify discontinuities along the edge of the missing patch.

More formally, let m ∈ R ^{d} denote to a binary mask that will be used to drop out a specified portion of an image. The generator receives as input m ⊙ x where ⊙ denotes element-wise multiplication. The generator outputs x_G = G(m ⊙ x, z) R ^{d} and the inpainted image x_I is given by:

$$x_I = (1 - m) \odot x_G + m \odot x$$

The CC-GAN objective is given by:

$$\min_{G} \max_{D} \; E_{x\sim X}[log(D(x))] + E_{x\sim X, m\sim M}[log(1 - D(x_{\mathrm{I}}))]$$

3.3 Hypotheses

This section describes two hypotheses. Section 3.3.1 describes the quantitative evaluation of AVI architecture introduced to address the use of GAN for data augmentation. Section 3.3.2 describes the perceptual study of CCGAN using Lupus dataset for the qualitative evaluation.

3.3.1 AVI Hypothesis Testing

H_0: There is no statistically significant difference in the classification error rate of the Le-Net Classifier by including the GAN generated synthetic images in the training dataset.
H_α: There is a statistically significant difference in the classification error rate of the Le-Net Classifier by including the GAN generated synthetic images in the training dataset.

3.3.2 Hypothesis Testing of the Perceptual Study on CCGAN

H_0: There is no statistically significant difference in the preference of the artificial images generated by CCGAN approach, over other state-of-art baselines by healthcare professionals for Lupus Diagnostics.

H_α: There is statistically significant difference in the preference of the artificial images generated by CCGAN approach, over other state-of-art baselines by healthcare professionals for Lupus Diagnostics.

3.4 AVI Experiments

The hypothesis was that by including artificially generated sample images from GAN in the training dataset, the accuracy of the classifier should increase. To test this hypothesis a sample Lupus dataset was used to test the error rate of the Le-Net Classifier. The classifier was validated by the MNIST dataset.

3.4.1 Variables

The number of artificial images m generated by GAN is the independent variable and the corresponding Expected Error Rate $E(r_i)$ of the Neural Network classifier which is approximated over 100 iterations is the dependent variable for the null hypothesis to be validated. It is noteworthy to call-out that the relationship between the various parameters including 'n' the number of images used for training as true bucket, 'k' the number of false images for training, α the ratio between training and testing dataset, 'M' the number of images generated over 't' epochs, and the expected error rate $E(r_i)$ over 'i' iterations is highly researched and there is not enough clarity on the ideal setting for such context of classification. In order to narrow the scope of the research, all other variables excluding m and $E(r_i)$ were treated as control variables or hyper-parameters in Machine Learning terminology.

3.4.2 AVI Experiment on MNIST

The MNIST database (LeCun, 1998) is a database that consists of images of handwritten digits. These handwritten digits are commonly used for training by neural network classifiers primarily due to its small size and less training time. The Deep Learning model as discussed earlier has two training sets labeled as "Output True" and "Output False". The MNIST dataset modified as JPEGs by Kaggle (*Kaggle: Digit Recognizer*, n.d.) was used for the purpose of Image generation using GAN. The total number of observations of the MNIST dataset was 43,510. Each

digit had a training sample size of 4,351(N). However, only a binary classification of digit '3' was used for simplified binomial classification. The classifier identified number '3' as true and all other observations as false. The algorithm randomly chose *'n'* observations from the 'N' total number of observations. It then chose nα observations and trained GAN to generate 'M' images generated between 50 and 400 epochs which are hyper-parameters. α belongs to real numbers between 0 and 1.

The algorithm randomly chose *'m'* observations from 'M' total number of generated observations. The Le-Net classifier (LeCun et al., 1998) is trained with *'m+nα'* images first and then with just $n\alpha$ images as True. It had *'k'* equally distributed digits randomly picked from MNIST dataset apart from number 3 as False. The accuracy of both the trained models, one with the m GAN generated images and the other without the m images was tested with *n(1-α)* observations. The experiment was repeated *'i'* times and the expected error rate $E(r_i)$ of all the iterations will be reported. The experiment will be carried out for different values of n where m is the independent variable and $E(r_i)$ is the dependent variable and the null hypothesis was validated.

3.4.3 AVI Experiment on Lupus dataset

The dataset for this experiment consists of *'n'* = 30 images from different sources of publicly available images of Lupus related facial lesions, majorly from the image library of American College of Rheumatology (*Image Library*, n.d.). Similar to the previous experiment on the MNIST dataset, n number of observations were chosen at random from this pool of images. The dataset used for training of AVI had k random images from the UK Bench dataset (Nister & Stewenius, 2006). These k images were used for training the classifier as false whereas $n\alpha$ images of Lupus were used for training the classifier as true. The $n\alpha$ images were used to train the GAN to generate M images trained between 50 and 400 epochs which are hyper-parameters. From this image pool, m images were chosen at random. The

accuracy of both the trained models, one with the m GAN generated images and the other without the m images was tested with $n(1-\alpha)$ observations. The experiment was repeated *'i'* times and the expected error rate $E(r_i)$ of all the iterations were reported. The experiment was carried out for different values of n where m is the independent variable and $E(r_i)$ is the dependent variable and the null hypothesis was validated.

The causal effect of the *'m'* images on the accuracy of the Le-Net Classifier was studied in a similar setting to the previous experiment with the only difference being working with a different dataset.

3.5 Perceptual Study on the quality of CCGAN using Lupus dataset

Human observation plays an important role in the reliability of the quality of the artificial images generated from GANs. The goal of this study was to compare the performance metrics of CCGAN with other baselines on Lupus dataset. Therefore the idea is to use the Amazon Mechanical Turk platform for the Human Intelligence Task. The experiment used pair-wise A/B tests implemented and demonstrated in the prior work (Chen & Koltun, 2017). The MTurk participants were shown around 30 pairs of images which were randomly pooled from the equally distributed database of different baselines that will be compared with CCGAN for the same label of Lupus. There were sentinel pairs to make sure that the workers were not falsely logging their entries without context. The workers were asked to select the image that closely resembled Lupus in each pair. The images were all shown at the same resolution (256 X 256). An example of the A/B tests that were conducted on Amazon Mturk is illustrated in Figure 3.2.

The position of the image in the left-right sequence was randomly chosen. The participants were allowed to choose their selection in unlimited time manner so that they can carefully examine the pattern of the simulated images. However, the time taken to complete all the comparisons was measured to know how quickly each

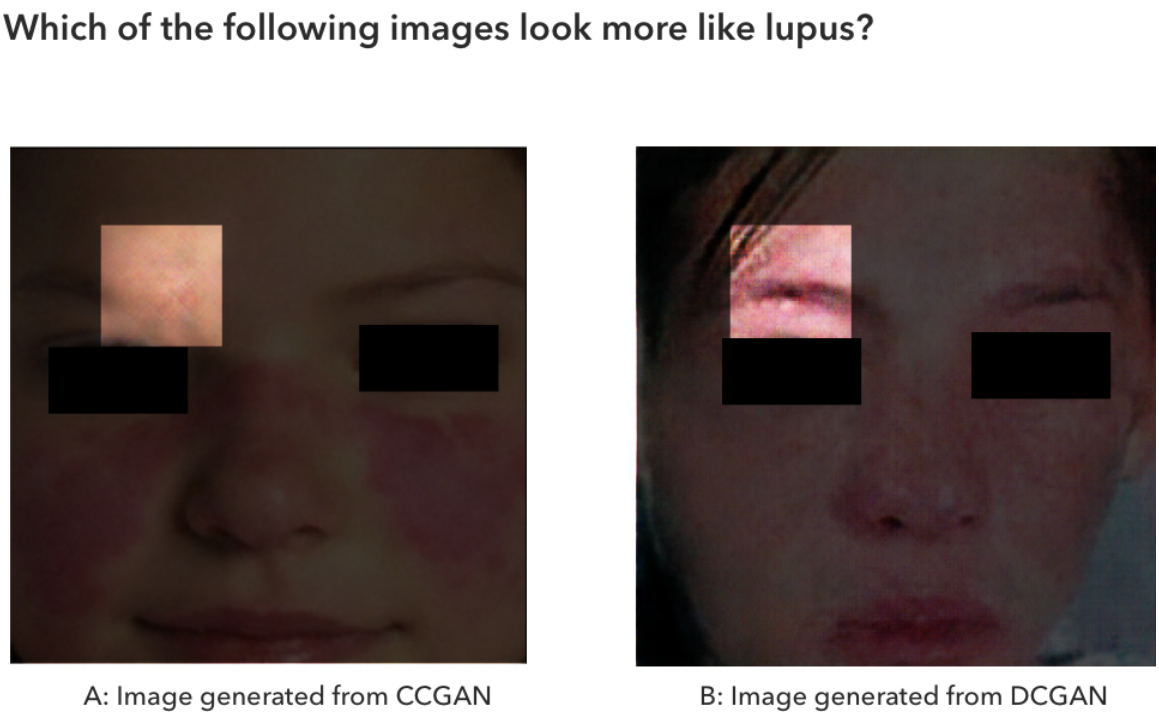


Figure 3.2. Sample Question from the Perceptual Study.

Table 3.1.

Hyper parameters used for training of CCGAN, DCGAN and SAGAN

Hyper-parameter	*Value*
Images fed for generation	30 (Lupus Images)
Learning Rate	0.0002
Batch size	1
Dimension of Noise Vector	100
Image Size	256
Adam Optimizer Decay	0.5
Size of Random Mask for CCGAN	64

architecture is preferred over the other. It is noteworthy to mention that the hyperparameter tuning was made the exact same for all the architecture to ensure minimum confounding variables as summarized in Table 3.1. The hardware configurations used for training CCGAN, SAGAN and DCGAN is tabulated in Table 3.2.

3.5.1 Population

The population of this study was pathologists, general physicians, radiologists, and health-care professionals.

Table 3.2.

GPU Configurations used for Training CCGAN, SAGAN and DCGAN

Manufacturer	NVIDIA
GPU Name	NVIDIA GeForce GTX 1080 Ti
Chip Type	GeForce GTX 1080 Ti
Digital To Analog (DAC) Type	Integrated RAMDAC
Total Memory	19292 Mega Bytes
Display Memory (VRAM)	11127 Mega Bytes
Shared Memory	8165 Mega Bytes

3.5.2 Sample

The sample of the population was the healthcare professionals from Amazon Mechanical Turk aka MTurk (*Amazon MTurk*, n.d.). Amazon Mechanical Turk is a marketplace where human intelligence is required for classification and other purposes. The MTurk service empowers researchers all around the world to remotely conduct research studies. Participants were pre-recruited for such online studies by Amazon and the rewards are determined by the investigators based on the length and duration of the survey. The criterion for the recruitment was broadly classified as 'Employers in Healthcare'using the Premium recruiting options by Amazon MTurk. The sample size was 45.

3.5.3 Variables and Statistical Analysis

The preference of the images generated from CCGAN is compared with other state-of-art baselines like DCGAN (Radford et al., 2016) and Self-Attention GAN (Zhang et al., 2019). The Bradley-Terry model (Bradley & Terry, 1952) is used for the statistical analysis to answer this research question. It is used to find the probabilities of every item being chosen over another when compared with other items in the set, based on repeated pairwise comparisons between the items within the set (Bradley & Terry, 1952).

3.6 Perceptual Study to measure absolute realism

One of the major shortcomings of the above-proposed method is that it does not reveal how realistic experts think the images are in an absolute sense. In order to overcome this shortfall, another perceptual study was done where the experts were asked to rate the absolute realism of the synthetic images generated from CCGAN, SAGAN, and DCGAN. All the images were randomized to ensure a fair comparison. A Likert scale of 4 values namely "Strongly Agree, Agree, Disagree, Strongly Disagree" was used to find the absolute realism of the images. Similar to the previous study, the survey was conducted using the Amazon MTurk participants where 26 participants were asked to rate 10 synthetic images each from CCGAN, SAGAN and DCGAN. The participants identified as "Healthcare Professionals" had unlimited time to make the judgment. Therefore 780 judgments were made on the whole. An example of the sample study conducted is illustrated in Figure 3.3.

As it can be seen in Figure 3.3, random masked patches were chosen for comparison in order to ensure fair judgement between CCGAN and other baselines as CCGAN in-paints missing pixel information from the surrounding pixel.

One of the other shortcomings of the previous study was that the assumption of the healthcare professionals from Amazon MTurk platform to have a fair knowledge about Lupus. However, in order to mitigate the assumption, all the

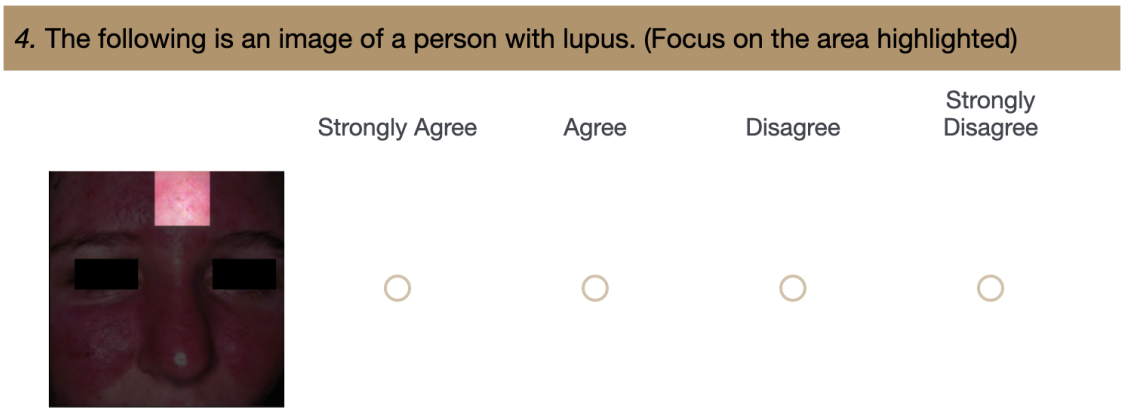


Figure 3.3. Sample Question from the Perceptual Study to measure absolute realism.

participants were given a short brief about Lupus and were shown real images of Lupus before starting the experiment. The brief and images shown are illustrated in Figure 3.4.

3.6.1 Population

Similar to the previous perceptual study, the population of the additional perceptual study was pathologists, general physicians, radiologists, and health-care professionals.

3.6.2 Sample

Similar to the previous perceptual study, the sample of the population was the healthcare professionals from Amazon Mechanical Turk (*Amazon MTurk*, n.d.). The criterion for the recruitment was broadly classified as 'Employers in Healthcare'using the Premium recruiting options by Amazon MTurk. The sample size was 26.

3.6.3 Variables and Statistical Analysis

The preference of the images generated from CCGAN is compared with other state-of-art baselines like DCGAN (Radford et al., 2016) and Self-Attention GAN (Zhang et al., 2019). The Likert scale gives absolute values on realism and implicit comparison is made between the architectures.

3.7 Summary

This chapter provided details about the frameworks and methods that were used in the research study including the sample, the data collection, the variables and the data analysis.

Figure 3.4. Introductory brief about Lupus (From Online Survey).

CHAPTER 4. RESULTS

This chapter explains the findings from the experiments and the analysis of the experiments conducted. This chapter has two sections. The first section explains the findings of Research Question 1 and the second about Research Question 2 correspondingly.

4.1 Findings from Research Question 1

The first research question that was put forth was whether the accuracy of a Network classifier like Le-Net (LeCun et al., 1998) be increased by including GAN generated synthetic images in the training dataset.

To test our hypothesis a sample Lupus dataset was used to test the error rate of the Le-Net Classifier. The classifier is validated by the MNIST dataset.

4.1.1 Experiment on MNIST

The MNIST database (LeCun, 1998) is a large database of handwritten digits that is commonly used for training by neural network classifiers. The Deep Learning model as discussed earlier has two training sets labeled as "Output True" and "Output False". The MNIST dataset modified as JPEGs by Kaggle (*Kaggle: Digit Recognizer*, n.d.) was used for the purpose of Image generation using GAN. The total number of observations of the MNIST dataset used was 43,510. Each digit had a training sample size of 4,351(N). However, only a binary classification of digit '3' was used for simplified calculation. The classifier identifies number 3 as true and all other observations as false. The algorithm randomly chooses *'n'* observations from the 'N' total number of observations. It then chooses nα observations and

Figure 4.1. GAN generated images from MNIST

trains GAN to generate 'M' images where α belongs to real numbers between 0 and 1. The sample of the synthetic images is shown in Figure 4.1. The algorithm randomly chooses *'m'* observations from 'M' total number of generated observations. The Le-Net classifier (LeCun et al., 1998) is trained with *'m+nα'* images first and then with just $n\alpha$ images as True. It had *'k'* equally distributed digits randomly picked from MNIST dataset apart from number 3 as False. The accuracy of both the trained models,one with the m GAN generated images and the other without the m images was tested with *n(1-α)* observations. The experiment is repeated *'i'* times and the expected error rate $E(r_i)$ of all the iterations was reported. The experiment was done for different values of n where m is the independent variable and $E(r_i)$ is the dependent variable. While training the Le-Net classifier, internal validation is done where 75% of the training dataset is actually used for training and the remaining 25% of the dataset is used for validation. The training loss, training accuracy, validation loss and validation accuracy of one of the iterations

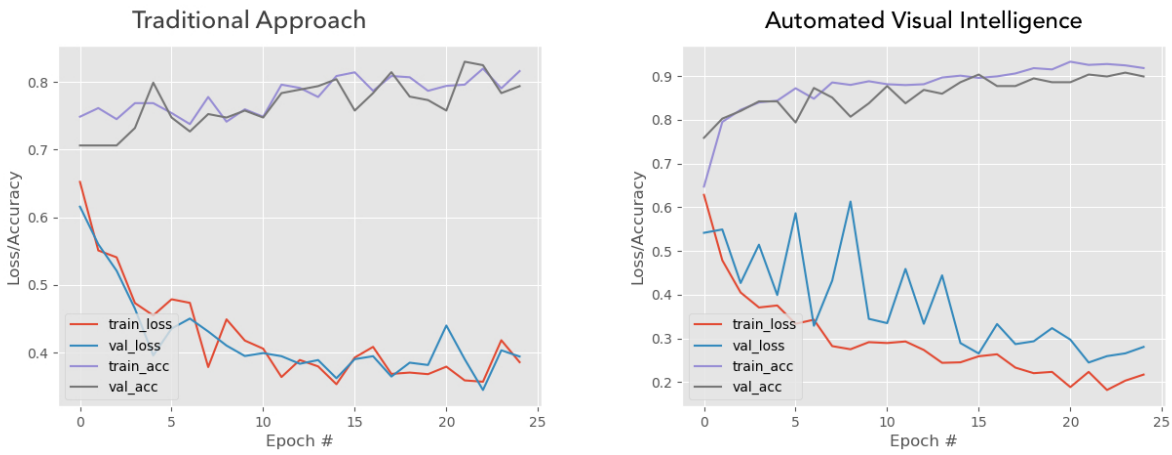


Figure 4.2. Training Accuracy and Loss, Validation Accuracy and Loss

where N = 4351, M=1000, n=250,$\alpha = 0.8$, $m = 250$, $k = 450$ is shown the Figure 4.2 comparing our approach with the traditional approach.

4.1.2 Experiment on Lupus Dataset

The dataset for this experiment consisted of *'n'* = 30 images from different sources of publicly available images of Lupus related facial lesions. The data set used for training of AVI had *'k'* random images from the UK Bench dataset (Nister & Stewenius, 2006). These *'k'* images were used for training the classifier as false where as *'nα'* images of Lupus was used for training the classifier as true. The *'nα'* images were used to train the GAN to generate M images from which *'m'* images are chosen at random. A handful of the images generated are shown in Figure 4.3.

A two-step process was applied to the dataset. First, the Le-Net classifier was trained with $n\alpha$, excluding the m GAN images and the corresponding error rate was recorded for the *n(1-α)* validation images. Second, the Le-Net classifier is trained with *'m + nα'* images and the corresponding error rate was observed for the same *n(1-α)* validation images. The experiment is repeated for i iterations to draw inferences.

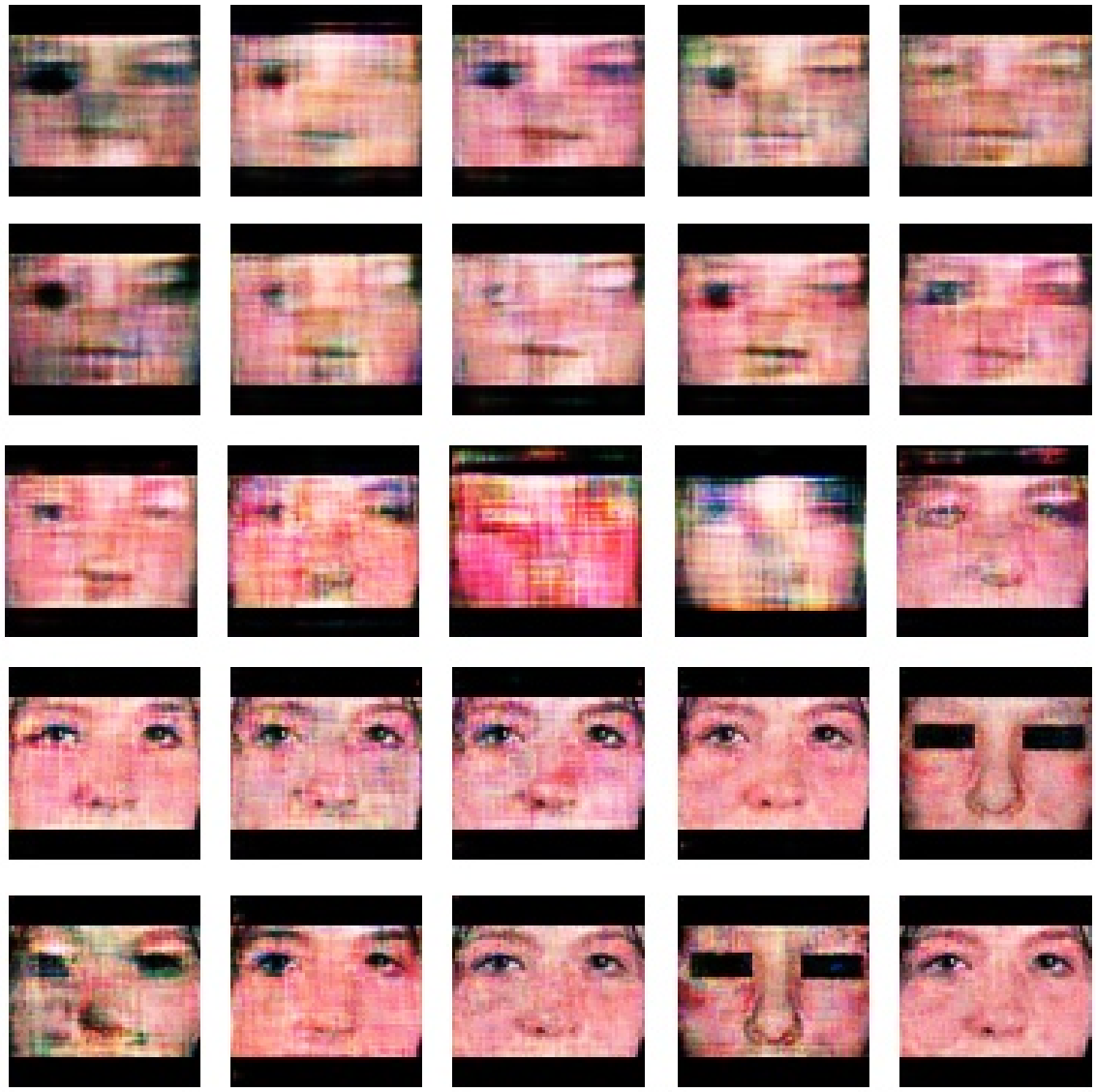


Figure 4.3. GAN generated images from Lupus Dataset

4.1.3 Analysis of Results

The experiment was repeated with various parameters and the corresponding expected error rates observed are summarized in Table 4.1.

The expected error rates are plotted in Figures 4.5, 4.6, 4.7, and 4.8.

Figure 4.4. Classification Sample

Table 4.1.

Summary of Expected Error Rates Comparison between AVI and Traditional Approach for different Parameters

n	α	k	m	i	Le-Net $E(r_i)$	AVI $E(r_i)$
30*	0.8	450	30	100	0.408	0.109
50	0.8	450	50	300	0.956	0.706
100	0.8	450	100	300	0.641	0.540
200	0.8	450	200	300	0.413	0.402

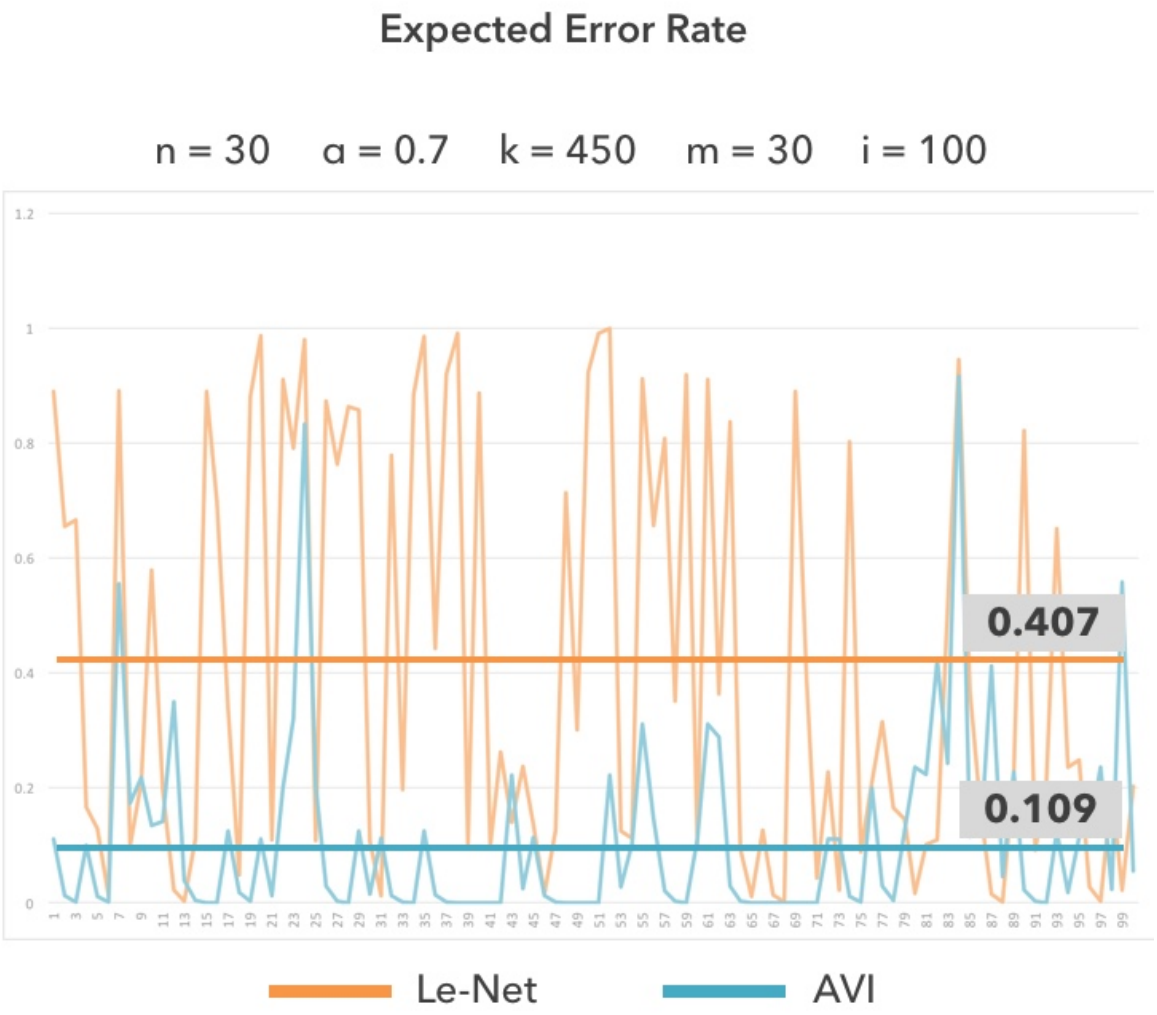


Figure 4.5. Expected error rate comparing Le-Net and AVI for n=30 (Lupus Dataset)

For the Lupus experiment, the data shows there is a 73.22% decrease in the error rate from 0.407 to 0.109 where the number of images is 30. The trend from the Table 4.1 shows that as the number of images n increase, the difference in the error rate becomes insignificant. The initial classification test results are summarized in the illustrated in Figure 4.4 where it was tested with random available images from the Image Library of American College of Rheumatology (*Image Library*, n.d.) [1].

[1] Accessed with permission from American College of Rheumatology.

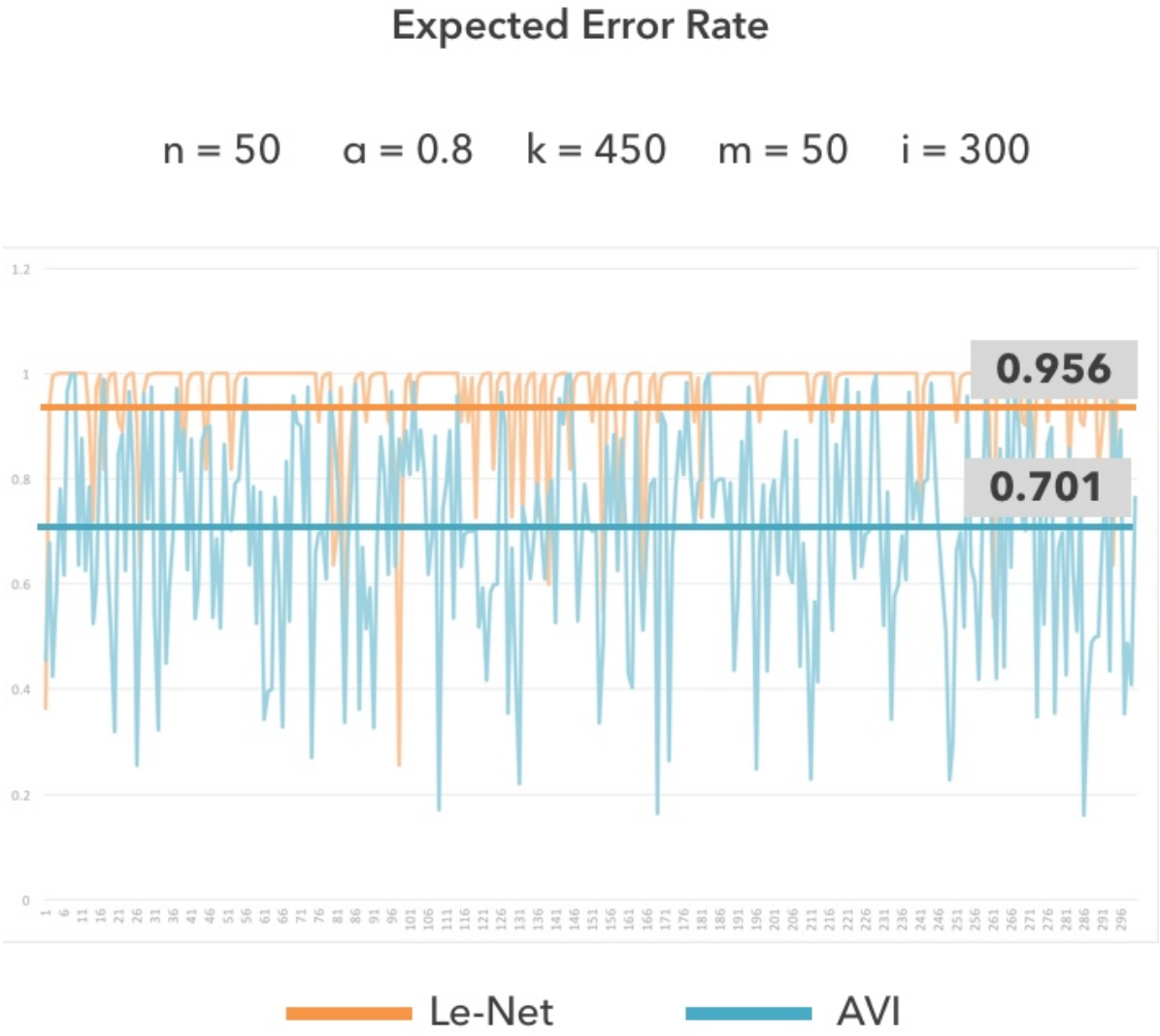


Figure 4.6. Expected error rate comparing Le-Net and AVI for n=50 (MNIST Dataset)

Figure 4.2 shows that there was a significant decrease in the cost function of the validation data and the training loss denoted by val_loss and train_loss respectively. Similarly, there was a prominent increase in the accuracy of the training dataset and

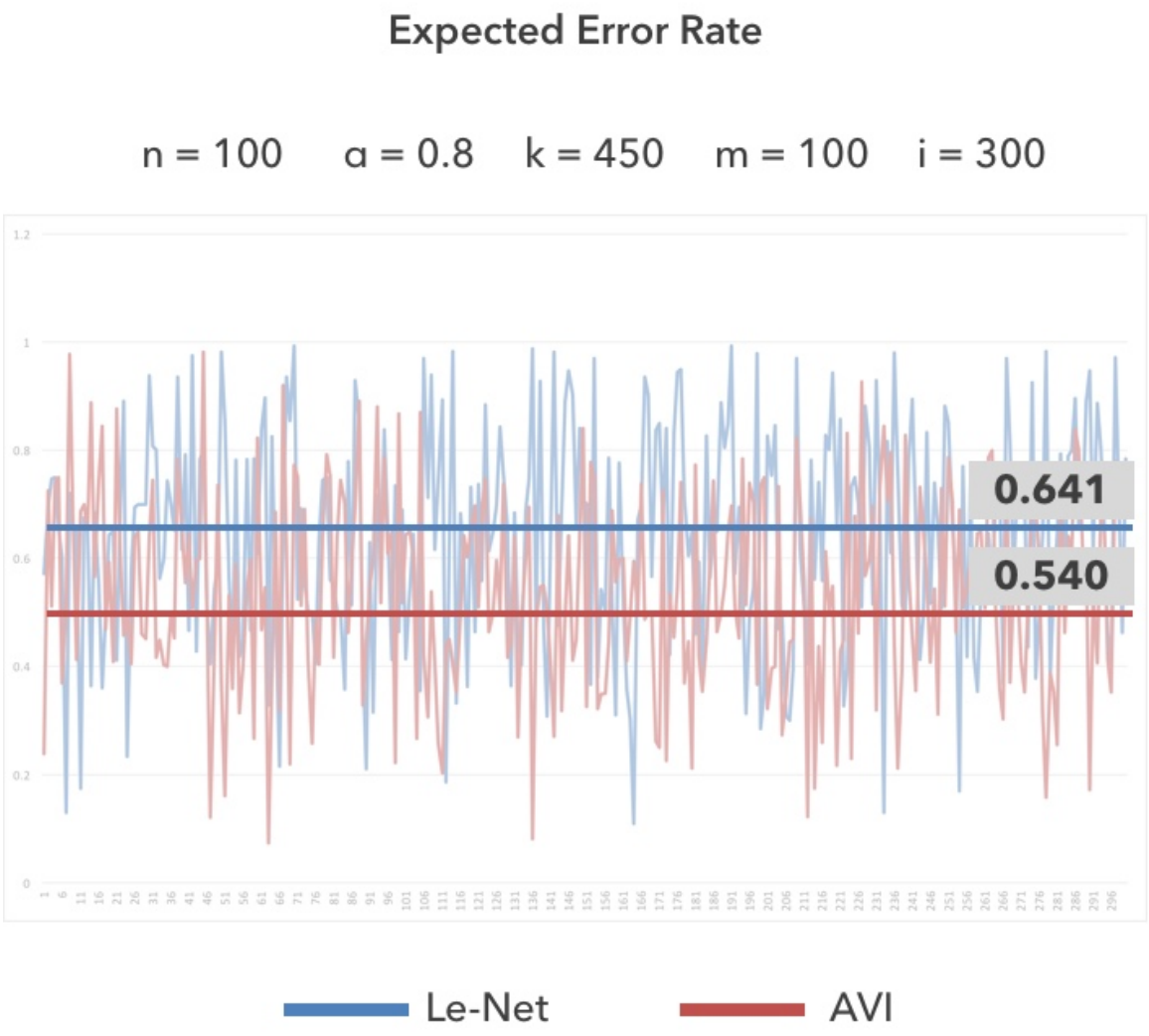


Figure 4.7. Expected error rate comparing Le-Net and AVI for n=100 (MNIST Dataset)

the validation dataset denoted by train_acc and val_acc correspondingly in Figure 4.2.

To answer the research question more concretely, a two-sample MannWhitney U test with normal tables is used to find the statistical significance of the relation between the dependent and independent variable since it is more robust

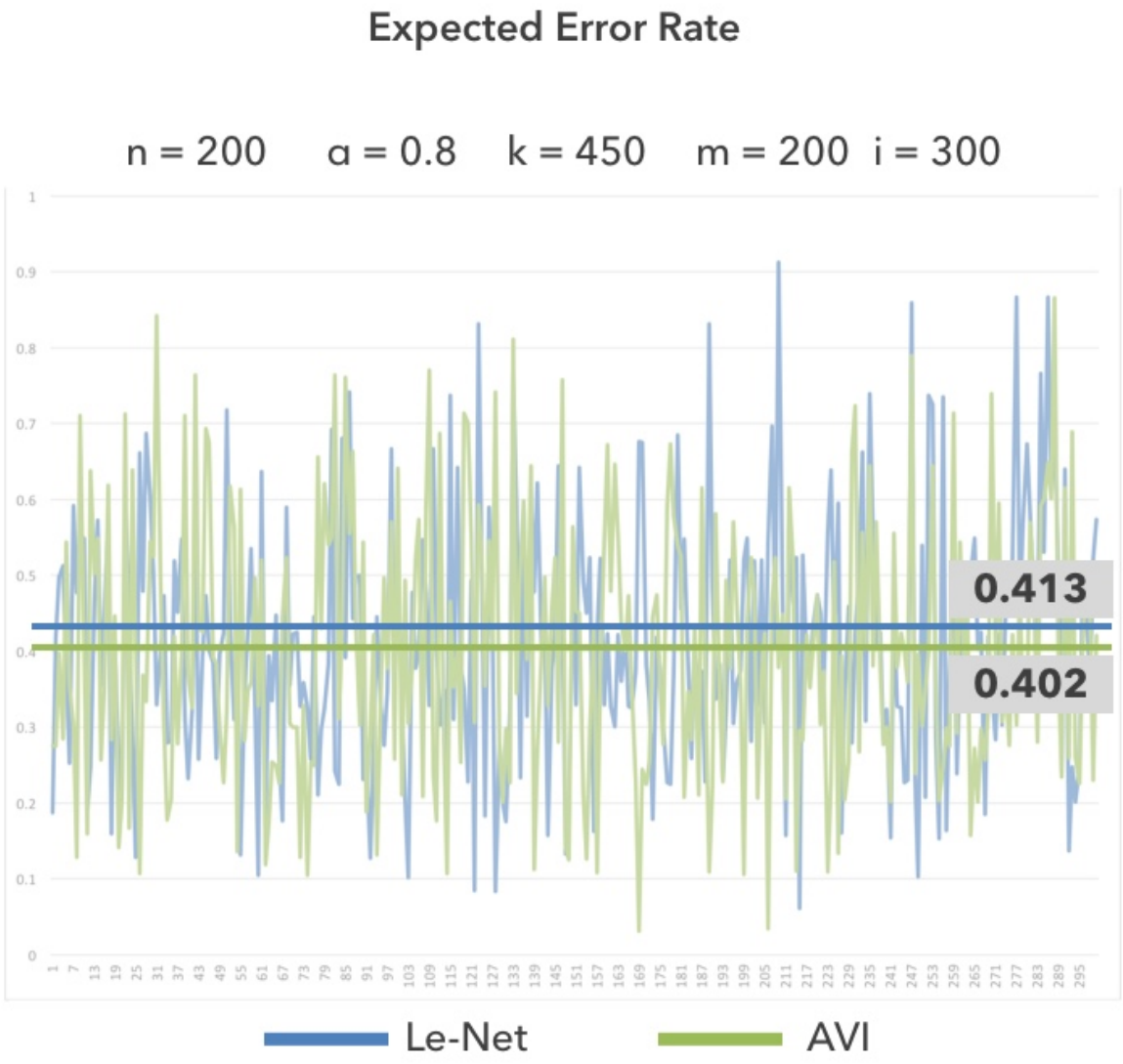


Figure 4.8. Expected error rate comparing Le-Net and AVI for n=200 (MNIST Dataset)

in case of presence of outliers within the dataset. To recall, the dependent variable for the research question is the Error Rate r_i and the independent variable is the *'m'* GAN generated synthetic images used for training. The control condition did not have *'m'* images in training whereas the experimental condition had the *'m'* images in training. The number of observations for both the control group and the

Table 4.2.

Experiments and corresponding p-values

Name of the Experiment	*p-values*
Lupus Experiment (n=30, $n_c = 100$, $n_e = 100$)	$0.00003 < 10^{-3}$
MNIST Experiment (n=50) , $n_c = 300$, $n_c = 300$)	$0.000001 < 10^{-3}$
MNIST Experiment (n=100, $n_c = 300$, $n_c = 300$)	$0.0002 < 10^{-3}$
MNIST Experiment (n=200, $n_c = 300$, $n_c = 300$)	$0.391 > 10^{-3}$

experimental group is denoted by n_c and n_e respectively. The corresponding p-values of all the experiments are summarized in Table 4.2.

4.2 Findings from Research Question 2

The research question that was put forth was whether Context Conditional GAN (CCGAN) (E. Denton et al., 2016) be used to generate simulated images that show or contain manifestation of Lupus from normal images without Lupus. In order to answer this research question, various state of the art GAN architectures was experimented. The first and most influential architecture was the one proposed by (Mirza & Osindero, 2014). It was found that conditional GAN can be used to generate artificial images of Lupus.

The subsequent research question that was put forth was whether the CCGAN generated images preferred to generate artificial images that appear like Lupus over other state-of-art baselines like DCGAN (Radford et al., 2016) and Self-Attention GAN (Zhang et al., 2019). To answer this research question an online study was conducted using Amazon mTurk where 45 participants were asked to compare and classify the images generated from these state-of-art architectures based on how they closely resemble like Lupus. Since CC-GAN inpaints a masked patch based on the surrounding pixel information, the participants were forced to concentrate on the in-painted location by slightly masking the other pixel information on both the images of comparison. The example of the fair comparison is shown in Figure 4.9. Such an arrangement is inevitable in order to enforce fair a judgment from the participant.

Table 4.5 showcases the results of randomly fashioned comparisons of images generated by models trained from the Lupus dataset. It can be seen that images generated from CCGAN architecture are preferred over the other state of the art architectures.

While comparing CCGAN with SAGAN it was seen that CCGAN was preferred 54% of the comparisons. Similarly while comparing with DCGAN, CCGAN was preferred 53.8 % of the comparisons while the chance is at 50%. Qualitative comparison of the architectures is shown in Figure 4.10. The preference

Figure 4.9. Fair Comparison between CC-GAN and other architectures

Table 4.3.

Preference Comparison between CCGAN, DCGAN and SAGAN

CCGAN vs SAGAN	*CCGAN vs DCGAN*	*DCGAN vs SAGAN*
54% (CCGAN)	53.8% (CCGAN)	56.7% (DCGAN)

comparison between CCGAN, DCGAN and SAGAN is summarized in Table 4.3. Using the Bradley Terry Model, the probability of choosing one architecture over the other is summarised in Table 4.4.

4.2.1 Results from Perceptual Study for Research Question 2

As stated in earlier sections, to overcome the shortcomings of the first perceptual study, an additional perceptual study was done in a similar setup where

Table 4.4.

Probabilities of choosing one architecture over other based on the annotations by 'Healthcare Professionals' in Amazon Mturk

CCGAN over SAGAN	*CCGAN over DCGAN*	*SAGAN over DCGAN*
0.540	0.538	0.433

26 Amazon Mturk participants were asked to rate the absolute realism of the synthetic images generated from CCGAN, SAGAN, and DCGAN. It can be seen that 17.31 % of all the judgements were "Strongly Agree" in favor of CCGAN which is relatively higher when compared to 11.9% of SAGAN. This is in-turn relatively better than 8.46% of DCGAN. CCGAN was again a clear winner in case of "Agree" as a response. The results are summarised in Table 4.4 and are visualized using bar graphs in Figure 4.11. The actual data from the perceptual study can be seen in Appendix B in Table B.1, B.2 and B.3. The last column of each table represents the sum of pair-wise comparisons for each participant number. From Table B.1, CCGAN was chosen 54% of all the comparisons while comparing with SAGAN. From Table B.2, CCGAN was chosen 53.8% of all the comparisons while comparing with DCGAN. From Table B.3, SAGAN was chosen 43.3% of all the comparisons while comparing with DCGAN.

Table 4.5.

Responses count comparing CCGAN, SAGAN and DCGAN

Name	*Strongly Agree*	*Agree*	*Disagree*	*Strongly Disagree*	*Total Judgements*
CCGAN	45	132	68	15	260
SAGAN	31	118	91	20	260
DCGAN	22	118	94	26	260

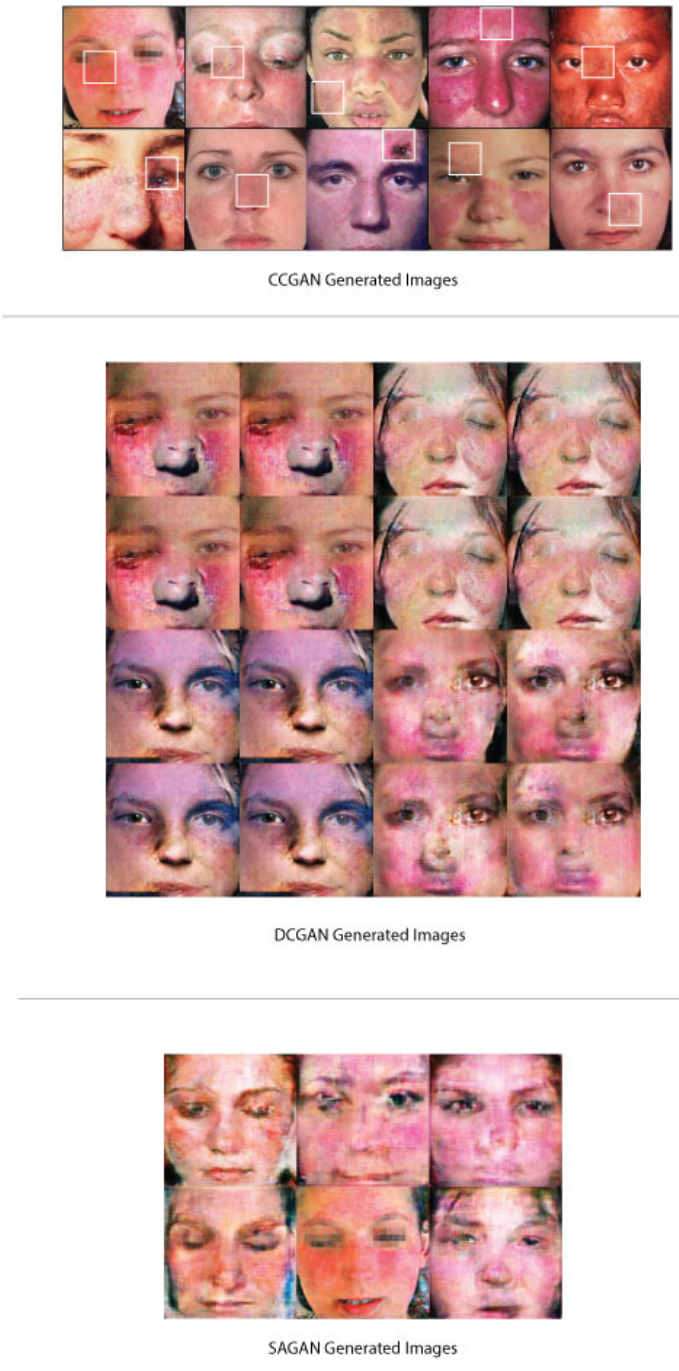


Figure 4.10. Qualitative Comparison between CC-GAN and other architectures

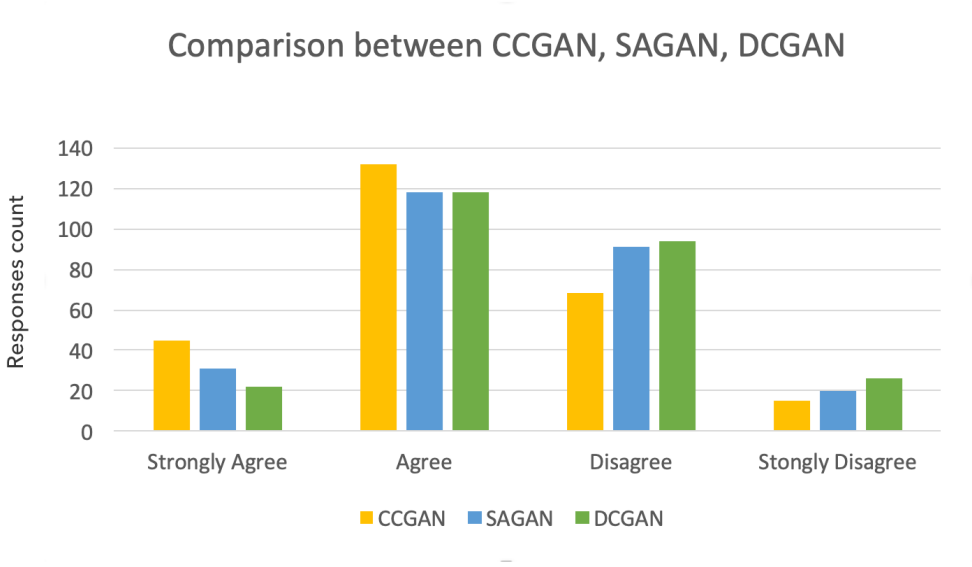


Figure 4.11. Bar Graph comparing CCGAN, DCGAN and SAGAN

CHAPTER 5. DISCUSSION

This chapter gives holistic discussions and conclusions. This chapter also projects useful recommendations to the research community that would enhance lupus research and diagnostics.

5.1 Discussion

The discussion section has been divided into two sections for research questions 1 and 2, respectively.

5.1.1 Discussion about Research Question 1

The hypothesis that was put forward was that there is a statistically significant difference in the classification error rate of the Le-Net Classifier by including the GAN generated synthetic images in the training dataset. Various parameters and the corresponding results are discussed in the Results Chapter.

Form Table 4.2, it can be seen that for all the experiments except for the last experiment where n= 200 (is sufficiently large), the results are statistically significant with $p < 10^{-3}$ confidence interval. Therefore the null hypothesis H_0 can be rejected for all the experiments where n = 30 (Lupus), n = 50 and 100 (MNIST). An important trend to be noticed is that as the number of samples used for training increases, the difference in the corresponding error rates of the experiments tends to be nullified and hence the null hypothesis holds true for last experiment n=200. Hence, the trials for n = 250 or more does not have any impact on the findings of the experiment. Therefore, GAN significantly helps to improve the classification of smaller datasets but not larger datasets which have abundant data. However, the

primary motive of this research is to improve the diagnostics of diseases which does not have large data to work with. Therefore using GAN in such context proves to be highly beneficial. Moreover, the classifier trained with Lupus images showed that Lupus can be differentiated from diseases with similar manifestations like Melasma, Rosacea and Sunburn as shown in Figure 4.4. This research has used Lupus as a model. However, the research can be extended to other diseases with similar symptoms and manifestations.

5.1.2 Discussion about Research Question 2

To recall, the research question that was put forth was whether Context Conditional GAN (CCGAN) (E. Denton et al., 2016)could be used to generate simulated images that show or contain manifestation of Lupus from normal images without Lupus. Subsequently, whether the CCGAN generated images are preferred to generate artificial images that appear like Lupus over other state-of-art baselines like DCGAN (Radford et al., 2016) and Self-Attention GAN (Zhang et al., 2019). Results showed that conditional GAN is capable of producing images that resemble images with cutaneous manifestations of Lupus and examples are shown in the Results chapter. However, as stated earlier, human observation and annotation are really important in determining the best architecture that could be used for diagnostics for heterogeneous diseases like Lupus. The online perceptual study revealed that CCGAN is indeed preferred over the other state of the art architectures. However, analyzing using the Bradley Terry model (Bradley & Terry, 1952), the statistical significance still needs closer introspection as CCGAN wins over the other baselines by a narrow margin. Therefore, the null hypothesis cannot be rejected. To recall, the null hypothesis that was articulated was that there is no statistically significant difference in the preference of the artificial images generated by CCGAN approach, over other state-of-art baselines by healthcare professionals for Lupus Diagnostics. Qualitatively analyzing the preference of CCGAN reveals

that CCGAN takes advantage of the surrounding pixels information while providing an estimation of the masked patch. Some of the useful attributes include the preservation of the identity of the faces in the images used for training, retaining the same color tone of the images that help in simulating 'butterfly rash' of a similar color tone, and easier projection of the simulated rashes on the images fed for training.

Qualitatively observing, synthetic images that had red-colored rashes on the faces were preferred over other the compared images agnostic to the architecture that was used for training. Therefore, the color tone can be explored as one of the conditions for the generation of synthetic images in future experiments. From the additional perceptual study, it is evident that CCGAN performs better when absolute realism is measured using the Likert scale. It clearly provides a potential direction for future research direction to know why is CCGAN is preferred over other architectures.

Mapping back to the primary research objectives, the following statements can be made.

1. The accuracy of Le-Net classifier with the use of images generated from GAN in the training dataset has significantly increased. To be more specific, for dataset with limited training images ($n < 200$) with $p < 10^{-3}$ confidence interval.

2. Conditional GAN (Mirza Osindero, 2014) can be used to generate artificial images with cutaneous manifestations of Lupus from normal face dataset.

3. Human evaluation of the artificial images with cutaneous manifestations of Lupus generated from the Context-Specific Conditional GAN (CCGAN) is preferred over SAGAN, and DCGAN.

CHAPTER 6. RECOMMENDATIONS AND CONCLUSION

6.1 Recommendations

The future scope of this research will include using the synthetic images generated from GAN for other neural network classifiers like ResNet, AlexNet, Inception, VGGNet. The significant result evident by performing the experiment on the MNIST dataset has shown that GAN has the potential to increase the accuracy of other neural network classifiers. Potential domain applications of GAN include Medical Imaging where synthetic CTs, synthetic Magnetic Resonance Images, synthetic X rays could be generated even from a small dataset for image segmentation and classification purposes (Emami et al., 2018). Further work could be done by exploring the use of GAN in real-time for object detection where the model learns new labels and artificially generates more similar training images. For example, using real-time object detection models like Yolo(Redmon, Divvala, Girshick, & Farhadi, 2016) as a model to simulate real-time medical diagnostics in hospitals and other public places. Therefore GAN as a machine learning approach has tremendous use in a varied context specific to Medical Imaging and Diagnostics.

6.2 Conclusion

This research was the first step towards unlocking the potential strength of applying GAN to Lupus related image data by using it to generate artificial images with cutaneous manifestations of Lupus which can be used for various research purposes.

Therefore the medical research community need not be limited to what is publicly available and use it to generate synthetic data for classification, testing and training apart from the traditional data augmentation methods.

APPENDIX A. CODE USED

The following open-source python code is accessed from HyperGAN community (HyperGAN-Community, 2016) on Jan 10, 2019.

1. Generator Model

```
from hypergan.gan_component import GANComponent

class BaseGenerator(GANComponent):

    def __init__(self, gan, config, name="BaseGenerator",
    input=None, reuse=False):
        self.input = input
        self.name = name

        GANComponent.__init__(self, gan, config,
        name=name, reuse=reuse)

    """
        Superclass for all Generators.
        Provides some common functionality.
    """
    def create(self):
        """
        Create graph
        """
        self.sample = self.build(self.input)
```

```
        return self.sample

    def add_progressive_enhancement(self, net):
        ops = self.ops
        gan = self.gan
        config = self.config
        if config.progressive_enhancement:
            split = ops.slice(net, [0, 0, 0, 0], [-1, -1, -1,
            gan.channels()])
            if config.final_activation:
                split = config.final_activation(split)
            print("[generator] adding
            progressive enhancement", split)
            gan.skip_connections.set(
            'progressive_enhancement', split)


    def layer_filter(self, net, layer=None, total_layers=None):
        """
            If a layer filter is defined, apply it.
            Layer filters allow for adding information
            to every layer of the network.
        """
        ops = self.ops
        gan = self.gan
        config = self.config
        if config.layer_filter:
            print("[base generator] applying layer filter",
            config['layer_filter'])
```

```
            fltr = config.layer_filter(gan, self.config, net)
            if fltr is not None:
                net = ops.concat(axis=3, values=[net, fltr])
        return net

    def project_from_prior(self, primes, net, initial_depth,
    type='linear', name='prior_projection'):
        ops = self.ops
        net = ops.reshape(net, [ops.shape(net)[0], -1])
        new_shape = [ops.shape(net)[0], primes[0], primes[1],
        initial_depth]
        net = ops.linear(net,
        initial_depth*primes[0]*primes[1])
        print("projection ", net)
        net = ops.reshape(net, new_shape)
        return net
```

2. Discriminator Model

```
from hypergan.gan_component import GANComponent
import tensorflow as tf

class BaseDiscriminator(GANComponent):
    def __init__(self, gan, config, name=None, input=None,
    reuse=None, features=None):
        self.input = input
        self.name = name
```

```
        self.features = features
        GANComponent.__init__(self, gan, config,
        name=name, reuse=reuse)

    def create(self, net=None):
        config = self.config
        gan = self.gan
        ops = self.ops

        net = net or self.input

        net = self.build(net)
        self.sample = net
        return net

    def reuse(self, net=None, **opts):
        config = self.config
        gan = self.gan
        ops = self.ops

        self.ops.reuse()
        net = self.build(net, **opts)
        self.ops.stop_reuse()

        return net

    def add_noise(self, net):
        config = self.config
        if not config.noise:
```

```
            return net
        print("[discriminator] adding noise", config.noise)
        net += tf.random_normal(net.get_shape(), mean=0,
        stddev=config.noise, dtype=tf.float32)
        return net

    def resize(self, config, x, g):
        if(config.resize):
            # shave off layers >= resize
            def should_ignore_layer(layer, resize):
                return int(layer.get_shape()[1])
                       > config['resize'][0] or \
                       int(layer.get_shape()[2])
                       > config['resize'][1]

            xs = [px for px in xs if not should_ignore_layer(px,
            config['resize'])]
            gs = [pg for pg in gs if not should_ignore_layer(pg,
            config['resize'])]

            x = tf.image.resize_images(x,config['resize'], 1)
            g = tf.image.resize_images(g,config['resize'], 1)

        else:
            return x, g

    def layer_filter(self, net, layer=None, total_layers=None):
        config = self.config
        gan = self.gan
```

```
ops = self.ops
concats = [net]

closest = gan.skip_connections.get_closest
('progressive_enhancement', ops.shape(net))
if closest is not None:
    enhancers = gan.skip_connections.get_array
    ('progressive_enhancement', ops.shape(closest))

    # progressive enhancement
    new_shape = [ops.shape(net)[1], ops.shape(net)[2]]
    x = self.add_noise(self.gan.inputs.x)
    x = tf.image.resize_images(x,new_shape, 1)
    #TODO what if the input is user defined? i.e. 2d test
    size = [ops.shape(net)[1], ops.shape(net)[2]]
    enhancers = [tf.image.resize_images(enhance,
    size, 1) for enhance in enhancers]
    enhance = tf.concat(axis=0, values=[x]+enhancers)

    # progressive growing
    if config.progressive_growing:
        pe_layers = self.gan.skip_connections.get_array(
        "progressive_enhancement")
        step_index = len(pe_layers)//len(enhancers)-layer-1
        if step_index >= 0:
            print("Adding progressive growing mask ",
            step_index)
            mask =
            self.progressive_growing_mask(step_index)
```

```
                enhance *= mask

        concats.append(enhance)

    if 'layer_filter' in config and
    config.layer_filter is not None:
        print("[discriminator] applying layer filter",
        config['layer_filter'])
        filters = []
        stacks = self.ops.shape(net)[0] // gan.batch_size()
        for stack in range(stacks):
            piece = tf.slice(net,
            [stack * gan.batch_size(), 0,0,0],
            [gan.batch_size(), -1, -1, -1])
            filters.append(config.layer_filter(gan,
            self.config, piece))
        layer = tf.concat(axis=0, values=filters)
        concats.append(layer)

    if len(concats) > 1:
        net = tf.concat(axis=3, values=concats)

    return net
```

3. Le-Net- Open source code accessed from (*PyImageSearch*, n.d.) Last Retrieved on Jan 10, 2019.

```
# import the necessary packages
```

```
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dense
from keras import backend as K

class LeNet:
 @staticmethod
 def build(width, height, depth, classes):
 # initialize the model
 model = Sequential()
 inputShape = (height, width, depth)

 # if we are using "channels first", update the input shape
 if K.image_data_format() == "channels_first":
 inputShape = (depth, height, width)

 # first set of CONV => RELU => POOL layers
 model.add(Conv2D(20, (5, 5), padding="same",
 input_shape=inputShape))
 model.add(Activation("relu"))
 model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

 # second set of CONV => RELU => POOL layers
 model.add(Conv2D(50, (5, 5), padding="same"))
 model.add(Activation("relu"))
 model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
```

```
 # first (and only) set of FC => RELU layers
 model.add(Flatten())
 model.add(Dense(500))
 model.add(Activation("relu"))

 # softmax classifier
 model.add(Dense(classes))
 model.add(Activation("softmax"))

 # return the constructed network architecture
 return model
```

4. Le-Net training. Open source code accessed from (*PyImageSearch*, n.d.) Last Retrieved on Jan 10, 2019.

```
# USAGE
# python train_network.py --dataset
#images --model santa_not_santa.model

# set the matplotlib backend so figures
#can be saved in the background
import matplotlib
matplotlib.use("Agg")

# import the necessary packages
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import img_to_array
```

```
from keras.utils import to_categorical
from pyimagesearch.lenet import LeNet
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import random
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
 help="path to input dataset")
ap.add_argument("-m", "--model", required=True,
 help="path to output model")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
 help="path to output loss/accuracy plot")
args = vars(ap.parse_args())

# initialize the number of
# epochs to train for, initial learning rate,
# and batch size
EPOCHS = 25
INIT_LR = 1e-3
BS = 32

# initialize the data and labels
print("[INFO] loading images...")
```

```
data = []
labels = []

# grab the image paths and randomly shuffle them
imagePaths = sorted(list(paths.list_images(args["dataset"])))
random.seed(42)
random.shuffle(imagePaths)

# loop over the input images
for imagePath in imagePaths:
 # load the image, pre-process it, and store it in the data list
 image = cv2.imread(imagePath)
 image = cv2.resize(image, (28, 28))
 image = img_to_array(image)
 data.append(image)

 # extract the class label from the image path and update the
 # labels list
 label = imagePath.split(os.path.sep)[-2]
 label = 1 if label == "santa" else 0
 labels.append(label)

# scale the raw pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)

# partition the data into training and testing
# splits using 75% of
# the data for training and the remaining 25% for testing
```

```
(trainX, testX, trainY, testY) = train_test_split(data,
 labels, test_size=0.25, random_state=42)

# convert the labels from integers to vectors
trainY = to_categorical(trainY, num_classes=2)
testY = to_categorical(testY, num_classes=2)

# construct the image generator for data augmentation
aug = ImageDataGenerator(rotation_range=30,
width_shift_range=0.1,
 height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
 horizontal_flip=True, fill_mode="nearest")

# initialize the model
print("[INFO] compiling model...")
model = LeNet.build(width=28, height=28, depth=3, classes=2)
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
 metrics=["accuracy"])

# train the network
print("[INFO] training network...")
H = model.fit_generator(aug.flow(trainX,
trainY, batch_size=BS),
 validation_data=(testX, testY),
 steps_per_epoch=len(trainX) // BS,
 epochs=EPOCHS, verbose=1)

# save the model to disk
```

```
print("[INFO] serializing network...")
model.save(args["model"])

# plot the training loss and accuracy
plt.style.use("ggplot")
plt.figure()
N = EPOCHS
plt.plot(np.arange(0, N), H.history["loss"],
label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"],
label="val_loss")
plt.plot(np.arange(0, N), H.history["acc"],
label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"],
label="val_acc")
plt.title("Training Loss and Accuracy on Santa/Not Santa")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

4.Le-Net Testing. Open source code accessed from (*PyImageSearch*, n.d.) Last Retrieved on Jan 10, 2019.

```
# USAGE
# python test_network.py
# --model santa_not_santa.model --image images/examples/santa_01.png

# import the necessary packages
from keras.preprocessing.image import img_to_array
```

```
from keras.models import load_model
import numpy as np
import argparse
import imutils
import cv2

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--model", required=True,
 help="path to trained model model")
ap.add_argument("-i", "--image", required=True,
 help="path to input image")
args = vars(ap.parse_args())

# load the image
image = cv2.imread(args["image"])
orig = image.copy()

# pre-process the image for classification
image = cv2.resize(image, (28, 28))
image = image.astype("float") / 255.0
image = img_to_array(image)
image = np.expand_dims(image, axis=0)

# load the trained convolutional neural network
print("[INFO] loading network...")
model = load_model(args["model"])

# classify the input image
```

```
(notSanta, santa) = model.predict(image)[0]

# build the label
label = "Lupus" if santa > notSanta and
# santa > 0.9 else "Not Lupus"
proba = santa if santa > notSanta else notSanta
label = "{}: {:.2f}%".format(label, proba * 100)

# draw the label on the image
output = imutils.resize(orig, width=400)
cv2.putText(output, label, (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,
 0.7, (0, 255, 0), 2)

# show the output image
cv2.imshow("Output", output)
cv2.waitKey(0)
```

APPENDIX B. COLLECTED DATA FROM THE PERCEPTUAL STUDY

p1, p2..., p45 represent the participant numbers. *C1, C2..., C10* represent the number of pair-wise comparisons.

In Table B.1 '*0*' represent selections in favor of SAGAN over CCGAN and '*1*' represent selections in favor of CCGAN over SAGAN. The last column in the Table B.1 represents the sum of pair-wise comparisons for each participant number. It can be seen that '*243*' selections were made in favor of CCGAN out of 450 total comparisons.

In Table B.2 '*0*' represent selections in favor of DCGAN over CCGAN and '*1*' represent selections in favor of CCGAN over DCGAN. The last column in the Table B.2 represents the sum of pair-wise comparisons for each participant number. It can be seen that '*242*' selections were made in favor of CCGAN out of 450 total comparisons.

In Table B.3 '*0*' represent selections in favor of DCGAN over SAGAN and '*1*' represent selections in favor of SAGAN over DCGAN. The last column in the Table B.3 represents the sum of pair-wise comparisons for each participant number. It can be seen that '*195*' selections were made in favor of SAGAN out of 450 total comparisons.

Table B.1.

Pairwise comparison results between CCGAN and SAGAN

No	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	
p1	0	0	0	0	0	0	0	0	0	0	0
p2	0	0	0	1	0	1	1	1	1	1	6
p3	1	1	1	0	1	1	1	1	1	0	8
p4	0	0	0	1	0	0	0	0	1	0	2
p5	0	1	1	1	1	1	1	0	1	1	8
p6	1	1	1	1	1	1	1	0	1	0	8
p7	0	0	0	0	0	0	0	0	0	0	0
p8	1	0	0	1	1	0	1	0	1	1	6
p9	0	1	1	0	1	0	1	0	0	0	4
p10	0	0	0	0	0	0	0	0	0	0	0
p11	1	1	0	0	1	1	1	0	0	0	5
p12	0	0	0	1	1	0	1	0	0	1	4
p13	1	1	1	1	1	1	0	0	1	1	8
p14	1	0	0	1	1	1	1	1	1	1	8
p15	1	0	0	1	1	0	0	0	1	1	5
p16	0	1	1	1	0	1	1	0	1	1	7
p17	1	0	0	1	0	0	0	0	0	1	2
p18	0	0	1	0	0	0	0	0	0	0	1
p19	1	0	1	1	0	1	1	1	1	1	8
p20	1	1	1	1	1	1	1	1	1	1	10
p21	1	0	1	1	1	1	0	0	0	1	6
p22	0	0	1	1	0	1	1	1	1	1	7
p23	0	0	0	0	1	1	0	1	1	1	5
p24	1	1	1	1	1	1	1	1	1	1	10
p25	0	0	1	1	1	1	1	1	1	1	8
p26	1	0	1	1	1	0	1	0	0	0	5
p27	0	1	0	0	1	0	1	0	1	0	4
p28	0	1	0	1	1	1	0	0	0	0	4
p29	0	0	1	1	0	0	0	1	0	0	3
p30	0	0	0	0	0	0	0	0	1	1	2
p31	0	0	0	0	0	0	1	0	0	0	1
p32	1	1	1	1	1	1	1	1	1	1	10
p33	1	1	1	1	1	1	1	1	1	0	9
p34	1	1	1	1	1	1	1	1	1	1	10
p35	0	1	1	1	1	1	1	0	0	1	7
p36	0	0	0	0	0	0	1	0	1	0	2
p37	1	1	0	1	0	1	1	0	1	1	7
p38	0	0	1	1	0	0	1	1	1	1	6
p39	0	0	0	1	0	0	0	0	0	0	1
p40	0	0	0	1	1	0	1	1	1	1	6
p41	0	0	0	1	1	0	1	0	1	0	4
p42	0	1	0	1	1	1	1	0	1	1	7
p43	0	0	1	1	1	1	1	1	0	1	7
p44	0	0	0	1	1	0	1	1	1	1	6
p45	0	0	1	1	0	0	1	1	1	1	6
										Total	243
										%	54

Table B.2.

Pairwise comparison results between CCGAN and DCGAN

No	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	
p1	0	1	0	1	0	0	1	0	0	0	3
p2	0	1	1	1	0	1	1	1	0	1	7
p3	1	0	1	0	0	0	1	1	1	1	6
p4	0	0	0	1	0	0	0	0	0	0	1
p5	1	1	1	1	1	1	0	0	1	1	8
p6	1	0	0	0	1	1	1	1	1	1	7
p7	0	0	1	1	1	0	1	0	0	0	4
p8	0	0	0	0	1	0	1	0	0	1	3
p9	1	0	0	0	0	0	0	1	1	1	4
p10	0	1	0	1	1	0	1	0	0	0	4
p11	1	0	0	0	1	1	0	0	0	0	3
p12	0	0	1	1	1	0	1	0	0	1	5
p13	0	1	0	0	0	1	0	0	1	1	4
p14	0	0	0	1	1	1	1	0	0	1	5
p15	1	0	1	1	1	1	1	0	1	1	8
p16	0	1	0	1	0	1	1	1	1	1	7
p17	0	1	0	1	0	0	1	0	0	0	3
p18	0	0	1	0	1	0	1	0	0	0	3
p19	1	0	0	1	0	1	1	1	1	1	7
p20	1	1	1	0	1	1	0	1	0	1	7
p21	1	0	0	1	0	1	1	1	0	1	6
p22	0	0	1	1	1	1	1	1	1	1	8
p23	0	0	0	1	1	1	0	1	1	1	6
p24	1	1	1	0	0	1	0	1	1	1	7
p25	0	1	0	1	1	1	1	1	1	1	8
p26	1	0	1	1	1	0	1	0	0	1	6
p27	0	1	0	1	1	0	0	0	0	0	3
p28	0	0	1	1	1	1	1	1	1	1	8
p29	1	1	1	0	1	1	0	1	1	1	8
p30	0	1	0	1	1	0	0	0	0	0	3
p31	0	1	0	1	0	0	1	0	0	0	3
p32	1	0	1	0	0	0	1	1	1	1	6
p33	1	0	0	0	1	1	1	1	1	0	6
p34	1	1	1	0	1	1	0	0	1	1	7
p35	0	1	1	1	1	1	1	1	1	0	8
p36	0	1	1	1	1	0	1	0	0	0	5
p37	0	1	0	1	1	1	1	0	1	1	7
p38	0	1	0	1	1	0	1	0	1	1	6
p39	0	0	0	1	1	0	1	0	0	0	3
p40	0	0	0	1	0	1	1	1	0	1	5
p41	0	0	0	1	1	0	0	0	0	1	3
p42	0	0	0	1	0	0	1	0	1	1	4
p43	0	1	1	1	0	1	1	1	0	1	7
p44	0	1	1	1	0	0	1	0	0	0	4
p45	0	1	0	1	1	0	1	0	1	1	6
										Total	242
										%	53.8

Table B.3.

Pairwise comparison results between SAGAN and DCGAN

No	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	
p1	1	1	1	1	1	1	1	0	0	0	7
p2	1	0	0	0	0	1	0	1	0	1	4
p3	0	0	0	0	0	0	1	0	0	0	1
p4	0	1	1	0	1	0	1	0	0	0	4
p5	1	0	0	0	0	1	0	0	0	1	3
p6	1	0	0	1	0	1	0	1	1	1	6
p7	1	0	1	1	0	1	1	0	0	1	6
p8	0	0	0	0	0	1	1	0	0	1	3
p9	0	0	0	1	0	0	0	1	0	1	3
p10	1	1	1	1	1	1	1	0	0	0	7
p11	0	0	1	0	1	0	0	0	0	0	2
p12	0	0	0	0	0	1	1	1	0	1	4
p13	0	0	0	0	0	1	0	0	0	1	2
p14	1	0	1	0	1	1	1	0	0	1	6
p15	1	1	0	0	0	1	1	0	0	0	4
p16	0	0	0	0	0	1	0	0	0	1	2
p17	1	1	1	1	0	0	1	0	0	1	6
p18	1	1	1	1	1	1	1	0	0	0	7
p19	0	1	0	0	0	0	0	1	1	1	4
p20	1	0	0	0	1	0	0	0	0	1	3
p21	0	0	0	0	0	0	1	1	1	1	4
p22	0	0	1	0	1	1	0	1	1	0	5
p23	1	0	0	0	1	1	1	1	1	1	7
p24	1	1	0	0	0	0	0	1	0	1	4
p25	0	0	0	0	0	0	0	0	0	1	1
p26	0	1	1	0	0	1	1	0	0	0	4
p27	1	1	1	1	1	1	1	0	0	0	7
p28	0	0	1	0	0	1	1	1	1	1	6
p29	1	0	1	0	1	1	0	0	0	1	5
p30	1	1	1	1	1	1	1	0	0	0	7
p31	1	1	1	1	1	1	1	0	0	0	7
p32	0	0	0	0	0	0	0	1	1	1	3
p33	0	0	0	0	0	1	1	1	0	1	4
p34	0	0	1	0	0	1	0	0	1	1	4
p35	0	0	1	0	0	1	0	0	0	0	2
p36	0	1	1	1	1	0	1	0	0	0	5
p37	0	1	1	1	1	0	1	0	0	0	5
p38	0	0	0	1	0	1	0	1	0	1	4
p39	0	1	1	0	0	1	1	0	0	1	5
p40	0	1	0	0	0	1	0	0	0	1	3
p41	0	0	0	0	0	1	0	1	0	1	3
p42	0	1	0	0	0	0	1	0	0	1	3
p43	1	1	0	0	0	1	0	1	1	1	6
p44	0	0	0	0	0	1	1	0	0	1	3
p45	0	0	0	1	0	1	0	1	0	1	4
										Total	195
										%	43.3

APPENDIX C. IRB APPROVAL

PURDUE
UNIVERSITY

HUMAN RESEARCH PROTECTION PROGRAM
INSTITUTIONAL REVIEW BOARDS

To:	VETRIA BYRD KNOY
From:	STEPHEN ELLIOTT, Chair Biomedical IRB
Date:	04/03/2019
Committee Action:	**Expedited Approval - Category**(7)
IRB Approval Date	04/03/2019
IRB Protocol #	1902021817
Study Title	Generative Adversarial Networks for Lupus Diagnostics and Medical Applications
Expiration Date	04/02/2022
Subjects Approved:	100

The above-referenced protocol has been approved by the Purdue IRB. This approval permits the recruitment of subjects up to the number indicated on the application and the conduct of the research as it is approved.

The IRB approved and dated consent, assent, and information form(s) for this protocol are in the Attachments section of this protocol in CoeusLite. Subjects who sign a consent form must be given a signed copy to take home with them. Information forms should not be signed.

Record Keeping: The PI is responsible for keeping all regulated documents, including IRB correspondence such as this letter, approved study documents, and signed consent forms for at least three (3) years following protocol closure for audit purposes. Documents regulated by HIPAA, such as Authorizations, must be maintained for six (6) years. If the PI leaves Purdue during this time, a copy of the regulatory file must be left with a designated records custodian, and the identity of this custodian must be communicated to the IRB.

Change of Institutions: If the PI leaves Purdue, the study must be closed or the PI must be replaced on the study through the Amendment process. If the PI wants to transfer the study to another institution, please contact the IRB to make arrangements for the transfer.

Changes to the approved protocol: A change to any aspect of this protocol must be approved by the IRB before it is implemented, except when necessary to eliminate apparent immediate hazards to the subject. In such situations, the IRB should be notified immediately. To request a change, submit an Amendment to the IRB through CoeusLite.

Continuing Review/Study Closure: No human subject research may be conducted without IRB approval. IRB approval for this study expires on the expiration date set out above. The study must be close or re-reviewed (aka continuing review) and approved by the IRB before the expiration date passes. Both Continuing Review and Closure may be requested through CoeusLite.

Unanticipated Problems/Adverse Events: Unanticipated problems involving risks to subjects or others, serious adverse events, and serious noncompliance with the approved protocol must be reported to the IRB immediately through CoeusLite. All other adverse events and minor protocol deviations should be reported at the time of Continuing Review.

Ernest C. Young Hall, 10th Floor - 155 S. Grant St. - West Lafayette, IN 47907-2114 - (765) 494-5942 - Fax: (765) 494-9911

Figure C.1. IRB Approval Protocol 1902021817

PURDUE
UNIVERSITY

HUMAN RESEARCH PROTECTION PROGRAM
INSTITUTIONAL REVIEW BOARDS

To:	BYRD, VETRIA L
From:	ELLIOTT, STEPHEN JOHN, Chair Biomedical IRB
Date:	05 / 29 / 2019
Committee Action:	**IRB Approval of Amendment, Expedited Category (7)**
Approval Date:	05 / 29 / 2019
IRB Protocol #:	1902021817
Amendment Version	Amendment-001:
Study Title:	Generative Adversarial Networks for Lupus Diagnostics and Medical Applications
Expiration Date:	04 / 02 / 2022
Subjects Approved:	100

The above referenced protocol amendment has been approved by the Purdue IRB.

The expiration date for IRB approval has not been altered.

Approved study documents are in the Attachments section of this protocol in CoeusLite.

You are required to retain a copy of this letter for your records.

We appreciate your commitment towards ensuring the ethical conduct of human subject research and wish you well with your study.

Ernest C. Young Hall, 10th Floor - 155 S. Grant St. - West Lafayette, IN 47907-2114 - (765) 494-5942 - Fax: (765) 494-9911

Figure C.2. IRB Approval Amendment for Perceptual study

APPENDIX D. SAMPLE SURVEY RESPONSES

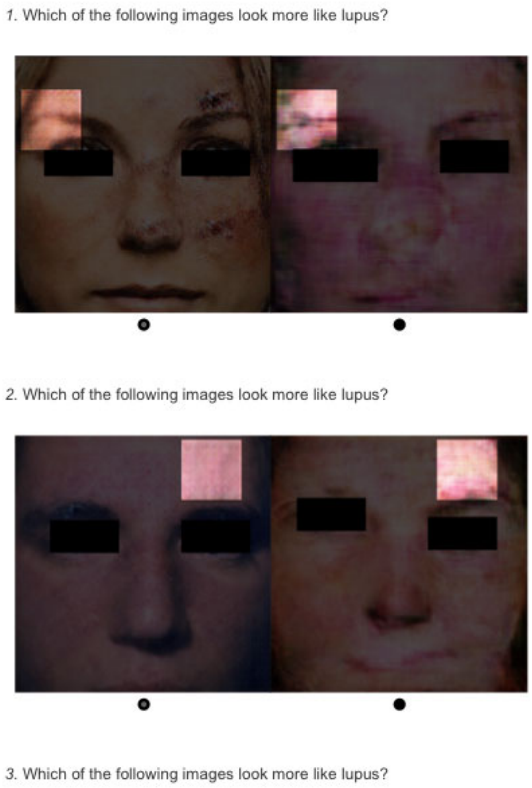


Figure D.1. Sample survey responses of pairwise comparisons of images from 1-2.

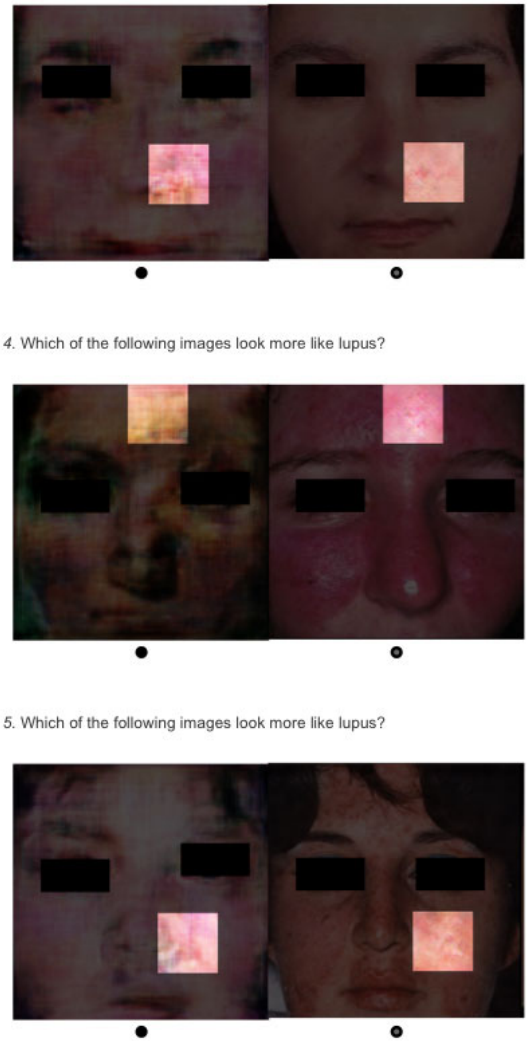


Figure D.2. Sample survey responses of pairwise comparisons of images from 3-5.

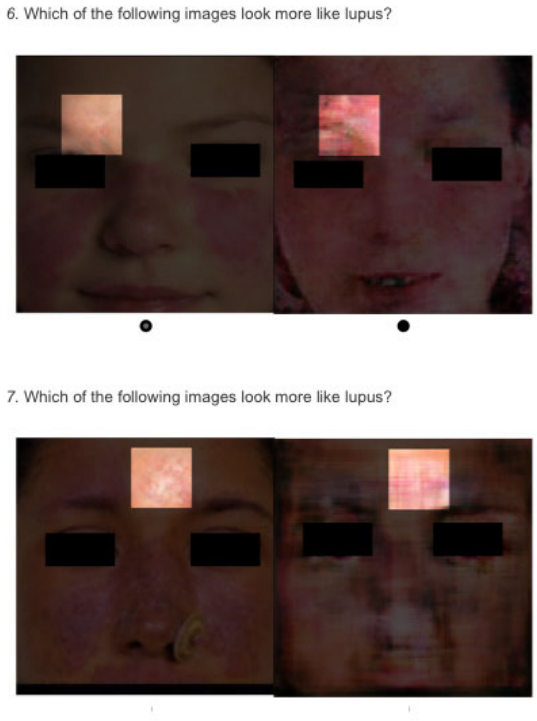
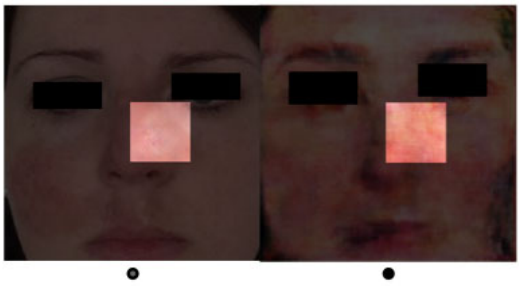
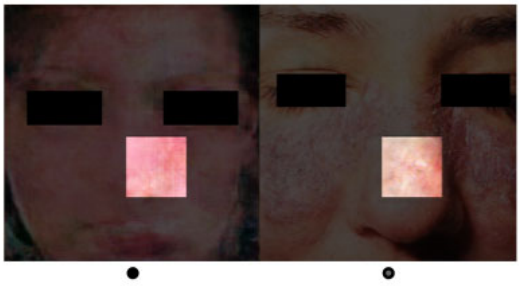


Figure D.3. Sample survey responses of pairwise comparisons of images from 6-7.



9. Which of the following images look more like lupus?



10. Which of the following images look more like lupus?

Figure D.4. Sample survey responses of pairwise comparisons of images from 8-10.

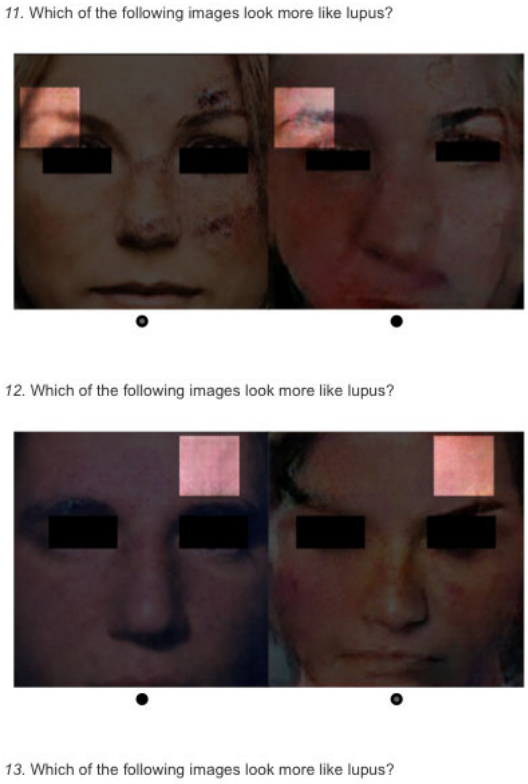


Figure D.5. Sample survey responses of pairwise comparisons of images from 11-12.

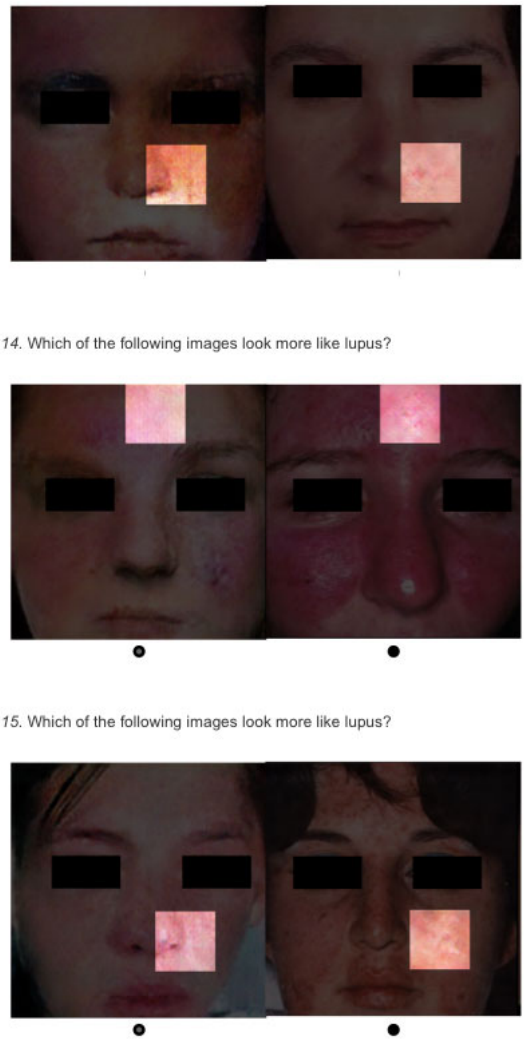
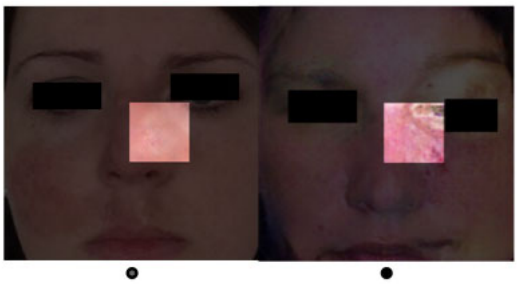
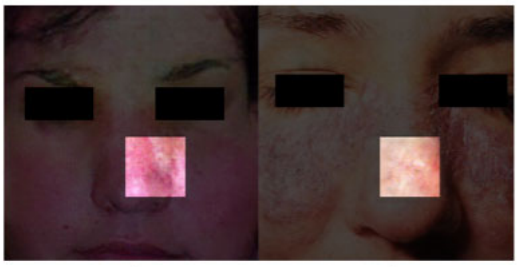


Figure D.6. Sample survey responses of pairwise comparisons of images from 13-15.

Figure D.7. Sample survey responses of pairwise comparisons of images from 16-17.



19. Which of the following images look more like lupus?



20. Which of the following images look more like lupus?

Figure D.8. Sample survey responses of pairwise comparisons of images from 18-20.

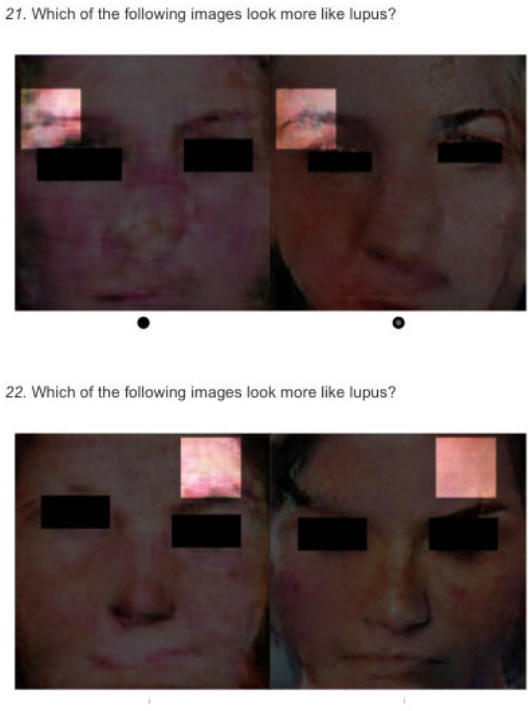


Figure D.9. Sample survey responses of pairwise comparisons of images from 21-22.

24. Which of the following images look more like lupus?

25. Which of the following images look more like lupus?

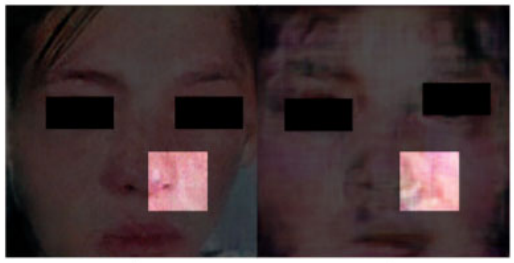


Figure D.10. Sample survey responses of pairwise comparisons of images from 23-25.

Figure D.11. Sample survey responses of pairwise comparisons of images from 26-28.

Figure D.12. Sample survey responses of pairwise comparisons of images from 28-30.

LIST OF REFERENCES

Akaike, H. (1969). Fitting Auto-regressive Models for Prediction. *Annals of the Institute of Statistical Mathematics*, *21*(1), 243–247.

Amazon MTurk. (n.d.). `https://www.mturk.com/`. (Accessed: 2019-01-30)

Antipov, G., Baccouche, M., & Dugelay, J.-L. (2017). Face Aging with Conditional Generative Adversarial Networks. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 2089–2093).

Bellman, R. (1978). *An introduction to artificial intelligence: Can computers think?* Thomson Course Technology.

Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, *39*(3/4), 324–345.

Brock, A., Donahue, J., & Simonyan, K. (2019). Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations.* Retrieved from `https://openreview.net/forum?id=B1xsqj09Fm`

Chen, & Koltun, V. (2017). Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)* (Vol. 1, p. 3).

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., & Abbeel, P. (2016). InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. (Nips). Retrieved from `http://arxiv.org/abs/1606.03657` doi: 10.1007/978-3-319-16817-3

Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., & Feng, J. (2017). Dual path networks. In *Advances in Neural Information Processing Systems* (pp. 4467–4475).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)* (pp. 248–255).

Denton, E., Gross, S., & Fergus, R. (2016). Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*.

Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Advances in neural information processing systems* (pp. 1486–1494).

Donahue, J., & Darrell, T. (2017). Adversarial Feature Learning. *International Conference on Learning Representations (ICLR)*(2017), 1–17.

Durugkar, I., Gemp, I., & Mahadevan, S. (2017). Grenerative Multi-Adversarial Networks. *In 2017 International Conference on Learning Representations (ICLR)*, 1–14.

Emami, H., Dong, M., Nejad-Davarani, S. P., & Glide-Hurst, C. K. (2018). Generating synthetic CTs from magnetic resonance images using generative adversarial networks. *Medical Physics*. Retrieved from `http://doi.wiley.com/10.1002/mp.13047` doi: 10.1002/mp.13047

Ferenkeh-Koroma, A. (2012). Systemic lupus erythematosus: nurse and patient education. *Nursing standard (Royal College of Nursing (Great Britain) : 1987)*, *26*(39), 49–57. Retrieved from `http://www.ncbi.nlm.nih.gov/pubmed/22787993` doi: 10.7748/ns2012.05.26.39.49.c9134

Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *Neurocomputing*, *321*, 321–331.

Gastaldi, X. (2017). Shake-Shake Regularization of 3-branch Residual Networks. *In 2017 International Conference on Learning Representations workshop*, *1*(3), 3–6.

Gauthier, J. (2014). Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, *2014*(5), 2.

Gibson, E., Li, W., Sudre, C., Fidon, L., Shakir, D. I., Wang, G., ... others (2018). Niftynet: a deep-learning platform for medical imaging. *Computer methods and programs in biomedicine*, *158*, 113–122.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in neural information processing systems* (pp. 2672–2680).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer vision and Pattern Recognition(CVPR)* (pp. 770–778).

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

Hiepe, F. (2014). Where is lupus hidden? *Presse Medicale*, *43*(6P2), e143-e150. Retrieved from `http://dx.doi.org/10.1016/j.lpm.2014.04.005` doi: 10.1016/j.lpm.2014.04.005

Hjelm, R. D., Jacob, A. P., Che, T., Trischler, A., Cho, K., & Bengio, Y. (2018). Boundary-Seeking Generative Adversarial Networks. *In 2018 International Conference on Learning Representations*, 1–17.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)* (pp. 4700–4708).

Huo, Y., Xu, Z., Bao, S., Bermudez, C., Plassard, A. J., Liu, J., ... Landman, B. A. (2018). Splenomegaly segmentation using global convolutional kernels and conditional generative adversarial networks. In *Medical imaging 2018: Image processing* (Vol. 10574, p. 1057409).

HyperGAN-Community. (2016). *HyperGAN Community.* Retrieved from https://github.com/255BITS/HyperGAN}. GitHub.

Image library. (n.d.). Retrieved from `https://www.rheumatology.org/`

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967-5976.

Kaggle: Digit Recognizer. (n.d.). \url{https://www.kaggle.com/c/digit-recognizer}.

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *2018 International Conference on Learning Representations.* Retrieved from `https://openreview.net/forum?id=Hk99zCeAb`

Kaul, A., Gordon, C., Crow, M. K., Touma, Z., Urowitz, M. B., van Vollenhoven, R., ... Hughes, G. (2016, 6). Systemic lupus erythematosus. *Nature Reviews Disease Primers*, *2*, 16039. Retrieved from `http://dx.doi.org/10.1038/nrdp.2016.39` `http://10.0.4.14/nrdp.2016.39`

Krizhevsky, A., Sutskever, I., & Geoffrey E., H. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, 19. doi: 10.1109/5.726791

LeCun, Y. (1998). The MNIST Database of Handwritten Digits. *http://yann. lecun. com/exdb/mnist/*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998, Nov). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. doi: 10.1109/5.726791

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis*, *42*, 60–88.

Liu, M.-Y., Breuel, T., & Kautz, J. (2017). Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems, pages=700–708(NIPS).*

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2017, Oct). Least squares generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)* (p. 2813-2821). doi: 10.1109/ICCV.2017.304

Marion, T. N., & Postlethwaite, A. E. (2014, 9). Chance, genetics, and the heterogeneity of disease and pathogenesis in systemic lupus erythematosus. *Seminars in Immunopathology*, *36*(5), 495–517. Retrieved from `https://doi.org/10.1007/s00281-014-0440-x` doi: 10.1007/s00281-014-0440-x

Marshall, E. (2002). Lupus: Mysterious Disease Holds Its Secrets Tight. *Science*, *296*(5568), 689–691. Retrieved from https://science.sciencemag.org/content/296/5568/689 doi: 10.1126/science.296.5568.689

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Nister, D., & Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *2006 ieee computer society conference on computer vision and pattern recognition (cvpr'06)* (Vol. 2, pp. 2161–2168).

Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., & Dean, J. (2018). Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., & Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems* (pp. 2352–2360).

Pyimagesearch. (n.d.). Retrieved from https://www.pyimagesearch.com/

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. *In 2016 International Conference on Learning Representations (ICLR).*

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016, 6). You Only Look Once: Unified, Real-Time Object Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted intervention* (pp. 234–241).

Roth, H. R., Lu, L., Liu, J., Yao, J., Seff, A., Cherry, K., ... Summers, R. M. (2016). Improving computer-aided detection using convolutional neural networks and random view aggregation. *IEEE transactions on medical imaging*, *35*(5), 1170–1181.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems, pages=2234–2242.*

Senaras, C., Niazi, M. K. K., Sahiner, B., Pennell, M. P., Tozbikian, G., Lozanski, G., & Gurcan, M. N. (2018). Optimized generation of high-resolution phantom images using cgan: Application to quantification of ki67 breast cancer images. *Public Library of Science (Plos) one*, *13*(5), e0196846.

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., ... Rabinovich, A. (2015, June). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 1-9). doi: 10.1109/CVPR.2015.7298594

Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., & Catanzaro, B. (2018). High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Vol. 1, p. 5).

Wang, Y., Yu, B., Wang, L., Zu, C., Lalush, D. S., Lin, W., ... Zhou, L. (2018). 3D conditional generative adversarial networks for high-quality PET image estimation at low dose. *NeuroImage*, *174*(March), 550–562. doi: 10.1016/j.neuroimage.2018.03.045

Wolterink, J. M., Leiner, T., Viergever, M. A., & Išgum, I. (2017). Generative adversarial networks for noise reduction in low-dose CT. *IEEE transactions on medical imaging*, *36*(12), 2536–2545.

Yamada, Y., Iwamura, M., Akiba, T., & Kise, K. (2018). *Shakedrop regularization.* Retrieved from `https://openreview.net/forum?id=S1NHaMW0b`

Yang, J., Kannan, A., Batra, D., & Parikh, D. (2017). LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation. *In 2017 International Conference on Learning Representations (ICLR).*

Yu, C., Gershwin, M. E., & Chang, C. (2014). Diagnostic criteria for systemic lupus erythematosus: A critical review. *Journal of Autoimmunity*, *48-49*, 10–13. Retrieved from `http://dx.doi.org/10.1016/j.jaut.2014.01.004` doi: 10.1016/j.jaut.2014.01.004

Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019, 09–15 Jun). Self-attention generative adversarial networks. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (Vol. 97, pp. 7354–7363). Long Beach, California, USA: PMLR. Retrieved from `http://proceedings.mlr.press/v97/zhang19d.html`

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., & Metaxas, D. (2017, Oct). Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)* (p. 5908-5916). doi: 10.1109/ICCV.2017.629

Zhao, J., Mathieu, M., & LeCun, Y. (2017). Energy-based Generative Adversarial Network. *In 2017 International Conference on Learning Representations (ICLR).* Retrieved from `https://openreview.net/forum?id=ryh9pmcee`

Zhu, J., Park, T., Isola, P., & Efros, A. A. (2017, Oct). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)* (p. 2242-2251). doi: 10.1109/ICCV.2017.244