

Evaluating the Efficacy of GAN-Generated Synthetic Medical Images in Enhancing Deep Learning-Based Diagnosis of Rare Diseases

by

ARITRA GHOSE

21101233

ANIKA ISLAM

21101298

NUSRAT JAHAN NUHA

21141009

MANSURA AKHTER

21201329

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
February 2025

© 2025. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

ARITRA GHOSE
21101233

ANIKA ISLAM
21101298

NUSRAT JAHAN NUHA
21141009

MANSURA AKHTER
21201329

Approval

The thesis/project titled “Evaluating the Efficacy of GAN-Generated Synthetic Medical Images in Enhancing Deep Learning-Based Diagnosis of Rare Diseases” submitted by

1. ARITRA GHOSE (21101233)
2. ANIKA ISLAM (21101298)
3. NUSRAT JAHAN NUHA (21141009)
4. MANSURA AKHTER (21201329)

Of Fall, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on February 7, 2025.

Examining Committee::

Supervisor:
(Member)

Md. Tawhid Anwar
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Co-Supervisor:
(Member)

Md. Sabbir Ahmed
Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

The leap in medical research is challenged by the scarcity of publicly available data. The insufficient and walled data results in hampering innovative research in this field. The primary objective of this thesis is to compare and evaluate some of the existing Generative Adversarial models and their efficacy on solving this data scarcity issue by generating accurate and close to the real medical images in large volume effectively. This work proposes some effective tweaks and techniques that help immensely in bringing stability to these generative models. Classification models like ResNet50 and Grad-CAM are used to validate the image generation capabilities of these models. The findings of this thesis offer insights into the potential of GAN-generated synthetic medical images to significantly improve the diagnostic capabilities of deep-learning models for rare diseases. As a result, This work not only advances medical imaging and diagnosis but also provides a valuable solution to the persistent issue of data scarcity in the realm of rare medical conditions, ultimately leading to improved patient outcomes and healthcare practices.

Keywords: Medical Imaging; Generative Adversarial Networks; Deep Learning; Generative Modeling; Explainable AI; Rare Disease; Synthetic Medical Images

Acknowledgement

Firstly, all praise to the Almighty for whom our thesis have been completed without any major interruption.

Secondly, to our supervisor, Md. Tawhid Anwar, for his kind support and advice in our work. He helped us whenever we needed help.

Thirdly, to our co-supervisor, Md. Sabbir Ahmed, who always helped us in every stage during our thesis journey.

And finally to our parents and siblings without their thorough support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

Declaration	i
Approval	ii
Abstract	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
1 Introduction	1
1.1 Research Problems	2
1.2 Research Objectives	3
1.3 Workflow	4
2 Literature Review	6
3 Dataset Description	17
3.1 Dataset Description	17
3.2 Data Preprocessing	20
4 Methodology	25
4.1 Model Description	25
4.1.1 Deep Convolutional Generative Adversarial Network	25
4.1.2 Wasserstein Generative Adversarial Network	26
4.1.3 ResNet50 and Grad-CAM	28
4.2 Model Architecture	29
4.2.1 Deep Convolutional Generative Adversarial Network	29
4.2.2 Wasserstein Generative Adversarial Network	33
4.2.3 ResNet50 and Grad-CAM	36
4.3 Model Implementation/ Training Details	38
4.3.1 Deep Convolutional Generative Adversarial Network	38
4.3.2 Wasserstein Generative Adversarial Network	40
4.3.3 ResNet50 and Grad-CAM	41

5	Result Analysis	44
5.1	Image Generation	44
5.1.1	DCGAN	44
5.1.2	Wasserstein GAN	48
5.2	Image Validation using ResNet50 and Grad-CAM	52
5.2.1	Classification on Original, Partial and Fully Augmented Dataset	52
5.2.2	Validation using Grad-CAM	56
5.2.3	Validation using the Official Test Set	58
6	Discussion	59
7	Future Work	61
8	Conclusion	63
	Bibliography	63

List of Figures

1.1	Workflow	5
3.1	Randomly selected images from dataset	19
3.2	Pie Chart of instances across skin cancer classes in the dataset	21
3.3	Column chart for Comparison between the instance distribution across classes before and after geometric augmentation	24
3.4	Doughnut Chart for Dataset Split	24
4.1	Discriminator Architecture of DCGAN	31
4.2	Generator Architecture of DCGAN	33
4.3	Critic Architecture of WGAN	35
4.4	Generator Architecture of WGAN	36
5.1	Loss against Epoch plot	45
5.2	Comparison of DCGAN-generated images at 2nd and 6th epochs . .	46
5.3	Comparison of DCGAN-generated images at 12th and 18th epochs .	46
5.4	Comparison of DCGAN-generated images at 47th and 50th epochs .	47
5.5	Comparison of 450 x 450 pixels DCGAN-generated images at different epochs	48
5.6	Comparison of WGAN-generated images at different epochs for df class	49
5.7	Loss vs Batch	49
5.8	Loss Graph Progression	49
5.9	Comparison of WGAN-generated images at different epochs for vasc class	50
5.10	Loss vs Batch	50
5.11	Loss Graph Progression	50
5.12	Comparison of WGAN-generated images at different epochs for akiec class	51
5.13	Loss vs Batch	51
5.14	Loss Graph Progression	51
5.15	Column Chart to show Number of Images per Class during Classifi- cation over all three training datasets	52
5.16	Bar Chart to show Validation Accuracy among three training datasests for all the three classes	53
5.17	Bar Chart to show Recall among three training datasests for all the three classes	54
5.18	Bar Chart to show Precision among three training datasests for all the three classes	55

5.19	Bar Chart to show Specificity among three training datasets for all the three classes	55
5.20	Bar Chart to show F1 Score among three training datasets for all the three classes	56
5.21	Grad-CAM applied to df class	56
5.22	Grad-CAM applied to vasc class	57
5.23	Grad-CAM applied to akiec class	57
5.24	Bar Chart to show Validation Accuracy among three official test datasets for all the three classes	58

Chapter 1

Introduction

In the medical world, medical imaging plays a pivotal role to diagnose and treat a disease effectively. Machine Learning (ML) along with Artificial Intelligence (AI) has provided breakthrough results in the diagnostic medical field. Using statistical analysis and machine learning (ML) techniques, has greatly aided analytical research in medical imaging, providing vital insights into the pathological characteristics of any diseases. Particularly, deep learning (DL) has emerged as an effective tool that has been applied to various aspects of medical imaging tasks. Its effectiveness extends to classification, segmentation, detection, and reconstruction. DL models excel at creating high-level abstraction or estimating posterior probabilities for certain classes by expertly combining non-linear procedures. [32].

In DL technology, particularly in supervised learning scenarios, achieving optimal generalization performance demands practitioners obtaining a large dataset which is evenly distributed across different classes, and precisely labeled training sets. However, in many research domains, including medicine, acquiring a large and well-curated dataset that fulfills all the necessary criterias is often challenging. Hence, the DL research community has conducted numerous studies to solve issues with limitations, in which solutions were acquired using training datasets with fewer collected data, incomplete or weakly defined labels. Data augmentation (DA) is popular for improvising on the generalization abilities of DL models. It involves adding new data to the training set. Due to their convenience of use and intuitiveness, DA techniques have been applied widely in a variety of domains, especially in DL-based image classification applications [32].

For dealing with data augmentation, Generative Adversarial Networks (GANs) has become a promising tool. This approach includes generating synthetic data for training by employing a generative model that can learn data from datasets which are not predefined.[26]. The GAN comprises two main components: the discriminator and the generator. Competing with one another, these two components learn artificial distributions that are similar enough with real-world distributions. The data generated by GAN are aimed to enhance the target model's generalization performance by identifying patterns that were absent from the initial training set, and it should also highlight boundaries between various groups if the GAN successfully generates a distribution that is similar to the real dataset[25].

This study focuses on evaluating the efficacy of GAN-generated synthetic medical images in enhancing deep learning-based diagnosis, emphasizing on rare diseases. Because, rare diseases are very often combined with the problems of low prevalence and limited availability of data. With these characteristics, they pose unique challenges for traditional machine learning techniques. Hence, GANs are integrated into the pipeline. The result of it aims to generate realistic synthetic images. This whole process is intended to improve deep learning model performance and help overcome the problem of limited data availability.

Our goal in conducting this research is to leverage the power of GANs to generate synthetic images that could affect powerful deep learning models in detecting rare diseases accurately. We aim to provide insights on how feasible and effective can be the use of synthetic data on the improvement of deep learning algorithms' diagnostic abilities when it comes to uncommon diseases by systematically analyzing the performance metrics of these models.

This research has significant implications for the medical field since better diagnostics can identify rare diseases earlier and more precisely, allowing for immediate action and potentially better patient outcomes. Furthermore, knowing the benefits and drawbacks of GAN-generated synthetic images in the field of medicine will influence future research along with the continuous advancement of ML applications in this field.

1.1 Research Problems

One of the most commonly encountered problems, by medical imaging, is data scarcity. In medical research, various complex neural networks are often used to build models that can detect or classify various medical conditions. The goal for these models is to accelerate the medical diagnosis and research by accurately identifying problems just like humans. But, for these models to do their task accurately and successfully, it largely depends on the data that they are trained upon [7]. The better and varied the data they are used on training, the more precise and diverse they are in achieving their goals.

However, in the real world it is not easy obtaining real data from hospitals particularly because of the institutional protocols and patient privacy that also has to be ensured along as well [3] [18]. For this reason, a large collection of medical data is often not accessible in the public domain. This friction of data unavailability hampers medical research in a broad way. Dealing with insignificantly low amounts of data is also another common problem the researchers face. Does not matter how good and sophisticated the model is, training on low volume of data often leads to models showing inaccurate and unexpected results [7].

Also, the data scarcity problem is not only limited to insufficient volume of the data. Most medical datasets consist of data containing common medical conditions, but

the amount of data instances containing rare conditions and syndromes are very limited and insufficient [5]. For instance, medical conditions that are very common are likely to have vast amounts of data in these medical datasets, but Birt-Hogg-Dube, a rare and abnormal syndromes, are likely to be very limited and negligible in comparison [1]. This imbalance in the data, due to lack of variability, could hamper the model performance badly in tasks such as classification problems [14]. While the classes with higher instances of data may have incredibly higher accuracy in the classifications, the classes with too few instances may end up not having precise classification in the end [14].

Another problem found often in medical datasets are the inclusion of noise and artifacts in the images. This could arise from various changes to the clinical settings. For example, a normal dose CT performed on patients could be dangerous and harmful for them due to the radiation effects [4] [2] and thus most CTs these days are performed on low dose. But, these low dose CT images mostly effect the diagnostic performance as they end up with significant noise and artifacts. And so, noise and artifacts remain a challenging problem in medical imaging as well.

Moreover, in medical research, particularly in imaging, segmentation tasks such as labeling or annotation of data instances are also very time consuming and challenging. The inaccuracy and inefficiency in the process could become the reason for inaccurate and wrong outcomes.

And so, it is evident that there are various obstacles visible in medical imaging today that are slowing down our innovation and advancement in medical research. With this research, we are proposing a way to eliminate these problems using Generative Adversarial Networks (GAN) and we are very optimistic that it could very well be a solution that we could be longing for.

1.2 Research Objectives

The primary objective of our thesis is to help train skin cancer image classification to detect skin cancer more accurately; by generating synthetic images for data augmentation using the Wasserstein Generative Adversarial Network (WGAN) architecture with less computational power . The augmented dataset, consisting of the original and WGAN-generated images, will then be used to train Resnet50, which is a transfer learning architecture, for accurate prediction of skin cancer type depending on accurate detection of cancerous part of the images. Furthermore, to ensure model transparency and reliability, Gradient-weighted Class Activation Mapping (Grad-CAM) will be utilized to evaluate whether the synthetic images retain identity features of the original images and to verify on the credibility of the the classification model to accurately highlight cancerous regions within the images.

The main objectives of our thesis are as follows :-

1. To resolve the data imbalance of the existing dataset by augmentation methods;

ensuring improved representativeness and diversity for robust model training.

2. To generate high-resolution synthetic images (450×450 pixels) by implementing the WGAN.
3. To classify skin cancer images into appropriate diagnostic categories using the ResNet-50 model, both prior to and after augmenting the dataset with synthetic images.
4. To assess the explainability and reliability of the classification model using Grad-CAM, ensuring the accurate cancerous regions detection and alignment with the essential features of the original images.

1.3 Workflow

Primarily, the dataset of the rare disease, which we chose to be skin cancer, is obtained from the ISIC archive. It is the HAM10000 dataset. The dataset is checked and preprocessed using image transformation, geometric augmentation, data loader configuration and class wise dataset division. These preprocessing are required for effective training of the generator and discriminator on the class specific features, which is going to be used to generate more accurate and realistic images. After the preprocessing stage, the preprocessed dataset, which is our original dataset, is split into training and validation set by a ratio of 8:2, on which classification is implemented using ResNet50 and the accuracy is taken into account. Later, Generative Adversarial Network (GAN) models, such as Deep Convolutional Generative Adversarial Network (DCGAN) and Wasserstein Generative Adversarial Network (WGAN), are trained on each class specific previously pre-processed datasets. The generated images from both the GAN models are taken into consideration to choose the best images generated, which are eventually augmented into the main dataset. Our main aim of our thesis is to extend the dataset of skin cancer which is needed for better and efficient training of a machine or deep learning model. Again, the extended dataset is split in the ratio of 8:2 for obtaining the training and validation data, which is then fed into the ResNet50 for gaining the accuracy. After the completion of acquiring both the accuracies from the original and extended dataset, the accuracy is compared. GradCam is implemented to determine the explainability of the results obtained from ResNet50 and the inheritance of the actual features from the original images into the generated images. Finally, all these results are evaluated for further outcomes.

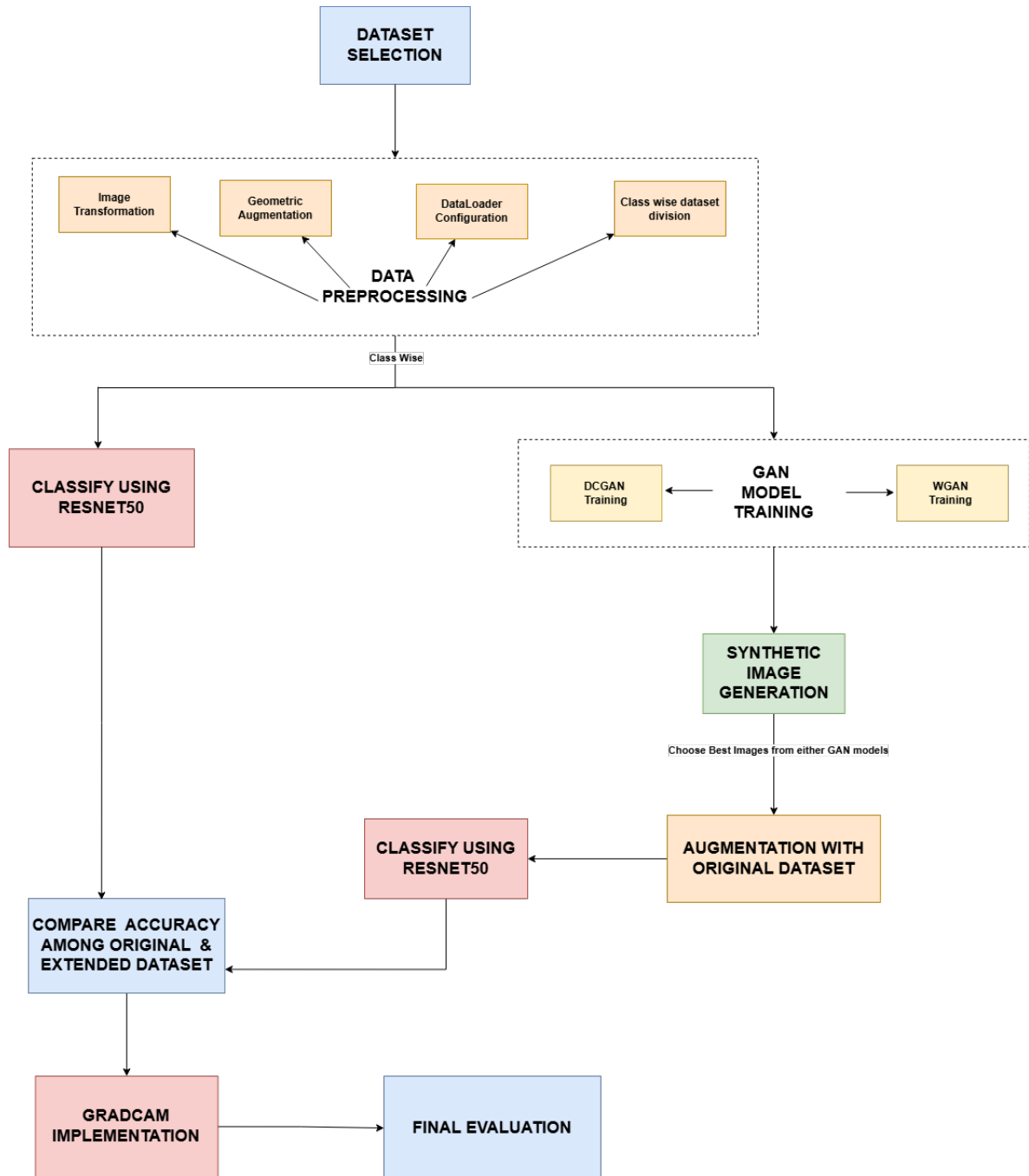


Figure 1.1: Workflow

Chapter 2

Literature Review

Yang, Q., et al. (2018) [22] introduced a generative adversarial network with wasserstein distance, WGAN, based approach in denoising the low dose CT images. In CT imaging, lowering the dose may be beneficial for the patients as it reduces the radiation consequences, but it also contrarily affects the images with increased noise and artifacts, further manipulating the diagnostic judgements. Several complex and advanced deep learning models, such as RED-CNN, Wavelet Network, previously tried to solve the problem by minimizing the mean squared errors between the ground truth and CT images, but although they were successful in denoising the images, all lacked structural detailing in the transformed images. To combat this problem, the authors of this paper proposed a GAN based method that takes into account the wasserstein distance between distributions and perceptual loss computing on the differences between images in an established feature space. In this process, WGAN transforms the images performed on low dose into ones in normal dose by estimating a good distance between the denoised image distribution of both sets, while the VGG based perceptual loss helps keeping the denoised images sharp and detailed. The author also compared the resulting images from the proposed network with other competitive models such as CNN-MSE, CNN-VGG, WGAN etc. The proposed WGAN-VGG model outperforms the competing ones indicating that the VGG based perceptual loss plays a crucial role in the generation process. All in all, the proposed network with wasserstein distance and perceptual loss could be considered a gold standard in denoising the low dose CT imaging, as it not only reduces noises in the images but also keeps them sharp, contrast and detailed.

Rezaei, M., et al. (2018) [17], in this paper proposed a novel architecture that uses conditional adversarial training approach for semantic segmentation of brain tumor. In clinical study, automated segmentation has been a continued challenge for the medical professionals due to its complex settings and hardware structure yielding variations in the appearances of the brain tumors, which further complicates the diagnosis and treatment process. The authors, in this paper, address the problem with a robust solution using Conditional GAN, simply known as cGAN. In the generator model, they initiate a Virtual-BatchNorm-Convolution to make a U-Net and for the discriminator network they use Reference-BatchNorm which is the form of Markovian GAN. The choice of these models are made so that the generator model does not lose the contextual data and the discriminator model does not face overfitting issues. The authors experimented the model on two types of datasets

provided by BraTs 2017 challenge and evaluation results were significantly good. It received a DICE score of 0.68 and Sensitivity and Specificity score being 0.99 and 0.98 respectively. To conclude, the proposed model shows a superior performance in segmentation of Brain Tumor and results are robust as well.

The authors of the study [27] illustrated a good overview of Generative Adversarial Networks and its current usage in the field of Medical Imaging. Since the field of medical imaging is ever growing and new works come out almost every other day, the authors had set a knowledge cutoff date of 1st January 2019 to process the paper only based on the works that were publicly available before that date. It starts with the original architecture of GAN proposed in 2014 and how it saw variations rapidly. Most of these variants are created due to the varying objective of either in its discriminator network, or generator network, or could be due to its architecture. Varying objective of the discriminator model introduces us to models such as f-GAN, LS-GAN, BiGAN, AC-GAN, InfoGAN, WGAN etc, while the generator model variations proposes models such as CycleGAN, CGAN, VAE-GAN etc. The changes in the architecture also ended up with new models such as DC-GAN, DCG-GAN, LAP-GAN etc. These variants, some or the other, are used in many applications of medical imaging such as Image Synthesis, Reconstruction, Segmentation, Classification, Detection, Registration, etc. Although, analyzing most of the works, it could be seen that most of the works of GAN has been done in the domain of image synthesis work, which is about 46% of the total. Image Reconstruction and Segmentation also tops the next positions of having larger use of GANs, about 37%. Magnetic Resonance (MR) seems to be the imaging modality that is most common among the others, which could be due to its requirement for an excessive amount of time in the sequence acquisition process. While there is an interesting future lying with GANs in medical imaging, there are also some challenges that are there to be eliminated as well. Most of the generations of the GANs have a common problem of pictures being blurry or very suboptimal and unsharp. These challenges are needed to be eliminated with better networks introduced that generate better images with more sharpness and feature clarity. Finally, it could be said that the generative adversarial network is still in its emerging phases and a lot could be achieved with GANs in medical imaging if we continue to work on it.

The authors of the paper [20] have proposed a way to generate abnormal and unusual synthetic MRI images with brain tumors using the generative adversarial networks. In medical image analysis, data scarcity is a formidable problem since the lack of sufficient abnormal data instances makes the entire set imbalanced and unworthy for training successful deep learning models with it. The lack of variability in the dataset results in poor classification, detection and diagnosis, further damaging the scope of medical innovation. The authors proposed a way to fill that void by introducing a new conditional generative adversarial model that can generate effective and anonymized synthetic MRI images of brain tumors. The proposed model consists of a image to image translation network called pix2pix, known as conditional GAN, to segment normal brain anatomy which then combined with BraTs provided tumor segmentations, makes a complete segmentation of brain with tumor. The model was trained on two publicly available datasets, the ADNI and BraTs dataset,

and showed impressive results in segmentation performance. Finally, it could be concluded that conditional GANs not only could be an effective model in creating new instances of quality synthetic data but having an input label mapped it could bring variations in the newly produced data as well.

Salehinejad, H., et al. (2018) [19], proposed a generative adversarial based approach to solve one of the most common problems faced while dealing with medical datasets - insufficient instances of rare conditions or syndromes [5]. Imbalanced datasets could be crucially challenging for even the most advanced classification algorithms to produce factually correct results. To solve this problem, the authors introduced a deep convolutional approach in balancing the data by augmenting new instances of synthetic data controllably using generative adversarial networks. A deep convolution neural network (DCNN) has been used in the generator and discriminator in the generation purpose for the proposed generative model (DCGAN). Although it does not specify which particular DCNN architecture could be used in the generation process, it proposes AlexNet as the DCNN architecture that could be well used on the newly created synthetic data for classification purposes. Furthermore, the model was used to generate high quality rare synthetic data to balance a private medical dataset consisting of chest X-rays and a classification test was performed upon it. The results were impressive and were approved by the certified Radiologists. The combined set of real and synthetic data had a classification accuracy of about $92.10(\pm 0.41)\%$. All in all, the proposed model shows an incredible promise in the data augmentation and classification tasks.

Rashid et al., 2019 [23] tried to work on the failings of skin lesions such as large dataset is needed for training the DL model properly and generation of large data requires expensive well-equipped instruments. For accurate classification using CNN, realistic looking synthetic dermatological images are generated using GAN and augmented to the existing dataset. The authors used the “International Skin Imaging Collaboration 2018 Challenge” dataset from which they used the results of 7 targets from the lesion image classification. The generator of GAN, made up of 4 deconvolutional layers with ReLu activation and 1 layer with tanh activation, determines the noise and produces an image and maximizes the probability of the discriminator to generate any mistakes. The produced images are augmented with the existing dataset and this dataset is fed into the discriminator, which has 4 deconvolutional layers with leaky-ReLu activation and 1 fully connected softmax layer. By binary classification, the discriminator predicts between a fake and real image. Along with this, it works as a classifier, by checking the precision, recall and F1 score, between the 7 targets. As a result, there are similarities between the synthesized and original images. To check the performance of the CNN, finely tuned architectures such as DenseNet and ResNet are fed with the same augmented dataset. Accuracy score is calculated for all three architectures, where the GAN outperforms the rest with an accuracy score of 86.1%. However, more finely tuned DL architectures are not being used in the performance comparison which could have outperformed the GAN.

In this paper [24], the authors introduced Fixed Point GAN using fixed point translation for image-to-image translation from same and cross domain; which is further

extended to detect brain lesion and localization model. To begin with, this GAN model successfully overcomes the limitations in cycleGAN and starGAN. cycleGAN uses two generators to avoid the problems faced by unpaired images. Both the generators take two different labels which can give robust performance. On the other hand, starGAN changes the unnecessary attributes along with the targeted one for cross domain translation and also fails to handle the same domain translation. Using fixed point transformation and introducing the combination of loss functions such as cycle consistency loss, adversarial loss and conditional identity loss, this GAN is developed. This new scheme is used in the generator to help it learn both same and cross domain transformation for generating the images. The discriminator remained predictive. Fixed point GAN performed better than the starGAN both qualitatively and quantitatively for cross and same domain transformations. For cross domain transformation, the new model prevents the other attributes from changing along with the target attribute and also improves accuracy. For same domain transformation, fixed point GAN handles the identity transformation without making any unnecessary changes and reduces reconstruction error generated by starGAN in image level L1 distance between generated image and input image. This model is then used to generate diseased images from normal images on BRATS 2013 dataset in brain lesion detection and localization. A difference map is formed by finding out the difference between the two type images. The maximum value around all the image pixels is used as the detection score for image level detection and color quantization followed by a component analysis done after binarizing the difference map for lesion detection. According to ROC and AUC curves, this newly introduced model with ResNet50 as the classifier outperformed StarGAN and f-AnoGAN. Sensitivity of fixed point GAN for a false positive per image is 84.5% in FROC analysis is proved to be best in comparison with the other GAN models. Since, it is a newly developed model so it may have unexplored limitations.

Yoo et al., 2021 [41] introduced an improvised CNN model for correct extraction of rare disease characteristics using rare retinal disease OCT images generated from normal OCT images with the help of cycleGAN based FSL methods. Using this model, an attempt has been made to train DL models and improve the accuracy of rare disease detection from the limited dataset available. An imbalanced training dataset and test dataset are divided randomly from the normal OCT images and major rare retinal diseases are collected from Kermany’s dataset and Google images. CycleGAN uses two generators: one for the normal OCT images and the other for the rare diseases. FSL framework extracts patterns from the rare diseases with the least dataset for augmenting the dataset using linear and elastic transformation (basic augmentation step). Using this, translation takes place from healthy OCT images to pathological OCT images. As a result, 5000 OCT images for each classes of rare disease is the highest performance obtained, and the least image generation time taken is 0.2s. Some of the few rare disease classes rejected this method of image generation due to overlapped features. The augmented dataset is fed into CNN model which is initially trained with inception v-3 model. For validation of this model, two schemes are used: five-fold cross validation using the entire dataset and inception v-3 based CNN model is trained using the final dataset on which validation has been done using the test dataset. For the five-fold cross validation, the overall performance of the proposed model is the best. For the latter scheme, the

proposed model outperforms the other FSL techniques and ophthalmologists without any prior knowledge of the rare disease based on the overall accuracy, Cohen's κ , RCI and Mathew's correlation coefficient of . But, experts with full knowledge outperforms this model. To validate the data imbalance, the test dataset is used; and using mobileNet v-2 and resNet provide similar classification performances to that of the inception v-3. Also, ROC curves are used for rare disease detection and saliency maps are generated using Grad-CAM for visualizing the features. However, this model has the following shortcomings: low resolution images are generated using the cycleGAN model which can affect the classification and fewer rare disease classes are included.

Alrashedy et al., 2022 [42] divided the Brain MRI dataset, collected from Kaggle, into two classes (images with and without Tumor) to be used by both DCGAN and Vanilla GAN for generating brain MRI images for data augmentation - to increase the instances and accuracy in classification for faster brain tumor detection. For classification, DL models such as CNN, ResNet152V2 and mobileNetV2 are trained using the augmented datasets. Also, the performance of these models are checked using a test dataset which consists of real MRI images. For generating MRI images using Vanilla GAN, the generator uses Bayes Theorem to produce fake data points for the image generation by taking noise and original images as inputs into the generative network; and the discriminator predicts the data point as fake or real by taking the original image and generated image as inputs in the discriminative networks. On the other hand, MRI images are generated using DCGAN where the generator uses convolutional transpose layers, batch norm and ReLu activation for transferring latent space using normal distribution of the dataset and the discriminator uses convolutional layers and works similar to that of the Vanilla GAN classification. Overall this procedure of image generation is a framework named brainGAN. Using both the augmented dataset, all three models are used to check the classification and performance. CNN works as a classification extractor which interprets features by using the labels of the input images for classifying the features in the test dataset. ResNet152V2 and mobileNetV2 works as feature extractors which interprets the features using pre-trained weights. The highest ratio for Tumor images, total accuracy and misclass value of CNN model classification using DCGAN generated MRI image dataset outperforms the Vanilla GAN generated images by 97.8%, 96.63%, 3.37% respectively. Using accuracy, precision, recall, loss, and AUC for interfering between the performances of all these three models - all models performed better using DCGAN generated dataset and among them ResNet152V2 outperformed with 99.09% accuracy. However, performance of other GAN types are missing here and DCGAN can work even much better if more classes are included.

Islam & Zhang, 2020 [30] generated synthetic PET images for three stages of Alzheimer's disease via DCGAN by using the ADNII baseline dataset. Using DCGAN, the proper detection of Alzheimer disease can be made with limited dataset, automated diagnosis of the diagnosis can be efficiently made and difficulty in annotating symptoms is reduced. DCGAN produces better quality synthetic images and a stable training dataset. Five convolutional transpose layers with 2d Batch-Norm for feature scale regulation and reLu activation; and a completely connected

network layer - has been used in the generator. This generator takes a vector from 100 random numbers using uniform distribution and produces the PET image. For the discriminator, CNN model is used - consisting of five convolutional layers along with Leaky ReLU activation (excluding the output layer) for preventing deep gradients, BatchNorm and strided convolutions for spatial dimension reduction instead of pooling layers involvement; a kernel with 5*5 size and a completely connected output layer; which has a Sigmoid function to generate likelihood probability scores for fake or real image prediction for the input PET image. For qualitative comparison, PET images which are synthesized and which are real are taken into accounts, and accuracy of 71.45% is obtained when the data augmented dataset is used for training a 2D CNN model. However, 3D images cannot be synthesized using this DCGAN.

Kang et al. (2020) [32] found that existing studies on using GANs for synthetic medical imaging in dementia diagnosis and prognosis lacked a quantitative evaluation and visual assessment using brain PET scans. To enhance the A estimation model, they developed a conditional GAN. They employed pre-trained DenseNet121 and model-agnostic metrics which can perform both visible and numeric assessment of generated images. For learning axial images from already augmented and pre-processed 3D FBB images, conditional WGAN-GP has been used in this method. For obtaining the generation performance, ML models such as SVM and NN have been taken into consideration. The FBB PET/CT images were retrospectively collected from Dong-A University Hospital, and the Institutional Review Board approved the study protocol. As for their following methodology, raw PET images underwent pre-processing, and 3D images were divided into A negative and positive groups for training and testing. GAN models were trained using the training set. For evaluating the similarity between generated and original images, a Visual Turing test and three quantitative metrics were used. The performance of the augmented model training dataset obtained from the training and generated sets is compared with the non-augmented model training dataset to acquire the target model's generalization performance. The experiment's findings showed that, in contrast to the Visual Turing test and t-SNE, quantitative measurements were more successful at identifying mode collapse and distribution similarity.

This study conducted by Zheng et al. (2020) [38] was aimed to evaluate if GAN could create authentic Optical Coherence Tomography (OCT) images that are suitable for educating retinal specialists and training deep learning (DL) models to classify different retinal disorders. They used the collected OCT images from the Kermany dataset. They used over 100,000 images from the dataset. Categorized them by different retinal conditions, and trained a deep learning model. They used synthetic images to test the model's generalization performance as well. For the expertised individuals hired for distinguishing between the original and synthesized images, the respective accuracy have been acquired: 59.50% and 53.67% respectively for both of the hired human experts to distinguish between real and synthetic images. For creation of high resolution OCT images, Progressively Grown Generative Adversarial Networks (PGGANs) has been used in this study. GANs use adversarial learning to train its discriminators and generators. PGGANs achieve high-resolution synthesis

by including new networks G and D during training, which is 256×256 pixels in this study. However, this study has some limitations. Firstly, the synthesized images had a lower resolution of 256×256 pixels, than the originally collected OCT images. By-default, the input size for the inception model is 299×299 pixels, and while GANs could potentially generate higher resolutions such as 1024×1024 or above, the extended required training time was not practical. Secondly, PGGANs generated four OCT images classes separately, but these were combined into two groups for training DL models. Handling multiple classification processes by training DL models, more beneficial outcomes for clinical applications can be obtained in the future. Thirdly, less common retinal disorders were not included in this research. Subsequent research will have to focus on generating images for these rare retinal disorders, potentially improving the diagnostic accuracy of DL models. Lastly, grading by retinal specialists may lead to an overestimation of diagnostic accuracy due to incorporation bias may result from the same reference standard as used in the Kermany study.

Yang et al. (2019) [26] conducted a study examining the influence of IoT on healthcare, with a focus on improving patient convenience and enabling remote monitoring for doctors. The increasing connectivity and data generation in this context highlight the need for a clinical decision support system which is robust. In order to deal with the problem, the researchers proposed a semi-supervised learning approach which incorporates Generative Adversarial Networks (GANs). The GAN is used to create an artificial labeled dataset, which improves classification accuracy and generalization. The paper also addresses the issue of class imbalance via GAN-based sampling. The strategy combines self-training and co-training strategies to increase label reliability for unlabeled data. The suggested framework for the clinical decision support system consists of Data Acquisition, Data Processing, Semi-supervised Learning, and Decision-making Support. This approach is intended to address conditions involving limited labeled data and imbalanced classes. Experimental assessments using UCI datasets show that the suggested strategy beats three other semi-supervised learning approaches used in this experiment in terms of accuracy and stability. Also, evaluations on a database of cerebral strokes demonstrate the usefulness of the approach. Comparable performance to S4VM is shown via computational efficiency study, with linear kernels typically displaying faster processing times. All things considered, this strategy offers a viable means of achieving efficient clinical decision support in IoT-based health services.

Yan et al. (2019) [25] addressed the serious threat of information leakage in collaborative deep learning networks, emphasizing the potential harm to victims and economic losses, in this paper. Focusing on decentralized deep learning, where users update parameters during multiple data passes, the paper aims to strengthen information protection under generative adversarial network (GAN) model attacks. Contributions include studying information security issues in collaborative deep learning, analyzing risks associated with GAN model attacks, investigating vulnerabilities enabling attackers to compromise the Parameter Server (PS), and proposing an algorithm to prevent GAN model attacks in CDL training, with efforts to enhance method stability and experimental verification. The well-known MNIST dataset has

been used for experimental purposes. In the experiment, a CNN-based architecture was employed, organized using the `nn.sequential()` container for a sequential layer feed-forward structure. The aim is to clarify the GAN model attack aspects. Firstly, both the attacker and the system adhere to the same protocols, allowing control over the learning rate. Secondly, attackers must align with the local model's learning rate as the discriminative network D is generated locally. Lastly, early protection measures concentrated on data encryption to secure the local model or accessed devices. The goal of these considerations is to keep collaborative deep learning secure from GAN model attacks. According to the testing results, if a malicious or infected user were to launch an attack on the system using the GAN model, the PS would be able to identify the intrusion and significantly raise the learning rate parameter. Despite being more precise and adaptable than existing information security measures, the proposed solution acknowledges the limitations of the GAN model attack. In particular, the GAN model attack reveals instability that is caused by local model training parameters and is prone to issues such as gradients that explode or vanish, highlighting areas that need to be improved.

This study led by Levine et al. (2020) [33] focused to generate realistic-looking pathology images using Generative Adversarial Networks (GANs), which will eventually lead to classifier performance improvement. Dataset has been collected for this study from the OVCARE archive and The Cancer Genome Atlas (TCGA) Genomic Data Commons Portal; which consists of five histotypes of ovarian carcinoma and a compromising whole slide images (WSIs) corresponding to five distinct cancer types. A comparative study of several generative models, such as deep texture synthesis, variational autoencoder, enhanced super-resolution GAN (ESRGAN), and progressive GANs, was carried out. Progressive GANs were found to be the only technique that could produce images of high-resolution (1024×1024 pixels) with low FID scores, indicating better images, and high information entropy. This was determined through visual inspection and quantitative measures like Frechet inception distance (FID) scores and information entropy. These outcomes suggested that progressive GANs could produce high-quality, visually authentic images similar to real ones. Consequently, their focus for systematic evaluation centered on images generated by progressive GANs, specifically those trained on individual cancer histological subtypes, which yielded more realistic results compared to class-conditional GANs.

Liu et al. (2020) [34] introduced LEAF-GAN which is considered as a novel model. It can produce four different types of images of grape leaf disease using GAN for different training models. Identifying grape leaf diseases timely and effectively is crucial for grape farming as it spreads rapidly. Many deep learning algorithms such as CNN have been used traditionally but they need high-quality and vast training data. Limitation to accurately detect grape disease occurs due to insufficient training images. Here comes the Leaf-GAN model, which not only addresses this issue by performing robust data augmentation but also handles overfitting problems, provides high-quality images and helps detect leaf diseases at an early stage. The researchers collected images of both diseased and healthy grape leaves from Plant-Village. The generator model is very similar to DCGAN containing seven de-

convolution layers along with ReLU activation and the number of channels present per layer decreases gradually. An output image is produced through the layer with tanh activation. On the other hand, the discriminator model uses the strategy of dense connectivity to ensure proper feature extraction and to address the vanishing gradient issue along with instance normalization. In order to, stabilize the model for training, they have used loss function with deep gradient penalty method and label smoothing technique which yields better results than WGANs. Also to identify proper features such as small lesions to avoid any further confusion, the model uses weight initialization which ensures proper distribution of outputs. A comparison was made between Leaf-GAN, DCGAN and WGAN via the Frechet inception distance. The Leaf-GAN model achieved better average results than the other ones. Again augmented dataset created by Leaf-GAN produced better results for various image classification models like AlexNet, VGGNet etc. To conclude, it has been proved through experiments that, the proposed model can produce efficient images with proper feature extraction, providing an effective solution to training image data shortage issues.

Yildiz et al. (2021) [40] have proposed a new deep learning algorithm based on GAN technology for subbasal nerves automatic segmentation. IVCN imaging is a diagnostic tool for observing corneal structure which allows us to diagnose and notice many diseases like - peripheral neuropathies caused by diabetes or chemotherapy, Parkinson's, dementia etc. IVCN needs a vast amount of image analysis, conventional ways (U-Net) for image segmentation aren't feasible due to their noisy nature and high error rates. Using the proposed algorithm in automatic segmentation can easily solve those problems. Both healthy and patients were included in collecting IVCN images. Later segmented 510 images were selected for the deep learning method. Images(segmented neurite maps) were generated and discriminated using CGAN and PATCH-GAN along with a modified version of UNet. Random distribution of gaussian noise, salt and pepper noise, and speckle noise were done for accessing robustness. For differentiating between the segmentation results obtained by algorithms depending upon U-Net and GAN and the manual. Corneal Nerve Fiber (CNF) has been used. Also, bias calculation was performed using Bland-Altman analysis. The performance of U-Net and GAN centered algorithms showed similarities in correlation. On the other hand, Bland Altman analysis showed lower bias and higher correlation for segmented images. However, GAN had higher accuracy when ROC curves were performed. Again GANs have two advantages over U-Nets that, they allow generator networks to create near-realistic images, which is useful in medical imaging when datasets are limited, and they are more sensitive to errors and digital image sources of noise. In this study, they have used a three-grader system, subject separation and inter-rater reliability test for increasing the reliability. Lastly, this paper solely focuses on using GAN for automatic segmentation and it has been shown that the GAN-based algorithm outperformed the U-Net method.

This paper [35] puts forth generation and evaluation of post-therapy OCT images using GAN-based technology which emphasizes on neovascular age-related macular degeneration (n-AMD) individualised treatments - upon anti-VEGF treatment. The anti-VEGF therapy response varies widely which makes it difficult for further

clinical decisions after a single shot. The issue is addressed in this study by using GAN to generate post-therapy OCT images which can help treat each patient individually. To collect data, both pre-therapy and post-therapy OCT images were collected and divided into training and testing datasets. The Pix2PixHD method which is based on GAN was implemented for image synthesis. Evaluation of the synthetic post-therapy OCT images were done on their quality, authenticity and predictive accuracy. The experimental results show that: for clinical interpretation, 92% of synthetic images are of acceptable quality and for prediction, post-treatment macular state with an accuracy rate of 85% was acquired. Previously a deep learning algorithm was applied for this same case, but it was not explainable or of high quality. However the GAN-based model was shown to be effective, but it struggled with substantial retinal structural deformation due to a lack of dataset instances and overfitting.

Spectrogram Deep Convolutional Generative Adversarial Network (S-DCGAN) proposed by Xu et al. (2020) [37] helps detect Parkinson's disease. PD is a nervous system disease with no cure, thus making it crucial to identify as early as possible. The symptoms include vocal cord damage or changes such as high amplitude, low harmonic ratio etc. For feature extraction from the voice signal, Mel Frequency Cepstrum Coefficient (MFCC) and e Human Factor Cepstral Coefficients (HFCC) are commonly used; while, KNN and SVM are used as a classifier. However, they have limitations such as inadequate voiceprint representation, complex parameter selection, and redundancy. The authors of this paper has proposed a hybrid model S-DCGAN along with ResNet50 for sample augmentation and to achieve the most accurate voiceprint recognition to address the mentioned issues. The study analyzed speech signals using the UCI dataset, which included 120 speech samples from 20 Parkinson's diseased patients and 20 healthy people. Collected voice signals are pre-processed and later converted into spectrograms. To do so, Fourier transformation, short-time power spectrum calculation and gray-scale mapping are done as it is quite complicated for GAN to process the signals directly. The proposed model combines feature-matching methods and the SN, in its architecture to create a more stable and effective model for high-resolution spectrograms. ResNet50 is used as a classifier as it increases the depth of the network, solves degradation issues and increases the fitting ability of the network. The PSNR, FID, SSIM and GAN-train methods are implemented for data augmentation. The efficacy of S-DCGAN was compared with DCGAN where S-DCGAN provided better image quality, stability and improved classification upon sample augmentation. Further experiments revealed that augmented samples enhanced the efficiency of AlexNet and VGG-16. To sum up, The S-DCGAN ResNet50 hybrid model enhances classification and recognition accuracy with minimal samples.

The researchers of the paper [29] proposed an algorithm called CGAN-IFCM which combines CGAN with improved fuzzy c-means clustering to identify voice disorders with an imbalance dataset. Their objective is to balance between the classes by incrementing the number of training samples. The paper uses VOICED and SVD databases which are available publicly where imbalanced classes can be identified. To increase the number of training samples, the paper uses CGAN to produce sam-

ples of the classes which are very few in the training set. Using CGAN also helps to restrict the generation of data for larger classes. The ICFM ensures multi-objective optimization, less variance and better clustering. Later, they analyzed CGAN-IFCM based on its performance, and necessity and compared it with cost-sensitive learning (CSL) and synthetic minority oversampling technique (SMOTE) method. In both binary and multiclass detection experiments, CGAN-IFCM outperformed SMOTE-IFCM, CSL-IFCM and CGAN-FCM. Whereas, SMOTE proved to have potential over-lapping of classes and CSL had an overfitting issue. The suggested CGAN-IFCM improved True negatives and True positives very impressively compared to existing works. Considering everything, the suggested model is proven to be effective while handling imbalanced datasets and detecting multiple classes.

This chapter gives a broad overview of the importance of GANs in the medical domain. Among all the reviewed GANs variants, two of the above works influenced our choice to implement the Wasserstein Generative Adversarial Network (WGAN) as our base model. One of the authors focused on the use of Wasserstein distance in generating high-quality normal-dose images from low-dose CT images [22]. This was further nurtured by the additional implementation of VGG which ensured sharpness and details to remain intact, along with the minimization of the noise production, during the transformation of the denoised images to its original form. With the presence of noise suppression and high quality image generation attributes, WGAN rose to the top of our thesis priority list.

Another WGAN related work has shown how WGAN generated high-quality synthetic images and overcame image generation problems by including data augmentation in the dataset used; which meets our thesis requirements [32]. Unlike the other GANs, the WGAN is more stable and reliable as it can resolve critical problems such as mode collapse and vanishing gradients. These attributes make WGAN to be more ideal for image generation purpose in our thesis.

Based on the attributes mentioned above, our thesis work progresses toward enhancing high resolution image generation by WGAN which would later be classified into skin cancer types by using deep learning models whose detection and generated images features would later be evaluated by explainable artificial intelligence (XAI).

Chapter 3

Dataset Description

3.1 Dataset Description

We used the HAM10000 (“Human Against Machine with 10000 training images”) dataset for our method evaluations. It is a publicly available dataset, consisting of dermatoscopic images of different skin lesions types. The dataset has a total collection of 10015 images which include images of 7 diagnostic categories.

The dataset is available through the ISIC archive (<https://isic-archive.com/>). It is a collection of multiple databases. As of February 12th, 2018, the archive stores a collection of 13786 dermatoscopic images [21]. The dataset was designed in the aim of boosting research on automated diagnosis of dermatoscopic images since reliable multi-class predictions were only available for clinical images of skin diseases but not for dermatoscopic images [21].

The dataset was provided to the participants of the ISIC 2018 classification challenge hosted by the annual MICCAI conference in Granada, Spain. The challenge was divided into three separate tasks that included segmentation, classification and attribute detection. But it is available for research outside the challenge as well [21].

The 10015 images that are present in the dataset, were collected over a period of 20 years. The origins of these images are mainly Austria and Australia. The documentation of these images differed greatly from one another. Images from Australia were acquired from the website of Cliff Rosendahl in Queensland, Australia. They used PowerPoint files for the images and Excel database for the metadata. Whereas, the images originating from Austria had varied storing formats as well. The images were obtained from the website of the Department of Dermatology at the Medical University of Vienna, Austria. They started storing images before the time of digital cameras. As a result, images and their metadata were stored in different other analog formats [21].

Images collected before the era of digital cameras were stored as diapositives. These diapositives were later digitized and stored as 8-bit JPEG images with a size of 800×600 pixels and 72 dpi. Some images were extracted from the digital dermatoscopy system MoleMax HD. The original images stored in the system had a

resolution of 1872×1053 pixels. These images were manually cropped and stored with the size of 800×600 pixels and 72 dpi as well. Images stored as PowerPoint files were extracted from the files and stored as source images by an automated process with the usage of python-pptx package. The year of documentation and corresponding identifier was included as part of the file names[21]. The current dataset, and the version we have used for our work consists of 24-bit RGB images, with uniform size of 600×450 pixels and 96 dpi[36].

The dataset has seven diagnostic categories. They are, Actinic Keratoses, Basal cell carcinoma, Benign keratosis, Dermatofibroma, Melanocytic nevi, Melanoma and Vascular lesions. The image distribution across these categories are shown in Figure 3.2

Actinic Keratoses(akiec)

The dataset has 327 instances under this category. Actinic Keratoses is also known as Solar Keratoses and Intraepithelial Carcinoma or Bowen’s disease. It is a common, non-invasive, variant of squamous cell carcinoma. If detected in an early stage, it can be treated locally without surgery. It is recognized as Actinic Keratoses when it appears on the face and recognized as Bowen’s disease when it appears on other parts of the body. Actinic Keratoses is induced by UV-light, thus the affected skin area bears marks of severe sun damage. Bowen’s disease is caused by human papilloma virus instead of the UV-light. The affected areas have pigmented variants [21].

Basal cell carcinoma(bcc)

There are a total of 514 images of this category present in our dataset. This category requires earlier detection because it is a common variant of epithelial skin cancer which rarely spreads to other sites of the body but grows destructively if untreated [21].

Benign keratosis(bkl)

We have a comparatively higher count of images present in our dataset that belongs to this category. A total of 1099 image instances are available in this dataset. It is a generic class and has three other subgroups. The subgroups are, seborrheic keratoses (“senile wart”), solar lentigo, and, lichen-planus like keratoses (LPLK). These subgroups may have different appearances in terms of dermatoscopic view, but they are similar biologically and reported under the same generic term histopathologically. Hence, images of these subgroups are grouped together under this one category [21].

Dermatofibroma(df)

This category has the lowest count of instances in the dataset. They have only a total of 115 images. It is a benign skin lesion which is also regarded as benign proliferation or an inflammatory reaction to minimal trauma. Affected areas bear

white patches in the centre denoting fibrosis and reticular lines underneath [21].

Melanocytic nevi(nv)

This category has the highest number of instances present in the dataset. A total of 6705 images belong to this category. This has a huge number of variants. Images of all the possible variants are present in the dataset under this category [21].

Melanoma(mel)

A total of 1113 images of this category are present in the dataset. It can appear in different variants as well including invasive and non-invasive types. Instances of all the possible variants are included in our dataset. Early detection of this category can lead to a quick recovery following a simple surgical excision [21].

Vascular lesions(vasc)

The dataset has 142 instances of this category. It can appear in different variants as well. Instances from all of the variations are included in the dataset. The affected area bears pigmented red and purple patches on skin [21].

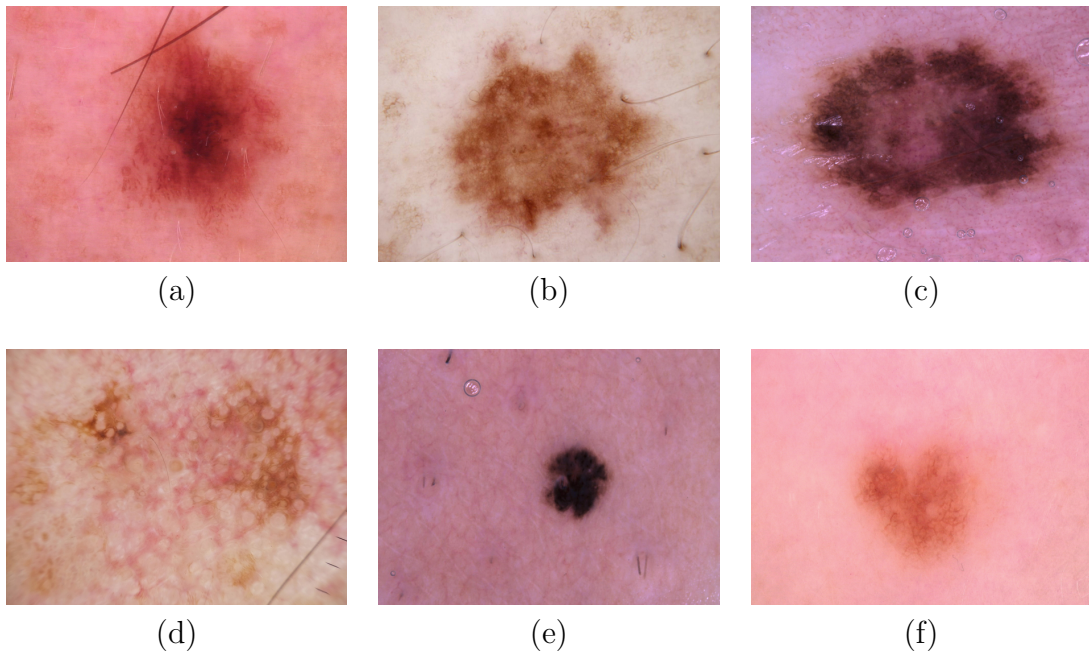


Figure 3.1: Randomly selected images from dataset

Some of the images from the dataset are selected randomly and portrayed in Figure 3.1 from where the spots on the skin are the areas which are used to determine the skin cancer type. These areas are the region of focus where a deep learning or machine learning model focuses on to make the required predictions.

Our selected dataset has already been used by other researchers for their research for image classification. Firstly, a paper [39] used fast GAN to train the dataset and focus on increasing the classes precision with lower instances. Later, the augmented dataset was used for image classification and used a mistake-driven approach along with Wide-ResNet to improve the accuracy in this stage. Final results showcase an accuracy rate of 95%. Another paper by Charan et al. (2020) [44] used this HAM10000 dataset to classify the images. They have used deep learning and CNN-based models to classify lesion images. Upon modifying existing images by changing their brightness, contrast, or other values randomly, they augmented the images with the dataset for further procedure. The augmented dataset was used in Efficient-net, Res-Net and other CNN models for classification and achieved almost 85.1 percent accuracy results. Another paper using this same dataset proposed a new model combining VGG-19 and ASPP for image segmentation to efficiently captivate semantic and contextual information. Jha et al. (2020) [31], used multiple datasets along with HAM10000 to evaluate the results. Double-U net outperformed the existing U-net in all experiments ensuring image quality and better segmentation masks for complex images. Lastly, a paper by Baygin, Tuncer, and Dogan, (2022) [43] used the HAM10000 dataset to train a machine learning-based skin cancer detection model. The original publicly shared dataset consists of images divided into two directories, denoting their train and test splits. However, for the purpose of their study, they combined images from both the directories for their training process of the proposed model. Their proposed pyramidal classification model also focused on only two main classes of the dataset, Benign and Malign. The study is entirely focused on detecting skin cancer from primarily obtained images in order to help specialists so that they can provide a better and quicker diagnosis. On contrary to all followed procedure in the mentioned papers, our study focuses on generating synthetic images from the images obtained from this dataset rather than performing a classification test. We then aim to seek expert opinions on the generated images to evaluate their quality and to acquire further assurance incorporating numeric metric values, we test a pre-trained classification model with our generated synthetic images.

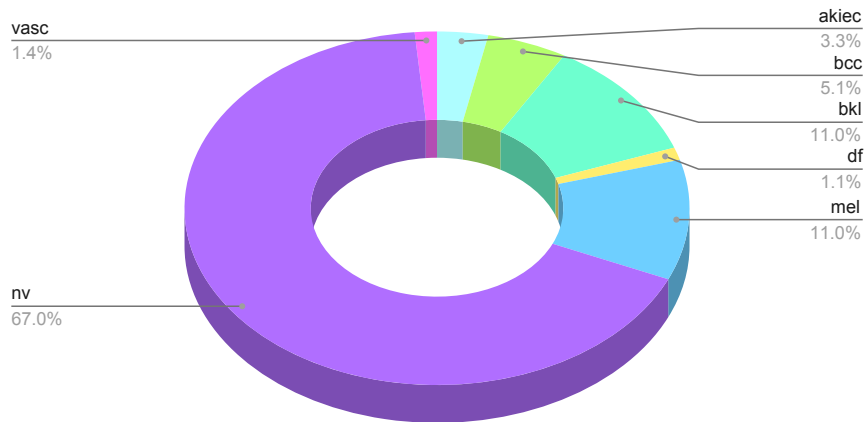


Figure 3.2: Pie Chart of instances across skin cancer classes in the dataset

3.2 Data Preprocessing

In the original dataset, total 10015 images were stored into two folders, denoting them as part 1 and part 2 in the naming convention, along with a csv file containing their metadata. As the very first step of pre-processing, we divided all the images into their category specific folders. A python script for automating this process and python's csv module to read metadata from the csv file originally provided with the main dataset - have been used. We further used the shutil module to copy images from the provided source folders and moved them to their category specific directories; and also used python's os module to handle these category specific directories.

An image is represented by a combination of colors : Red, Green and Blue. For one image, values for blue can be the dominant one and for another, values for red and green together can dominate the latter. In a machine learning model, dominance of images can vary. For instance, an image needs a variation of red, blue and green to show its actual self, but the trained model can produce an image with the color blue when that particular color channel blue overpowers the other color channels making the image inaccurate. For this reason, the RGB color channel values are adjusted using the mean and standard deviation in case of all the images available during the preprocessing.

Although this is a required data pre-processing step for most of the machine learning models dealing with images, a few things need to be adjusted differently for generative models when performing this normalization. Instead of using mean and standard deviation of the images, we use a mean value of 0.5 and standard deviation value of the same 0.5 for all color channels. This technique normalizes the value of the three color channels into a range between -1 and 1 . This normalization technique also turns out to be most convenient since we will be using an activation function like Tanh in the generator. Once normalized, we then convert these RGB pixel values into pytorch tensors.

Furthermore, images can be divided into batches using DataLoaders from the PyTorch library. The DataLoaders take the entire dataset, image size, batch size, shuffle, num_workers, pin_memory and drop_last as parameters. For our work, we have set the batch size to 16, shuffle to True and num_workers to 0. Setting the batch size to 16, each batch generated will contain 16 images. Batching is necessary for iterating images present in an individual file separately in the generative and discriminative networks. To acquire uniform image size to work with in the convolution layers of GAN models and resolve the size constraint, the images has been resized from 600×450 to 450×450 . On the other hand, the images will be shuffled for each batch due to setting shuffle to True. The purpose of shuffling is to allow the discriminator to work on unique image data instead of using any repeated data. Furthestmost, the number of workers is set to 0 which is needed to avoid the chances of multithreading problems which can arise due to data loading in CPU with limited resources. Pin memory is used to control the speed of data transfer from CPU to GPU which is needed when dealing with large datasets becomes mandatory. This pin memory is activated by setting its value to 1 and deactivated by setting

its value to 0. One of the parameters in the data loader which needs to be taken into consideration is the drop last. This ensures that the last batch to be discarded if the data size is not divisible by batch size. This excluding of batch is needed to ensure correct gradient calculation during training of the model using this dataset. To sum up, the updates of all these parameters allows the GAN model to optimize its training process over the dataset.

For image visualization at any stage, we are required to denormalize the images again using mean and standard deviation both 0.5 so that the values of the color channels are set back to their original one.

Moreover, the images have been divided into folders based on which skin cancer type they belonged to. Dividing a dataset into classes helps the GAN model to learn the class specific features of the images by focusing more on each and every detail of the images. These will later be used by the generator of the GAN model to generate more accurate and realistic synthetic images based on a specific class; which will ensure better prediction of skin cancer type when fed into a deep learning model for classification. This division also allows the generated images to overcome the overlapping of multi-class features; thus making the image generation maintain more stability. One of the many reasons include the overly generalised and dominating features present in the overall dataset to be excluded with the help of splitting the dataset into categories. This helps the generator of the model to avoid learning from these features and produce biased images revolving around those features only. Now, the generator can generate images focusing solely on the features available for a specific class not on the overall image distribution. Doing this, the generator can learn and the discriminator can predict with more precision which reduces the chances of error by either of them. For all the above reasons, our main goal of this preprocessing approach is to ensure the model to be trained on class specific images one at a time.

Not only the split of the dataset into classes allows the generator and discriminator to be trained specifically on a specific class, but also allows the detection of any data imbalance present. Data imbalance needs to be resolved so that the generator does not generate images by focusing more on the features of the classes with large image distribution compared to that of the smaller ones. This decreases the quality and realism of the generated images for lower classes and vice versa; which creates biases in the generator. To overcome this, we have approached the geometric augmentation method to increase the image distribution across the lower classes, as shown in Figure 3.2, to stabilize data imbalance. We have used transformations, such as 90° and 180° rotations, on the images to be added in the lower classes; so that identical features across all the images of the rest of the class are maintained. To keep the actual features of the original images of the classes intact, the image distribution has been increased by including a copy of some of the existing images in that specific class. Following this approach, we have tried to resolve the problem of data imbalance in the dataset.

The instance or the image distribution before and after applying the geometric

augmentation methods is shown in Figure 3.1 which clearly shows the instance distribution for lower classes i.e. Actinic Keratoses (akiec), Basal Cell Carcinoma (bcc), Dermatofibroma (df) and Vascular Lesions (vasc) has been increased for stabilizing the data imbalance present. A better visual representation to observe the difference in instance distribution across the classes caused by the geometric distribution is shown in Figure 3.3

	Before Geometric Augmentation	After Geometric Augmentation
Class	Instances	Instances
akiec	327	1635
bcc	514	1028
bkl	1099	1099
df	115	1150
mel	1113	1113
nv	6705	6705
vasc	142	1420
Total	10015	14150

Table 3.1: Instance Distribution across classes before and after geometric augmentation.

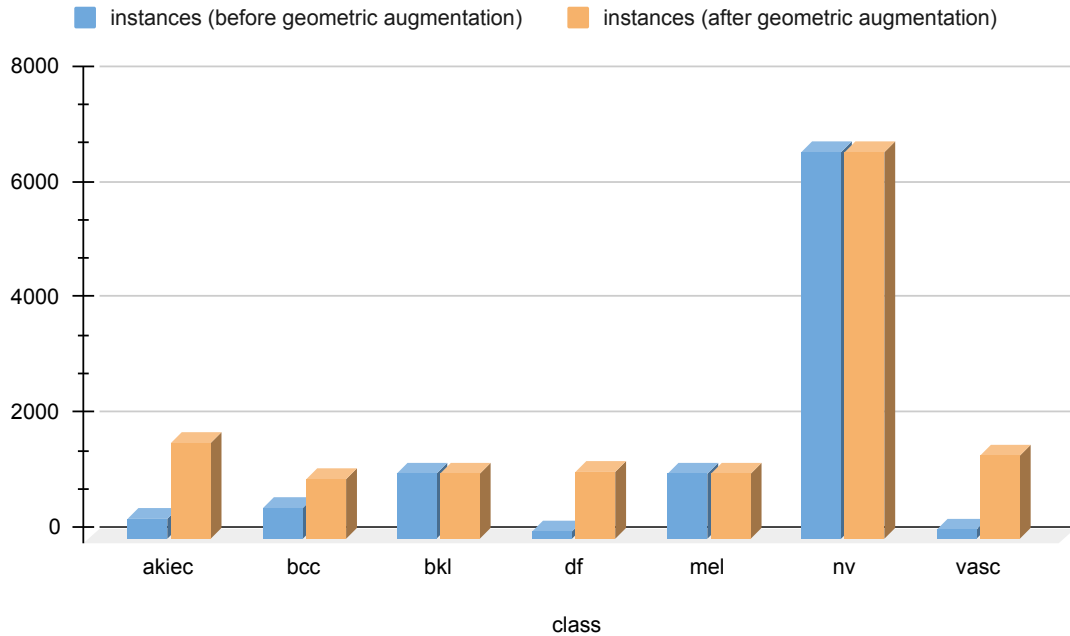


Figure 3.3: Column chart for Comparison between the instance distribution across classes before and after geometric augmentation

Later on, while performing classification tasks, we splitted our dataset along with the augmented images into a training and validation set by the proportion of 8: 2,

where each set contained 8012 and 2003 images respectively; which is displayed on Figure 3.4.

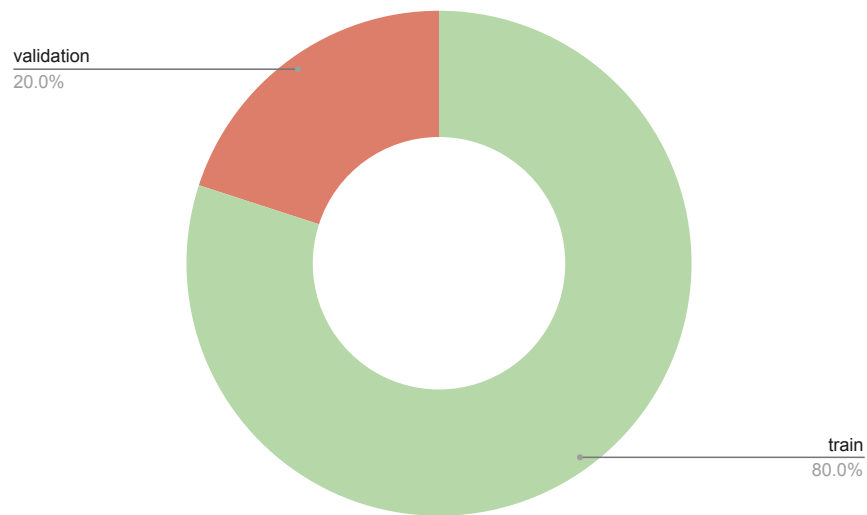


Figure 3.4: Doughnut Chart for Dataset Split

Chapter 4

Methodology

Generative Adversarial Network was a groundbreaking breakthrough in the field of generative modeling. The introduction of GANs, paved the way for a whole new world of possibilities. Using this sophisticated idea of adversarial process between two networks - a Generator and a Discriminator, researchers introduced multiple variations of the model that helps us do requirement specific tasks. They either improve the efficiency of the core model or transform it into ones that are very much effective for the specific tasks like segmentation, augmentation, classification etc. This chapter discusses the methodologies used to achieve our research objectives.

Lots of research has already been performed to find ways that speeds up the process of learning of these two networks - be it by adding gradient penalty on the loss, or using a whole lot of other optimization techniques. Since, the objective of our research was to find ways such that we can generate more synthetic images from the already existing ones, that too minimizing the required compute as much as possible, GAN was at the top of our choice from the beginning of course.

There are multiple variations of GANs available that are suitable for the task we wanted to perform, that is data augmentation. But we wanted one that was close to the original vanilla GAN so that we can start from a very scratch and base model and ultimately tweak and tune it as much as possible so the outcomes are best suited per our expectation. Hence, we chose DCGAN, initially, as the model of choice to generate synthetic images from our chosen dataset. Then later on, we used another model - Wasserstein GAN to satisfy the same purpose. Additionally, ResNet50 was used for classification purposes on both the original dataset and the augmented extended dataset with our generated images. Finally, Grad-CAM was implemented to visualize and validate the important features of the synthetic dataset.

4.1 Model Description

4.1.1 Deep Convolutional Generative Adversarial Network

The Deep convolutional GAN techniques are an upgrade over traditional GAN models which have better stable performance and generate good quality synthetic images

while consuming less computational resources. The traditional GAN proposed a model where there are two distinct models - a Generator, G and a Discriminator, D , who play a minmax game with each other to ultimately generate new images [9]. In this minimax game, the Discriminator tries to maximize accurately classifying the image between real and fake ones, and the generator tries to minimize the probability that the discriminator will correctly predict its output generated images are fake.

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

Improving this traditional convolutional GAN, DCGAN proposed a technique where the generator and the discriminator explicitly uses deep convolutional and transposed convolutional layers, respectively, as their architectures [12]. The proposed architecture also suggests LeakyReLU as the activation function, Batch norm layers and strided convolution and transpose convolution layers. Meanwhile, similar to vanilla GAN as well, the generator takes a latent vector as an input and tries to generate synthetic images while mapping it to the original image distribution. Whereas, the discriminator provides a probabilistic output to distinguish between generated fake images and original ones.

4.1.2 Wasserstein Generative Adversarial Network

WGAN was introduced to solve the vanishing gradient problem and mode collapse issue of standard GANs. Such an improvement is seen by introducing a loss function named Wasserstein distance or Earth-Mover (EM) distance. Standard GANs usually use functions like the Jensen-Shannon (JS) Divergence for their loss functions. It takes two probability distributions, P_r , which is the probability distribution for real data. And, P_g , which is the probability distribution for the generated data. The JS divergence tries to interpret how similar or different these two distributions are by taking an average of both the distributions and calculating how much each one diverges from this average[15].

However, Simple sequences of probability distributions fail to converge under JS divergence but easily converge under EM distance[15].

For example, in a hypothetical scenario, if we have a vertical line at $x=0$ in a 2 dimensional space and y -coordinate(z) is randomly given between 0 and 1; the real data happens to lie uniformly along this vertical line. The generated data happens to be a vertical line at $x=\theta$, instead of $x=0$. Here, the parameter θ can be adjusted to move the probability distribution of generated data closer to the probability distribution of real data. Or, in terms of visual transformation, the generated vertical line can be moved closer to the real one by moving this parameter. In such a scenario, the JS divergence calculates how different two distributions are. Then, if it sees that lines are at different places meaning $\theta \neq 0$, the JS divergence becomes $\log 2$. It

denotes that the algorithm couldn't find any overlap between the distributions and as a result the lines are very different from each other. But, as θ approaches to 0, and both the lines finally overlap when $\theta=0$, the JS divergence becomes 0. However, it does not provide smooth convergence. Only when the lines finally overlap as θ becomes 0, the JS divergence takes a big leap and reaches 0[15].

$$JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2, & \text{if } \theta \neq 0, \\ 0, & \text{if } \theta = 0. \end{cases} \quad (4.2)$$

In our DCGAN structure, we used Binary Cross Entropy (BCE) as our loss function. Which, in the given scenario, acts so much like the JS divergence did. In the case of $\theta \neq 0$, the BCE loss saturates as probability distribution of real data is close to 1 and probability distribution of the generated data is close to 0. As a result the generator fails to get meaningful training instability which yields training instability. And, when θ approaches 0, BCE loss also lead to vanishing gradient problem in a pretty similar way like JS divergence did. When θ becomes 0, BCE loss becomes minimum and generator outputs 0.5 for both real and fake data, denoting no distinction between the both. But the convergence isn't a smooth one and loses valuable gradients along the way[15].

Whereas, Wasserstein distance gives promising performance in terms of the JS divergence here. It measures the cost of moving the line from one point to another.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y) \sim \gamma} [|x - y|] \quad (4.3)$$

In the given scenario, the cost is the horizontal distance between the two lines. Which is proportional to $|\theta|$ [15].

$$W(P_0, P_\theta) = |\theta| \quad (4.4)$$

Therefore, it's evident that it gives useful gradient slope for optimization, and aids the generator in gradually improving its output.

It is also clear that EM distance creates a slope that is almost differentiable at every point, which also is an ideal characteristic for a loss function. Because when slopes like the ones created by JS divergence are unable to differentiate at most points and get locally saturated, problems like vanishing gradients occur.

WGAN, with Wasserstein Distance as its loss function, provides the generator with meaningful gradient information, mitigating the issue of vanishing gradients. Thus, the generator achieves better convergence and the result is seen in the better quality

of the outputs. It also contributes in reducing training instability and better optimization process.

4.1.3 ResNet50 and Grad-CAM

ResNet50

Residual network, widely referred to as ResNet50 follows a deep convolutional network architecture composed of 50 layers. The model adheres to complex and deeper networks which ensures acquiring non-linear diverse features. ResNet50 has the ability to overcome the vanishing gradient problem as it uses the “skip connections” approach. This approach allows the gradients to flow through a deeper network efficiently, bypassing some layers and effectively diminishing the vanishing gradient issue [45]. The architecture consisting of deeper layers ensures extracting hierarchical characteristics of data and the residual learning approach helps achieve better accuracy and a more robust training process, making this model superior to previously successful AlexNet, VGG, etc. A pre-trained ResNet50 is highly effective to be used for transfer learning purposes as its residual framework enables the reusability of network layers rather than building everything from scratch. Thus its model can be employed for classification, localization, segmentation and various other purposes.

Grad-CAM

Gradient weighted class activation mapping, simply known as Grad-CAM is a sophisticated approach to determine how classification models make their predictions. This technique visualizes significant regions of the input images that help the classification models make their predictions. The application of this self-explainable AI helps build trustworthiness and transparency. The convolutional neural network models or deep learning models are known to be black boxes as we can’t specify what features made them do their predictions. Transparency is a big challenge in fields like medical image classification and so on. Implementing Grad-CAMs in such cases helps build trust as now it can be visualized easily which regions of the image data contributed the most to the decision-making process of those models. It also helps in debugging the issues with the training process when the highlighted regions turn out to be irrelevant to the domain features. Thus by understanding how the model works and which regions are most relevant overall model performance can be improved with the help of Grad-CAM.

4.2 Model Architecture

4.2.1 Deep Convolutional Generative Adversarial Network

For our thesis, since we were dealing with an image size of 450×450 pixels, we built our Discriminator and Generator network in such a way to deal specifically with this image size. So, the discriminator network of our model takes an image size of 450×450 , while the generator outputs or generates an image of size 450×450 pixels

in the end.

The Discriminator Architecture

The Discriminator in our proposed model takes an image of size 450×450 and then it passes through 7 different layers to ultimately turn into a probabilistic value between 0 and 1 determining whether the image belongs to a real set of images or the fake ones.

In the first layer, since we used a batch size of 16, the discriminator receives a batch of 16 images with 450×450 pixels ranging in 3 color channels. So, the input tensor is of size $16 \times 3 \times 450 \times 450$ dimension. Then, inside the first layer, first a 2D convolution with kernel size 4, stride being 2 and padding being 1 is performed onto the tensor. We set the output channels to be 64, so that it starts with 64 feature maps from the first layer. The decision to use 64 output channels came from the very fact that most other models also use a similar number of feature maps and seemed like a standard choice for our task. Also, kernel size was chosen 4 as it was not either too small nor too high. We chose stride to be 2, so that the spatial dimension can be downsampled by a factor of 2. We set the bias to be false, since we did batch normalization immediately after this, in subsequent layers after this. We then perform a Leaky Rectified Linear Unit (LeakyReLU) as the activation function to bring non-linearity to our neural network in learning complex patterns among images [8].

$$f(x) = \begin{cases} x & \text{if } x \geq 0, \\ \alpha x & \text{if } x < 0, \end{cases} \quad (4.5)$$

Ultimately, passing through this first layer, the tensor ends up in $16 \times 64 \times 225 \times 225$ dimension.

In the next subsequent layers - layer 2 to layer 6, we use 2D convolutions of the same settings: kernel size being 4, stride 2 and padding 1. Stride being 2 through these layers, helps the downsampling of the spatial dimension throughout all these layers. Meanwhile, we increase the output channels in each layer by doubling down from the previous layer. This helps us extract more complex and abstract features as we go deeper in the discriminator. Unlike the first layer, in these layers, we perform Batch Normalization to improve training stability and make the network faster [10].

$$\text{running_mean} = (1 - \text{momentum}) \cdot \text{running_mean} + \text{momentum} \cdot \text{batch_mean}, \quad (4.6)$$

$$\text{running_var} = (1 - \text{momentum}) \cdot \text{running_var} + \text{momentum} \cdot \text{batch_variance} \quad (4.7)$$

Ultimately, the same activation function as the first layer - LeakyReLU is performed to conclude each of these layers.

By passing through all of the first 6 layers, the tensor eventually turns into a $16 \times 2048 \times 7 \times 7$ dimension of tensor where the feature map is now more dense and accurate in learning details. Starting from 64, doubling down at each layer, it becomes 2048 now as it enters the 7th and final layer. The final 2D Convolution is performed on the tensor in this layer, with kernel size 7, stride 1 and padding 0 that reduces the spatial size to $16 \times 1 \times 1 \times 1$. Now, unlike any of the previous layers, we flatten the tensor and eventually compute the probability. After the 7th convolution layer, we end up with a $1 \times 1 \times 1$ size tensor for each 16 images. We perform `nn.Flatten()` to convert all the multi-dimensional image tensors to a 1D tensor. So, we end up with a 16×1 tensor, where 16 is the batch size or number of images. Then finally, we apply the sigmoid function on each of these tensors.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (4.8)$$

By applying the sigmoid formula, we effectively end up getting a probability for each of the 16 images in a range of 0 to 1. This probability for each of the images indicates the discriminator's classification of whether the image is real or not.

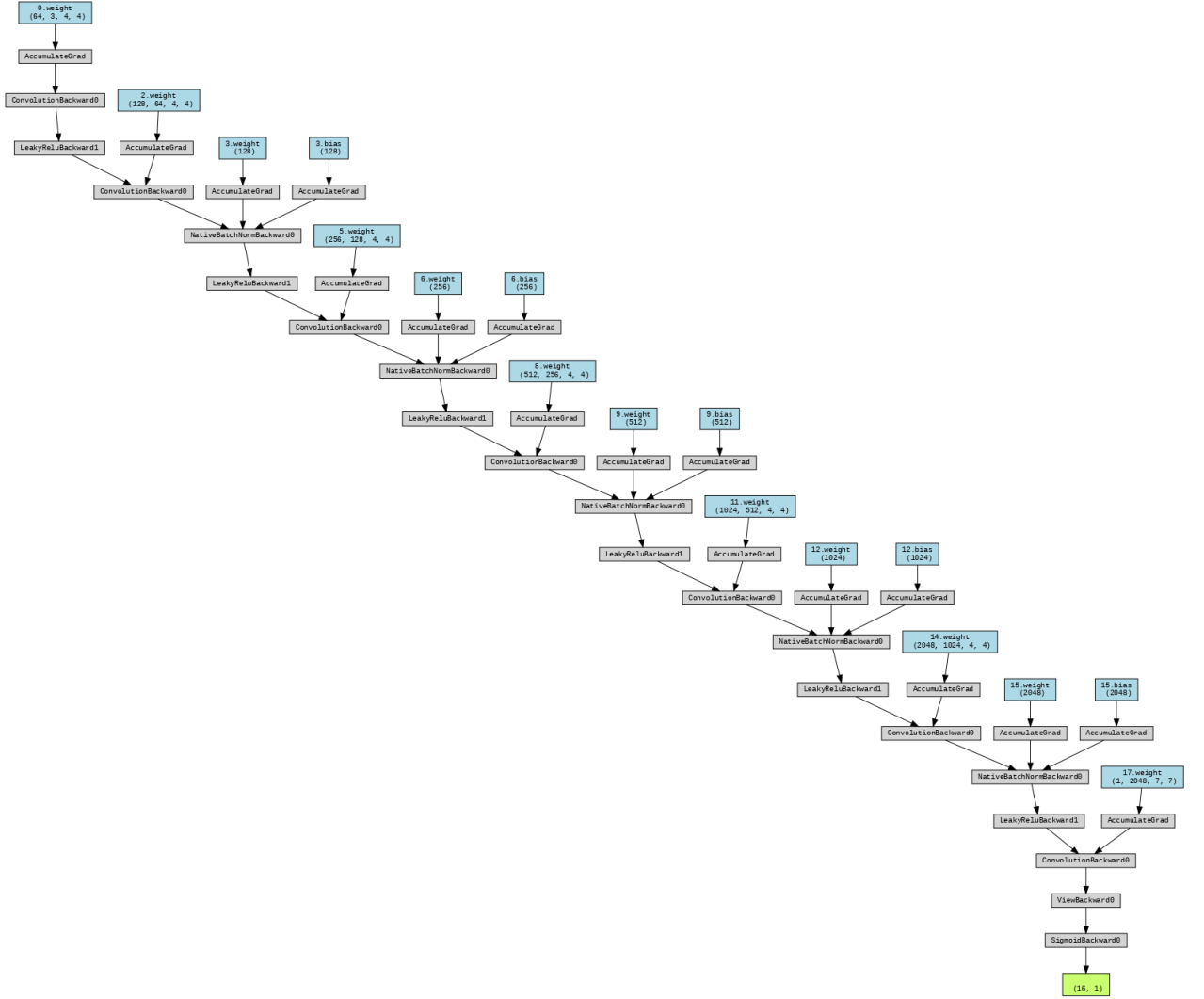


Figure 4.1: Discriminator Architecture of DCGAN

The Generator Architecture

The Generator works exactly the opposite way the discriminator works. It takes a latent tensor and gradually upscales it by passing through each of the 7 layers to eventually generate an image by end.

Since, we are using a latent vector of size 128, the generator in the first layer, receives a tensor of dimension $16 \times 128 \times 1 \times 1$. Unlike the discriminator, we use 2D transpose convolution in these layers that upscales the spatial dimensions by learning and effectively expanding the feature maps into bigger ones. The output feature for this is set to 2048 so that it can end up having a rich feature representation. The transposed convolution layer in the first layer uses a kernel size 7, stride 1 and padding 0 - similar to the settings in the last layer in discriminator. By passing through it, the input latent tensor turns into $16 \times 2048 \times 7 \times 7$ dimension. Then, we perform batch normalization on it just like we did in the discriminator, to improve training stability. Once that is done, we perform an activation function at the end of this layer, to achieve the same goal of bringing non-linearity. But, unlike the

discriminator, we use the ReLU function here[6].

$$f(x) = \max(0, x) \quad (4.9)$$

In the subsequent layers - layer 2 to layer 7, the transpose convolution layer is used with the same kernel, stride and padding values - 4, 2, 1 respectively. This ensures that with each passing layer the spatial dimension is upscaled by double. The output channels are also downscaled by a factor 2 in these layers so that as it goes deeper in the network, it focuses more on fewer and particular channels. So the output channels start from 2048 channels and go through 1024, 512, 256, 256, 128, 64 and then eventually 3 in the last layer. Similar to layer 1, the same batch normalization and activation function is performed with the same settings in all layers up to layer 6.

Then in the final layer, the transposed convolution results in the tensor to be sized in $16 \times 3 \times 450 \times 450$, where the 16 demotes the batch size or image counts. Then ultimately, we perform a hyperbolic tangent activation function on the tensor.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (4.10)$$

What this activation function effectively does is that it transforms each of the input values in the range from -1 to $+1$. So, finally after passing all these layers, starting with a latent tensor, we finally get a batch of 16 images as output that are sized 450×450 pixels each with 3 color channels.

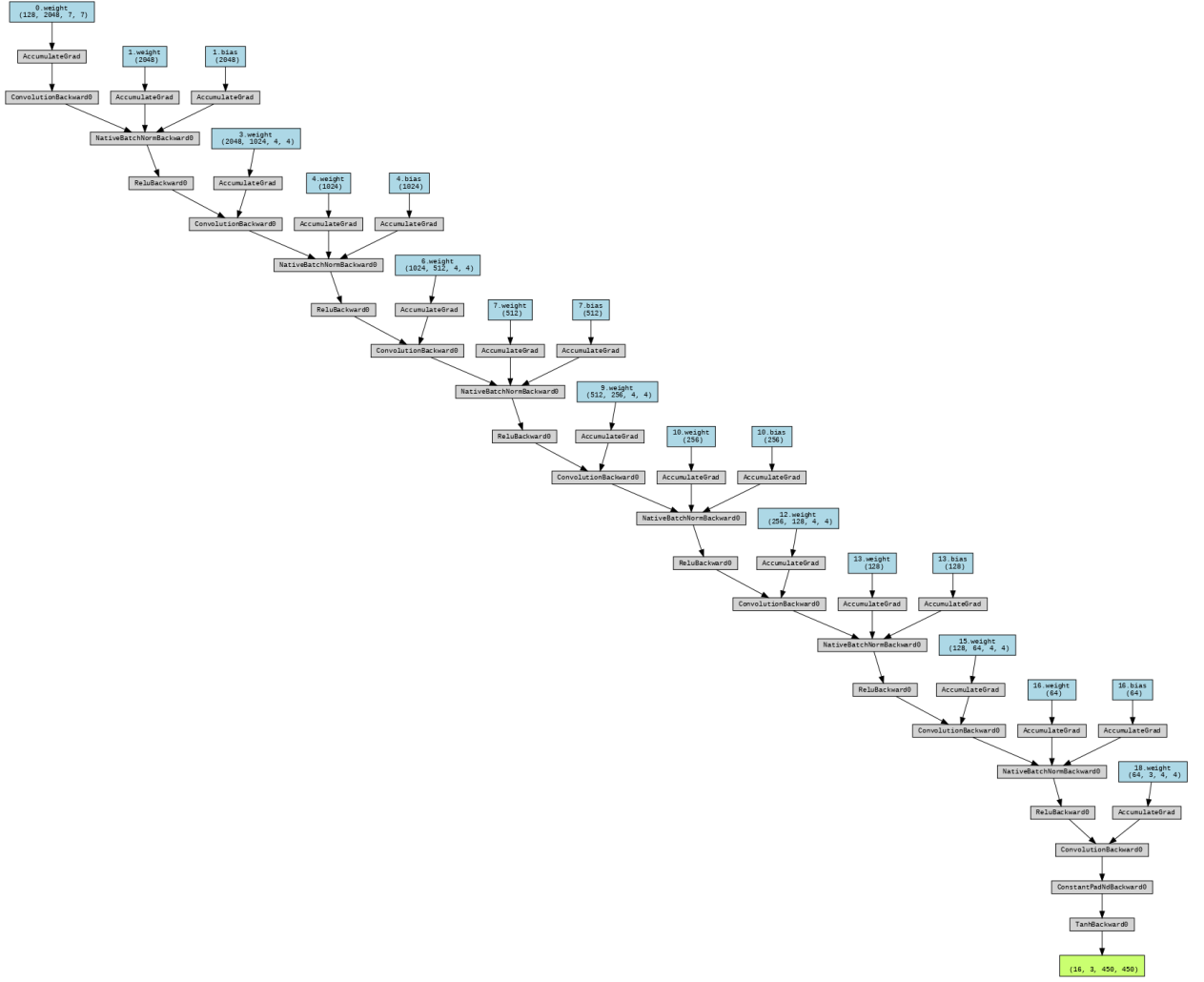


Figure 4.2: Generator Architecture of DCGAN

4.2.2 Wasserstein Generative Adversarial Network

For our thesis, we focused on producing 450×450 pixel size output images. Therefore the convolution layers of the generator and critic were designed accordingly.

The Critic

In WGAN, we have a critic instead of a discriminator. The critic takes the 450×450 images as input in the input layer. This input reduces the spatial dimension of the image by a convolution operation with stride value 2. It increases the feature depth from `channel_img` to `features_d`. The `channel_img` value is 3 for RGB images which gets increased to 32 by default. After that, LeakyReLU is applied with a small slope of 0.2 for introducing non-linearity for negative values. This process helps in avoiding inactive or dead neurons. Then the architecture has a total of 5 feature extraction blocks. Each block consists of one convolution layer, one instance normalization layer, and a LeakyReLU activation layer. Instance normalization is

applied to help the training get stabilized by normalizing feature statistics for every channel for each instance. Each feature extraction blocks double the number of feature maps. The final convolution layer flattens all the layers and reduces output dimension to 1×1 to output a scalar value for each image. The forward pass of the critic passes a batch of images through the critic layers and gets the generated scalar values in the final convolution layer as return value. Unlike DCGAN, WGAN does not have a sigmoid activation function because it doesn't need probability counts, rather it directly optimizes the Wasserstein distance.

We calculate the gradient penalty inside a custom made function called “compute_gradient_penalty”. The reason for calculating the gradient penalty is to perform the 1-Lipschitz constraint on the critic by penalizing large gradients. The function takes the critic model as a parameter. Along with that, it takes a batch of real images and a batch of synthesized images generated by the generator and one device information as parameters. The device information denotes where the computation should be performed. A weighted parameter alpha is created with values in range $[0, 1]$. This alpha is used to generate random points between real and fake images. Interpolated images are computed afterward. With the equation:

$$interpolated = \alpha \cdot real + (1 - \alpha) \cdot fake \quad (4.11)$$

.requires_grad_(True) has a boolean value true set inside so that gradients for these interpolated images are tracked by PyTorch, as it will be needed to calculate gradients later on. A variable named “mixed_scores” passes the interpolated image through critic to get critic scores on the interpolated images. Gradients are computed afterwards inside the gradient variable. Where inputs are the interpolated images, meaning we are computing gradients for interpolated images. Output is the gradients that we need, which is “of mixed_scores”. The parameter “create_graph” is set to true so that second-order derivatives are computed for backpropagation. The value of “retain_graph” is set to true as well so that the computation graph stays in memory for multiple iterations. The gradient is stored in the first output of the variable. After that the gradient is reshaped into a 2D tensor. Then the Euclidean norm or L2 norm is computed for each sample's gradient. By this it is measured how much the critic function varies for a small change in the input. As the WGAN-GP requires gradient norm to be close to 1 because it satisfies the 1-Lipschitz constraint, the gradient penalty is computed by penalizing gradients that derive from 1. Formula:

$$\lambda \cdot \mathbb{E}[(\|\nabla f(x)\|_2 - 1)^2] \quad (4.12)$$

After that the computed gradient penalty is returned which gets added to the Wasserstein distance during training.

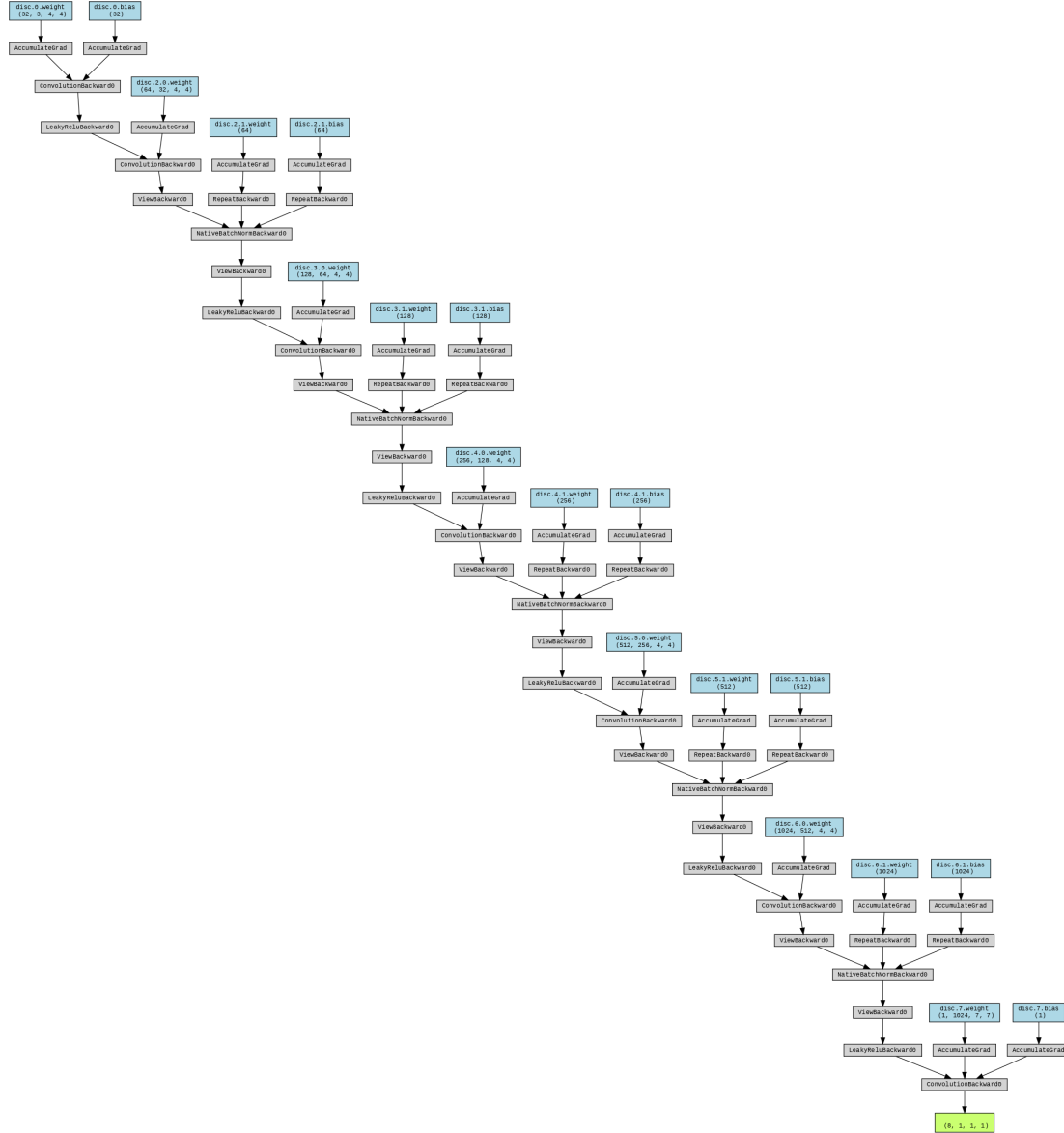


Figure 4.3: Critic Architecture of WGAN

The Generator

The generator takes a noise vector as input from the latent space and gradually transforms it to an image close to the images provided for training. In the input block, the latent vector or the noise vector expands it into a feature map using transposed convolution. The kernel size used in this layer is 7. After that we have a total of six upsampling blocks. Each of these blocks focuses on reducing the depth of the feature maps and doubling the output spatial dimensions. The final transposed convolution layer outputs an image of size $3 \times 448 \times 448$. Here 3 denotes channel number, RGB for our scenario. And the generated image is 448×448 in height and width. A constant padding layer is used to add 1 pixel padding to match the required 450×450 pixel resolution. A $\text{Tanh}()$ activation layer is used after that for keeping the pixel values in the range of $[-1, 1]$. The forward pass of the generator

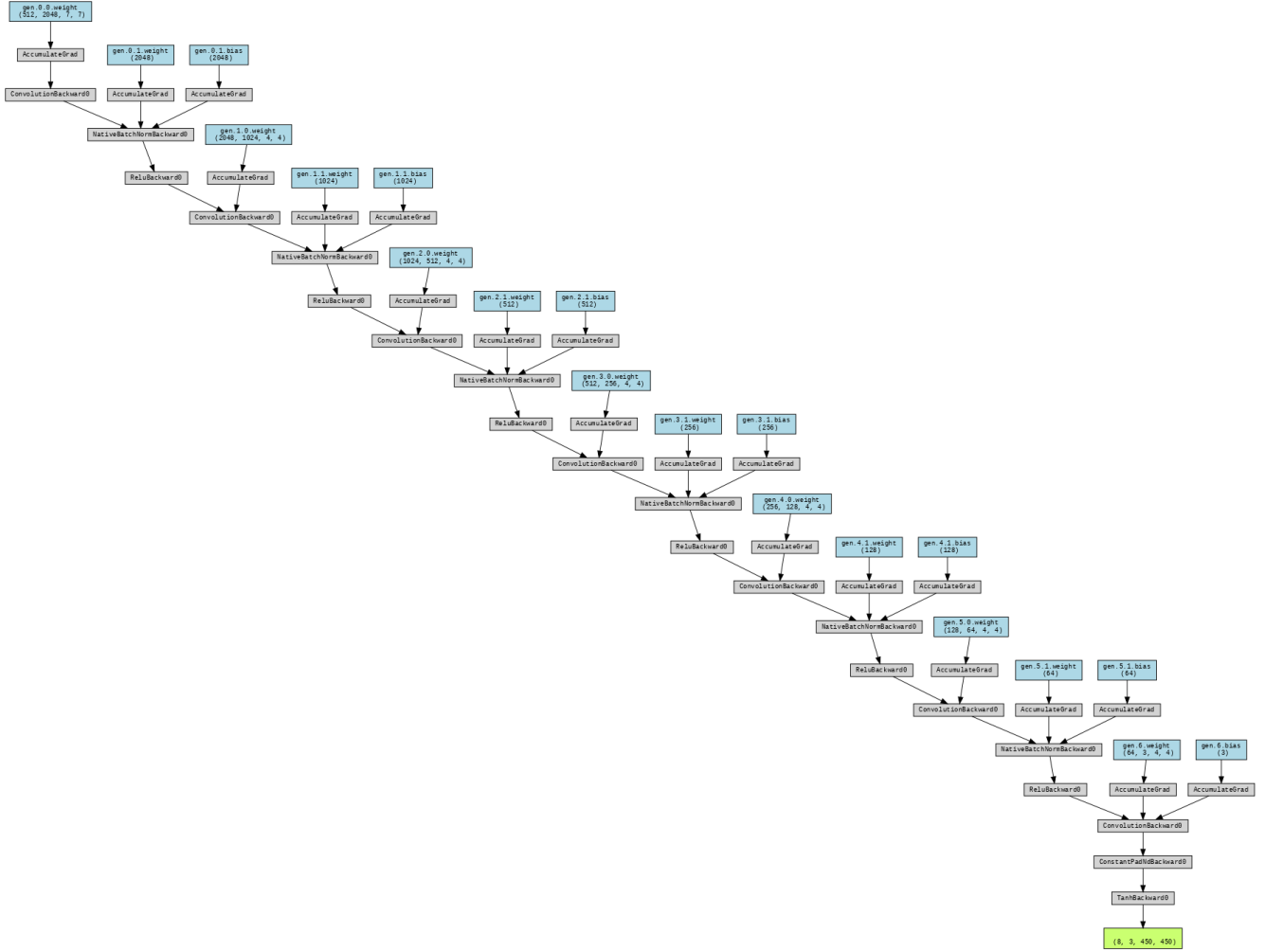


Figure 4.4: Generator Architecture of WGAN

passes the noise vector to the generator's input layer and we get a synthesized image as an output.

4.2.3 ResNet50 and Grad-CAM

ResNet50

In addition to prior segments, ResNet50 is a powerful architecture composed of deeper network layers that has the capability to solve issues that general convolutional networks had, such as the vanishing gradient problem which leads to insatiable training and lower accuracy. ResNet50 is able to address all such problems by having deeper network layers which consist of convolutional layers, convolutional and residual blocks and then finally fully connected layers [13]. At first, the input image is passed through a 7×7 convolutional layer containing 64 filters with stride 2. It is then followed by a 3×3 maxpool layer of stride 2. The initial convolution layer helps extract the hierarchical features of the input and the pooling layer down-sampled it spatially for further training. The image is then fed through multiple convolutional blocks and residual blocks during the training process. The convolu-

tional blocks consist of several convolutional layers, subsequently being followed by batch normalization, ReLU activation functions and skip connection. These help maintain stable architecture and better training performance and introduce non-linearity. The residual blocks follow bottleneck architecture, the core contribution of ResNet50, which contains 1×1 convolutional layers to reduce spatial dimension. It is then followed by another 3×3 convolutional layer to extract image features and then finally another 1×1 convolutional layer to restore the dimensions which were reduced in the first place. It allows mapping one layer with another which skips some connections in between due to the presence of skip connection; which ultimately helps build a deeper framework while mitigating gradient vanishing problems and allowing smoother gradient flow throughout the networks. It is then followed by identity blocks which mainly help create the skip connections and learn residual functions which map the input directly to the output of convolutional layers.

$$F(x) := H(x) - x \quad (4.13)$$

$H(x)$ is the underlying mapping that we desire to learn, whereas $F(x)$ is the residual learning. Rather than learning $H(x)$ from scratch, adding input back up with the residual learning helps achieve a similar desired outcome as it's easier to optimize [13]. The final part is fully connected layers which do the final classification. The base ResNet50 model employs a global average pooling layer to reduce the spatial dimension for final feature mapping. After that to predict final class probabilities softmax activation function is used [28].

Grad-CAM

Grad-CAM's architecture uses a CNN model as its backbone. The CNN model has to be already trained on the target dataset to serve as a backbone for hierarchical feature extraction. Grad-CAM then uses the knowledge learned by those models without modifying the model's workings. Then Grad-CAM targets some specific layers of the CNN model specially the last convolutional layer which provides significant information about feature extraction while retaining the spatial dimensions, meaning a possible link can be created between the output and input images to highlight important regions of the input images. Then the Grad-CAM framework uses the forward pass technique to record the feature maps of target layers by making an input image feed through the CNN model and also records the model's prediction. The recorded feature maps contain information about the CNN model's ability to detect specific features from different spatial locations. The core function of the Grad-CAM is its backward pass technique. Using back-pass methods the gradients of a specific class are recorded. The gradient calculation is done by back-propagating them only through the chosen target layers rather than making them go throughout the entire network. Then the gradient score of that specific class is calculated with respect to the corresponding layers' recorded feature maps [16]. Later to calculate the importance of each feature map as shown in Equation 4.14, global average pool-

ing is applied to these gradients which yields gradient weights corresponding to the feature maps of that specific class. These weights define how changes made in any spatial region of those maps can impact the prediction of the model. Again, to combine all the feature maps, the weights are multiplied by their corresponding maps and summed up together. The map is then passed through the ReLU activation function and batch normalization to help create the final heat map containing the positively correlated features and normalized values. These heatmaps are then overlaid upon the input images to identify and highlight the regions which contributed towards the CNN model’s decision-making process [16].

$$\alpha_k^c = \frac{1}{Z} \overbrace{\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \quad (4.14)$$

4.3 Model Implementation/ Training Details

We had already divided our dataset into 7 different folders based on their respective classes in the preprocessing stage. Once the architecture was finalized for DCGAN, we moved on to generating images using the DCGAN Model at first. In the second phase, we moved to finalize the Wasserstein GAN model and its core architecture adapting per our requirements. We generated images using the Wasserstein GAN in this phase. Finally, we augmented the original dataset with the images generated with Wasserstein GAN. Then we trained a classification model on both the original and extended dataset and applied Grad-CAM on the latter. A pre-trained ResNet50 was used in the classification phase.

4.3.1 Deep Convolutional Generative Adversarial Network

We started the training by picking the lowest, in terms of number of instances it has, class among all 7 - Dermatofibroma (df). It had only 115 numbers of images to begin with. For both the discriminator and the generator, we used the Binary Cross Entropy as the loss function. We used BCE for this model since it has been widely considered to be a standard loss function for most of the binary classification tasks. And, for a matter of fact, our discriminator is making a binary decision whether an image belongs to real distribution or not.

$$L = -\frac{1}{N} \sum_{i=1}^N (y_i \log(p_i) + (1 - y_i) \log(1 - p_i)) \quad (4.15)$$

For the Batch Normalization, we used the momentum as 0.8. We chose this high value to mainly update the running mean and variance slowly, giving more weight to the previous statistics for learning. Although we tweaked it multiple times, the high value seemed okay for our use case. For ReLU and Leaky ReLU activation

functions, we chose the slope to be 0.2. This is commonly used in most other GAN models and worked well for us as well.

We used Adam Optimizer as the optimizer for both the generator and discriminator training. We chose it as it is a widely used optimizer for most of the GAN models and also has a capability to adapt learning rates for each parameter [11]. This is a crucial ability to have in order to have faster convergence and ultimately good performance in general.

During initial training of 64 x 64 size images, we found a learning rate of 0.002 really good for both the discriminator and generator. In that setting, we used beta values 0.5 and 0.999. But, when we moved to train and generate the bigger sized images, we found the learning rate not working as expected. We tried and tested multiple learning rates ranging as high as 0.01 to as low as 0.0000001, for 450 x 450 size images. The beta values were kept the same for all cases. For all kinds of learning rates, in the stated range, we tried, the training was found to be very unstable. 0.00001 was the last used learning rate for both the discriminator and generator, before we moved on to train on images using Wasserstein GAN.

While keeping the learning rate to 0.00001, we used multiple techniques to combat the training instability:

Adding Noise to Real Images

We felt the discriminator probably was becoming more powerful, compared to the generator - which was causing the ultimate instability. So, we used a technique to add a certain percentage of noise to the real distribution so that the difference between real and fake distribution becomes very less distant. We added a 10% random noise to the real distribution during the training of the discriminator, so that it becomes trickier and tougher for the discriminator to determine the difference between a real image and fake image, ultimately taking more time in learning and more accurate learning as well.

Label Smoothing

We used label smoothing in the discriminator in making its probabilistic guess. We usually set the target label for real images as 1 and fake images for 0. But, here in this approach, we set the target label to be 0.9 for real images and 0.1 to be the same for fake images. These smoothed labels ensured that the discriminator becomes less certain in decision making when predicting the image. That way, the discriminator would not make a prediction for a real image to be perfect 1 and fake image to be perfect 0. This, in a way, helps the discriminator to not be way more powerful than the generator.

All these techniques and tweakings were made into our unsupervised DCGAN training to stabilize the training and make better generation of images. The addition of noise and smoothed labels helped in our training a lot, that we will discuss more in the results section. But, this model seemed to remain unstable for generating 450

$\times 450$ size images per our requirements. And thus we moved on to training Wasserstein GAN on the same dataset and with the same requirement in place.

4.3.2 Wasserstein Generative Adversarial Network

Hyperparameters

Initially we define a set of parameters for our model training. We needed to produce images having pixel size 450×450 . Hence, the IMAGE_SIZE parameter is set to 450. Images are of RGB format, therefore CHANNELS_IMG parameter is set to the value 3. The value of the parameter NUM_EPOCH differed from one class to another. We used a total of 700 epochs for the category having the lowest count of images, Dermatofibroma. We used 550 epochs for the category having the second lowest count in the dataset, Vascular lesions, and 500 epochs for the category Actinic keratoses. We set our learning rate to 0.0001 for the optimizers. We set the weight for gradient penalty terms to 10. We set our critic iteration number to 5. Meaning, the critic gets 5 times training per generator train. It is needed to train the critic well before each generator update so that the loss function becomes an estimation of the EM distance. We have used the same value as the base paper of WGAN-GP for our model training. We set the latent space dimension value to 512. Base number of feature maps in both generator and the critic is set to 32.

Data Transformation

We defined a transform variable. A series of transformations was pipelined and stored into this specific variable. Our dataset had images of size 600×450 . We used the Resize((IMAGE_SIZE, IMAGE_SIZE)) parameter to resize these images to the uniform dimension of 450×450 . The ToTensor() parameter was used to convert the PIL images to PyTorch tensor. Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5]) parameters have been used to normalize the image tensors using mean values and standard deviation values. The mean is [0.5, 0.5, 0.5], the standard deviation is [0.5, 0.5, 0.5] for each RGB channel. By this normalization process, pixel values get scaled to a range of $[-1, 1]$ from the range of $[0, 1]$.

Dataset Loading

We created a custom ImageDataset class to load image files from folders, apply transformations on them and return them as tensors. In the constructor of this class, two parameters need to get passed. One is the root directory, holding the information of where the images are stored. Another is the transform parameter to pass the transformation pipeline. The constructor has one instance variable self.images where it stores all the image file names in a list. The len method returns the number of images in the dataset and allows PyTorch's data loader to know how many samples are available. The getitem method retrieves the file path of the images. Open the images using a PIL library. Convert it to its RGB format. Then puts through the transformation pipeline and returns the processed image tensor. An object of the ImageDataset class gets created after calling the class inside the "dataset" variable and gets stored inside it. After that a PyTorch Dataloader is created inside the variable "loader". We set the batch size to 8 while setting ini-

tial hyper parameters and it gets used here while creating the PyTorch Dataloader. Shuffle value was set to true, so that the dataset gets randomly shuffled at every epoch to improve training and capturing information variation. Num of workers was set to 0, denoting data loading should happen on the main thread of the CPU. Pin memory was set to true as well, ensuring faster GPU transfers. Lastly, drop last value was set to true so that all batches remain of equal size and the last batch gets dropped if it has a batch size smaller than the given batch size.

Defining The Optimizer

We defined a device parameter that checks if CUDA compatible GPU is available, which returns a boolean to the variable. Based on that boolean value, a PyTorch object specifies where the tensors and models should be stored; `torch.device("cuda")` is selected if a GPU is available. Otherwise, `torch.device("cpu")` gets selected as default. After that, an instance of the generator class is created and transferred to the specified device. Similarly, an instance of the critic is created and sent to the specified device as well. After that we define the optimizers for the critic and the generator respectively. We used the Adam optimizer for updating parameters during training. Beta value 0.0 was used as a first order momentum term which decays past gradients. Beta value 0.9 was used as a second order momentum term which helps with stable training and avoids mode collapse.

4.3.3 ResNet50 and Grad-CAM

ResNet50

We used ResNet50 to validate the accuracy of our dataset. After data augmentation for implementation we followed the transfer learning method and used pretrained weights using FastAI's module vision learner. This vision learner module was already pretrained on imagenet dataset which provides a strong backbone making it an efficient classification model to be used for transfer learning. At first we imported all necessary libraries from fastai to implement our task. Later on to define our data loader, we created an object of DataBlock class named "cancer_cells". While creating the blueprint of our data loader, we sent various parameters inside the DataBlock. At first, blocks parameter is sent where ImageBlock means that our input was image data and output is CategoryBlock or categorical. The `get_items` parameter holds the path where input images were being retrieved from the specified directory. Using the random splitter we divided our dataset into validation and train subsets, where `valid_pct` was set to 0.2, which means that 80 percent data would be used for training and 20 percent for validation. Here seed was set to 42 to ensure models ability to reproduce consistent results upon multiple training sessions. The input images were resized into 128×128 by passing the value 128 through `RandomResizedCrop`. Additionally `aug_transforms()` parameter was passed to create variation of input data at batch-level for training purpose, and with that the creation of DataBlock was complete. Then the data loader was created using the DataBlock object upon calling the `dataloaders` method where a specified directory for the dataset was passed as the parameter. Finally for the training process a learner object is created through `vision_learner` function where we sent our data

loader as the first parameter, ResNet50 as the second parameter and accuracy metrics as third parameter to evaluate the overall performance. Then the learned object was fine tuned for 150 epochs to ensure training on our dataset. After training is done, we called our self defined function to calculate and visualize per class metrics such as f1 score, precision, recall, sensitivity and specificity.

Grad-CAM

To implement the grad-cam model, we trained resnet50 using our dataset and then saved the .pkl files to use later on. Then we created a self-defined Grad-Cam class and some helper functions `get_resnet_cam_layer()`, `apply_gradcam()` and `visualize_gradcam()` to implement this explainable ai approach properly.

The GradCam class

Our custom made grad-cam class consists of four methods. The constructor, `save_activation`, `save_gradient` and finally `__call__`. Within the class constructor, we defined instance variables to save the backbone model, target layers, activations and gradients. Then the built-in pytorch modules `forward_hook()` and `backward_hook()` were called with respect to the target layers to save the activation maps and gradients respectively. During the forward pass, the `save_activation` method takes the target layer's output. The output is then detached from the computational graph to avoid irrelevant gradient calculations. Then the output activations are stored inside an instance variable named `self.activation`. Similarly, during the backward pass, we need to capture the significance of feature maps and for that, we need to capture the gradients. For this task the `save_gradient` method is implemented which takes the gradient output of the target layer and detaches it from the computational graph then saves it inside `self.gradients`. Lastly, The `__call__` function helps calculate the gradient weights and weighted sum of the activations and returns them as a NumPy array. This function takes an image input and class index if provided otherwise set to "none" as default. It then activates the model's evaluation mode eventually performing forward pass and saving the predictions inside the output variable. If the class index is not provided it is then calculated from the output. Before the backward pass, `self.model.zero_grad()` is called to clear out all the stored gradients in previous steps. It ensures that the gradients calculated will only be of the target layers. Then the backward propagation is performed to capture the gradients from the target layers. Lastly, the saved gradients and activations are used to calculate the weighted gradients corresponding to each feature map and store them inside the "cam" variable. Later to help visualize the cam, ReLU activation and batch normalization are performed and the heatmap is generated in the form of a NumPy array.

The `get_resnet_cam_layer` Function

Except for the grad-cam class, another function implemented was `get_resnet_cam_layer` which helped find the target convolutional layers of our backbone ResNet50. We implemented it in such a way so that the function returns the last convolutional layer and if none is found it raises a `ValueError`.

The `apply_gradcam` Function

Finally, the `apply_gradcam` function was implemented. It takes a learning model, `image_path` as parameters and the `class_idx` is set to `none` as default. At first, it retrieves the image from the specified directory. It then applies preprocessing on the image using a transformation pipeline. The image is resized into a 224×224 tensor suitable for ResNet50 and then each pixel value is normalized using standard deviation and mean values computed previously from the training phase. The image is converted into an input tensor and it is also ensured that the model and the input are processed on the same device using “`learn.model.parameters().__next__().device`” this multiple sequential method calls. It then retrieves the target layer using `get_resnet_cam_layer` and creates an object of the GradCam class. Later the heatmap is retrieved by calling the object while passing the image retrieved earlier along with the target layer. The heatmap is then resized into 224×224 dimensions and also colorized using the `cv2.applyColorMap()` function. Lastly, the superimposed image originated by combining the original image and the heatmap. The `apply_gradcam` function then returns the original, heatmap and superimposed image.

The `visualize_gradcam` Function

Another helper function `visualize_gradcam` is implemented to help plot the outcomes after applying gradcam. It also takes a learning model, `image_path` as parameters and the `class_idx` and `save_path` is set to `none` as default. At first, our previously trained model’s .pkl files are loaded into a learner object. Then we called the `visualize_gradcam` function where we sent the learner object and specific image path as arguments. It then calls the `apply_gradcam` function and stores its results, which would be a superimposed image and heatmap for the original input image. It then helps plot those images to understand the grad-cam’s outcomes better.

Chapter 5

Result Analysis

In this section, we will talk about the resultant images obtained during the training of both DCGAN and Wasserstein GAN and also analyze how they fare in this specific task. Our goal is to analyze these two model's capabilities as well as their limitations in the context of generating images of bigger pixel sizes. We will take into account the accuracies obtained during the classification task performed on both the original dataset and the extended dataset with generated images and analyze further with explainable AI.

5.1 Image Generation

5.1.1 DCGAN

64×64 Pixel Images

In the early phase of our training we ran the DCGAN model with some slight changes to generate 64×64 images, before going to the actual training phase where we generated 450×450 size images. The goal for going towards the small size generation was to validate that we were in the right direction.

We changed the image size of the original dataset to 64×64 and with the changed architecture we ran DCGAN for a total of 50 epochs with a learning rate of 0.002. We kept track of the loss for both the generator and the discriminator network so that we can better understand how the model was learning with time.

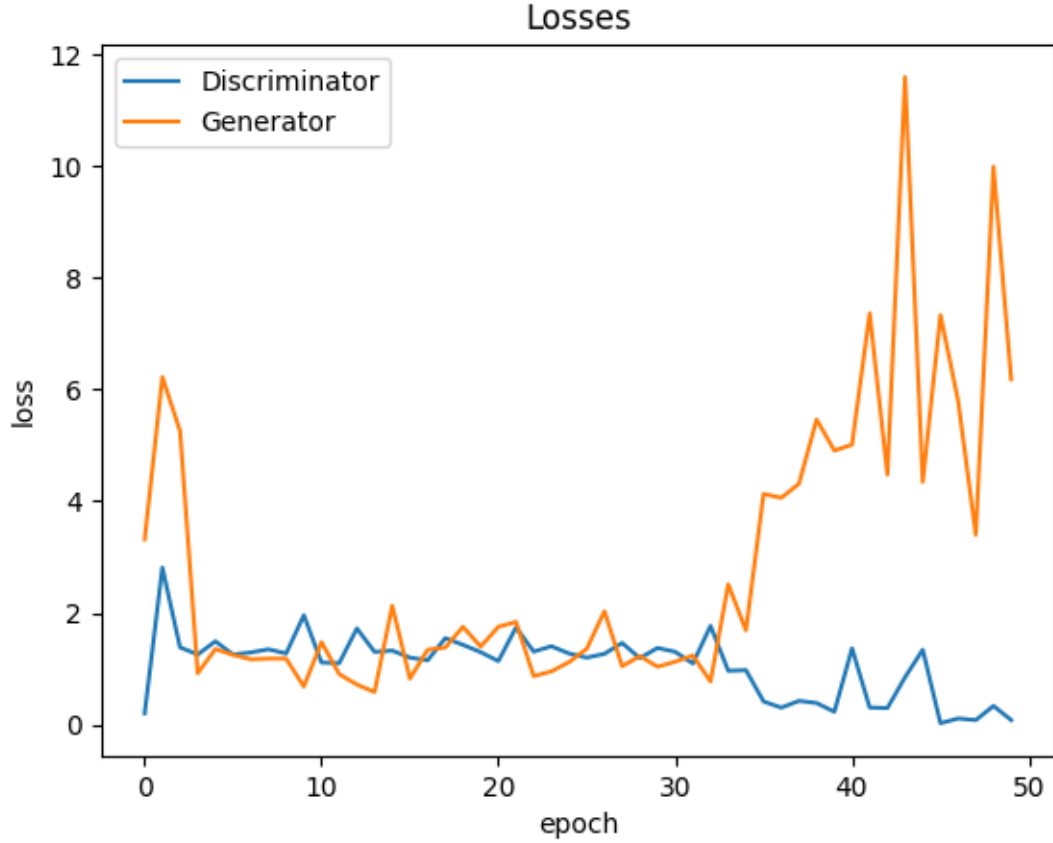


Figure 5.1: Loss against Epoch plot

After the very first epoch, the generator had a loss value around 3 which then jumped to over 6 after the second epoch. This was expected since, at the first epoch, both the generator had no idea how to generate a good image, while the discriminator had no idea to decide whether the generated image was from the training data or not. The noisy image generated by the generator was thus predicted to be an actual image by the discriminator. But, after the first epoch both the generator and discriminator started learning at their specified tasks which can be clearly visible in the graph. As more epochs went by, we can see that the loss for the generator started decreasing. Although, we can see that there are fluctuations in the loss throughout the graph, but that is because we chose a slightly big learning rate. Due to the larger value, it was taking large steps causing the fluctuations.

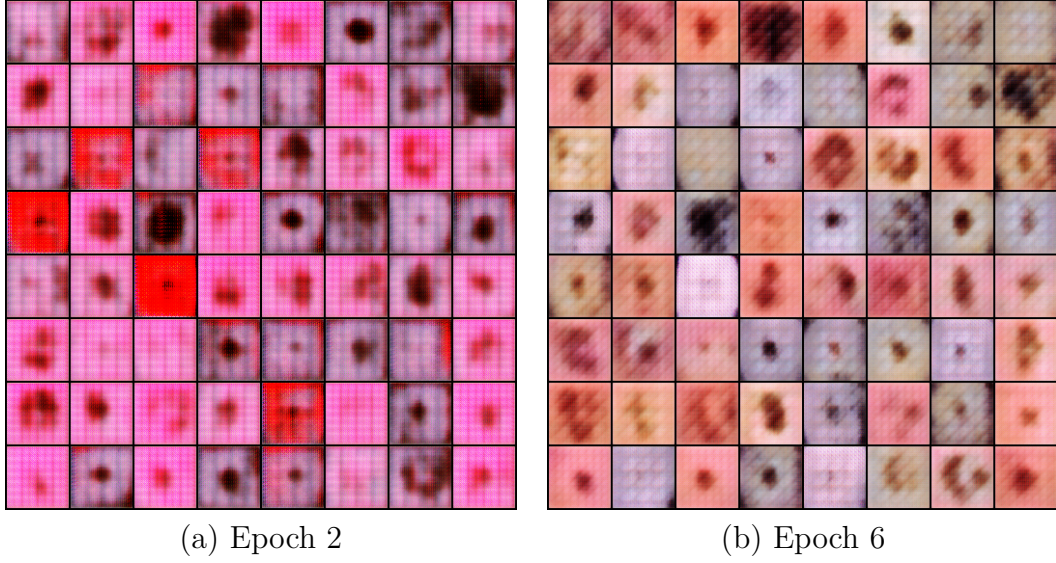


Figure 5.2: Comparison of DCGAN-generated images at 2nd and 6th epochs

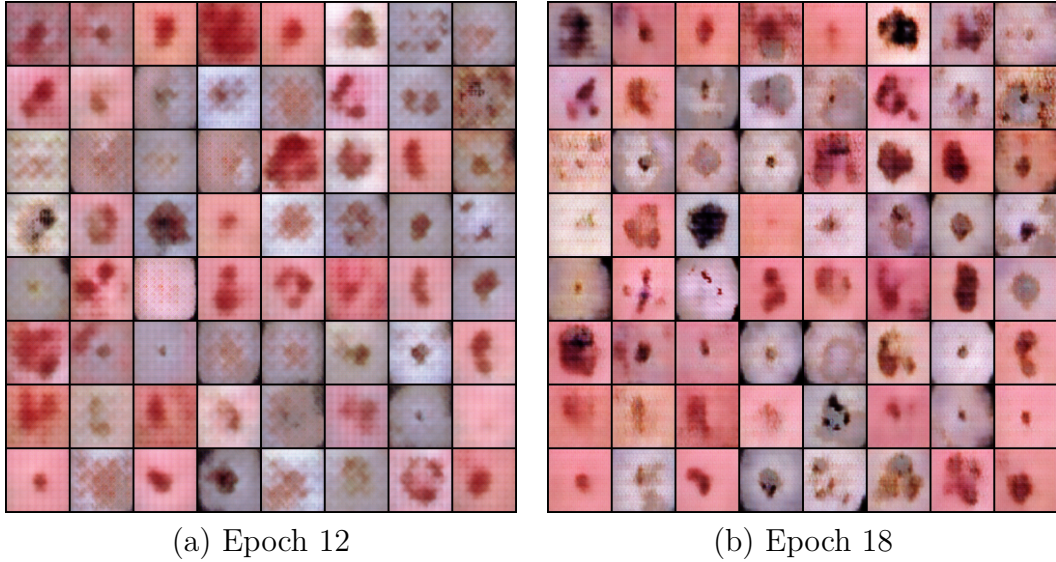


Figure 5.3: Comparison of DCGAN-generated images at 12th and 18th epochs

The images generated after each epoch also visually tells us that the model performed better as the training time went by. As we can see the images generated after the first few epochs were very noisy. But if we look at how the generated images transition over time, particularly after epoch 18, it could be seen clearly the improvement of the model. Eventually as the training went ahead, the model only got to learn more nuanced details of the real distribution.

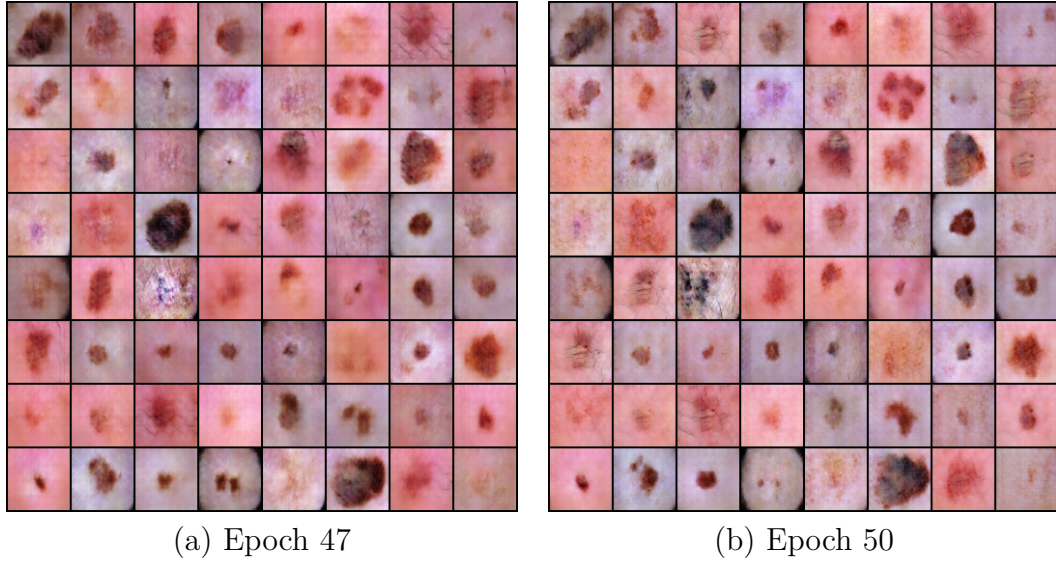


Figure 5.4: Comparison of DCGAN-generated images at 47th and 50th epochs

As the training was satisfactory, we jumped towards our primary goal for generating images keeping its original size intact, which is 450×450 pixels.

450×450 Pixel Images

The initial training of DCGAN, with the discussed architecture, was first performed on the class that had the lowest number of images. Dermatofibroma (df), although had a total of 115 images, was augmented into 1150 images during the preprocessing period. As we began our training, the training seemed to look very unstable. The generator would take a bit long to get the structure of the real images and when it does, after a few epochs, it suddenly just stops learning.

So we started tracking the real and fake score along with the loss values as we continued the training. We found out that the real score - which denotes the discriminator's mean probability of it being convinced the real image belongs to the real images, was consistently turning up as 1.0 after some 100 odd number of epochs. The discriminator loss also became 0 at the same time. Once, the real score became 1.0 and the fake score became 0.0, the next subsequent epochs also stuck there with the same real score, fake score and loss values. This meant that the discriminator was choosing all real images as real and all generated images as fake. The generator could not fight the discriminator to fool it. So we witnessed a vanishing gradient problem, a very popular problem in such models.

To combat this issue, we introduced two techniques, as specified in the Chapter 4, - addition of random noise into the real images and label smoothing. As we began re-training with all these techniques applied in our core model, the training seemed to start smoothly. The generated images over the first few epochs seemed that the model was learning the structure and the artifacts of the images. The images looked better than the one generated during the previous epoch. But things turned op-

posite again. After some epoch, we saw the model generate a random noisy image again. This seemed similar to the image from the very first epoch - just random noises. Then as the training progressed, the model again started getting better. Until again, after some epoch, it generated a noisy image again.

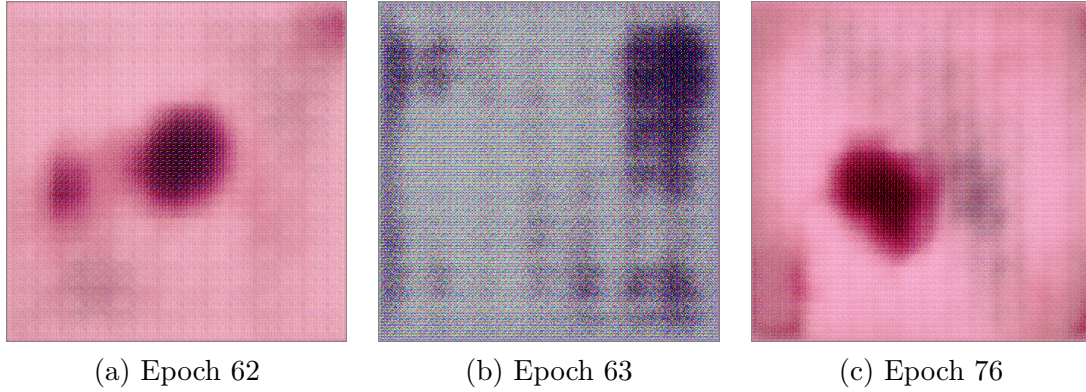


Figure 5.5: Comparison of 450 x 450 pixels DCGAN-generated images at different epochs

We decreased the learning rate in the hope that probably the model required a smaller rate for it to be stable. But, it did not work. We kept decreasing the learning rate as low as till 0.0000001, but this recurring result seemed to be still there. We then decided that it would be better to move on to Wasserstein GAN now. The DCGAN seemed to have hit the wall for our case.

5.1.2 Wasserstein GAN

The training of Wasserstein GAN seemed to look better for our case. We ran the model in three separated phases for three different classes. All these classes had the lowest number of instances.

The first phase of generation with the Dermatofibroma (df) class, showed better images from the very first epochs.

The loss curve for this training is not that impressive like the previous DCGAN 64×64 training. But that is because of the high pixel image regeneration in a low instance class, ultimately leading to mode collapse.

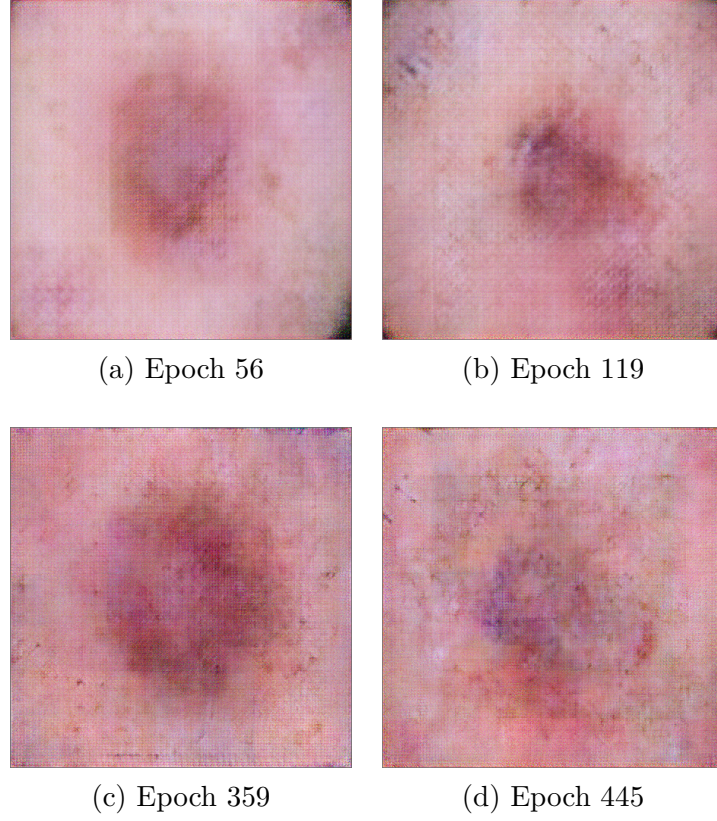


Figure 5.6: Comparison of WGAN-generated images at different epochs for df class

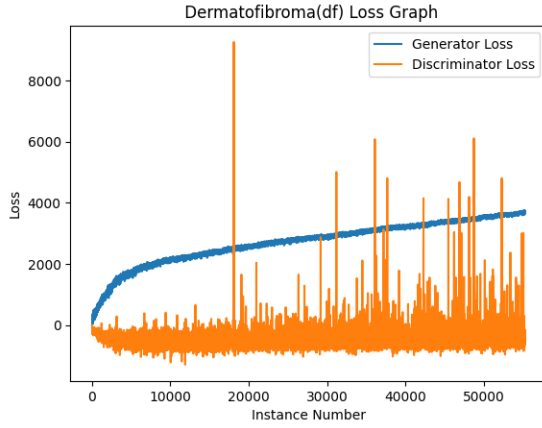


Figure 5.7: Loss vs Batch

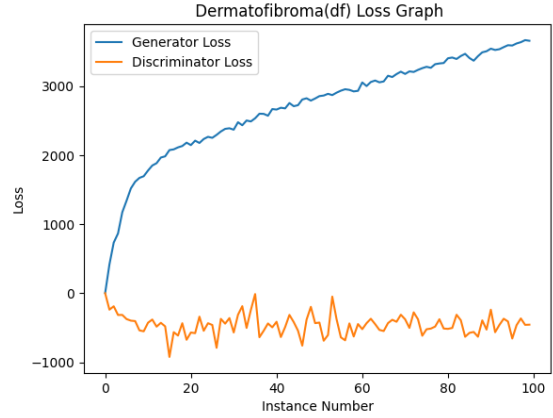


Figure 5.8: Loss Graph Progression

The Loss graph on the left is recorded after every batch of images for a total of 55300 times over the 700 epochs. On the other hand, the image on the left is the same loss graph recorded 100 times over a constant time interval. Ultimately, over these 700 epochs, we generated a total of 700 images.

During the second phase, we chose Vascular Lesions (vasc) - the class with 142 images in total. We ran the training for a total of 550 epochs to generate the same 550 instances of images.

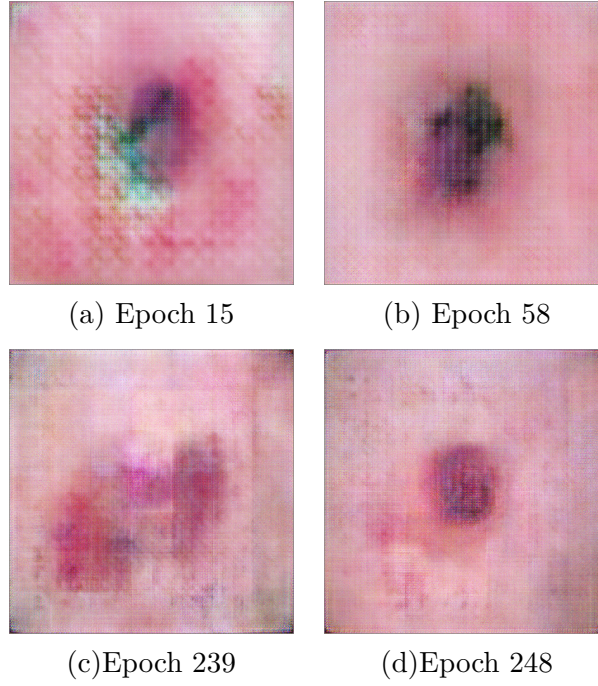


Figure 5.9: Comparison of WGAN-generated images at different epochs for vasc class

As usual, we noted the losses during the training to get a better idea of how the model was performing:

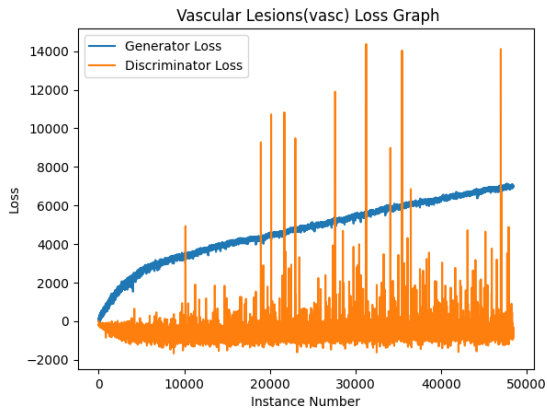


Figure 5.10: Loss vs Batch

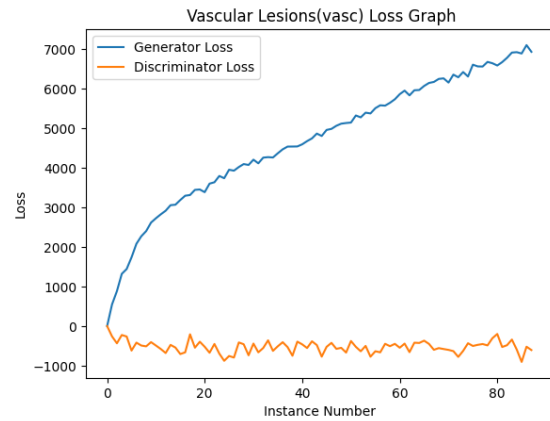


Figure 5.11: Loss Graph Progression

Finally, the class with the third lowest number of instances was chosen for the training. We ran the training on the Actinic Keratosis (akiec) class for a total of 500 epochs to generate the same 550 instances of images.

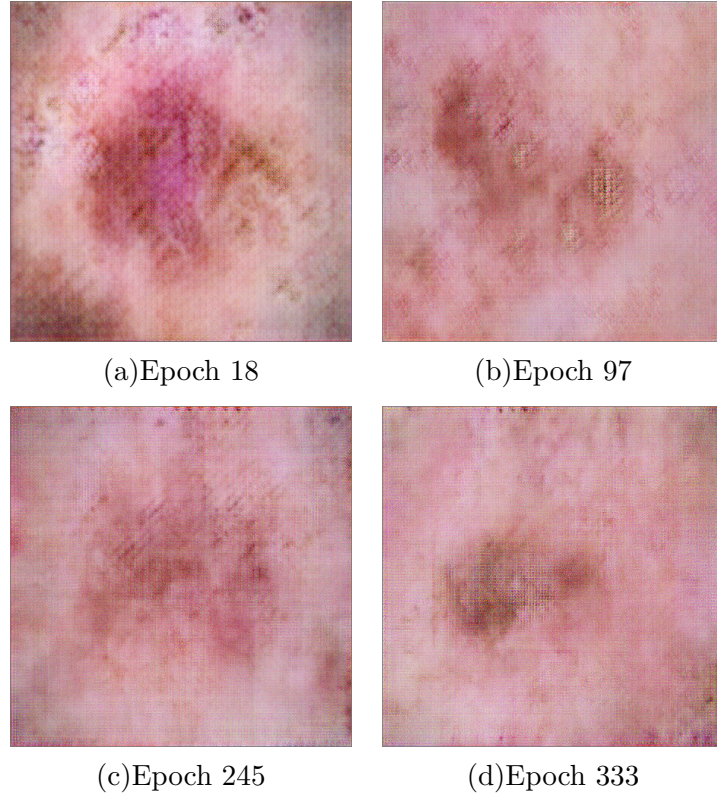


Figure 5.12: Comparison of WGAN-generated images at different epochs for akiec class

The recorded loss curve for this training given below is pretty similar to that of the previous two.

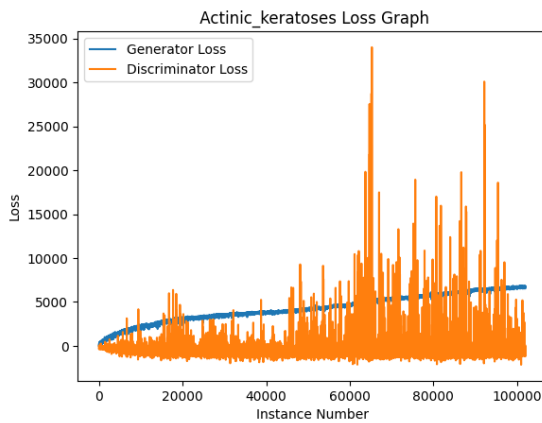


Figure 5.13: Loss vs Batch

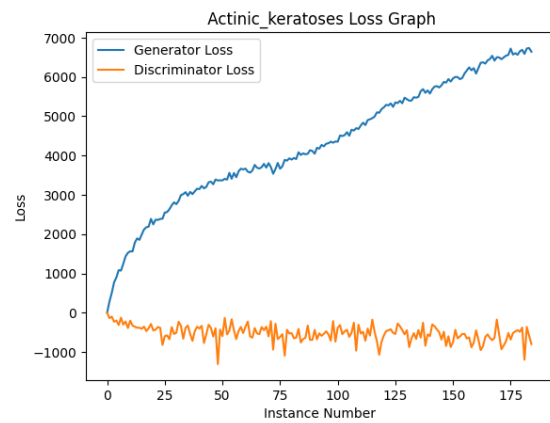


Figure 5.14: Loss Graph Progression

By the end of our training phase, we end up generating a total of 700 images for the class with the lowest number of instances, 550 for the second lowest and 500 for the third lowest cancel class.

5.2 Image Validation using ResNet50 and Grad-CAM

5.2.1 Classification on Original, Partial and Fully Augmented Dataset

As specified earlier, we first finetuned a pretrained ResNet50 model on our original dataset. This dataset only contained the real images and 80% were used for the training purpose. During this stage, the remaining 20% were used for validation purposes during the training. A total of 150 epochs were run for and, as well as the validation accuracy, the different metrics such as F1 Score, Precision, Recall and Specificity were recorded at the end of the training.

The same approach was used to fine tune another ResNet50 classification model, but this time around 100 – 200 images were chosen from the generated images of the three classes and augmented them later with the original dataset. We also ran this model for a total of 150 epochs and recorded the same accuracy as before.

Finally, all the images generated using Wasserstein GAN were used to augment the original dataset and a similar training as the previous ones were done on it from scratch.

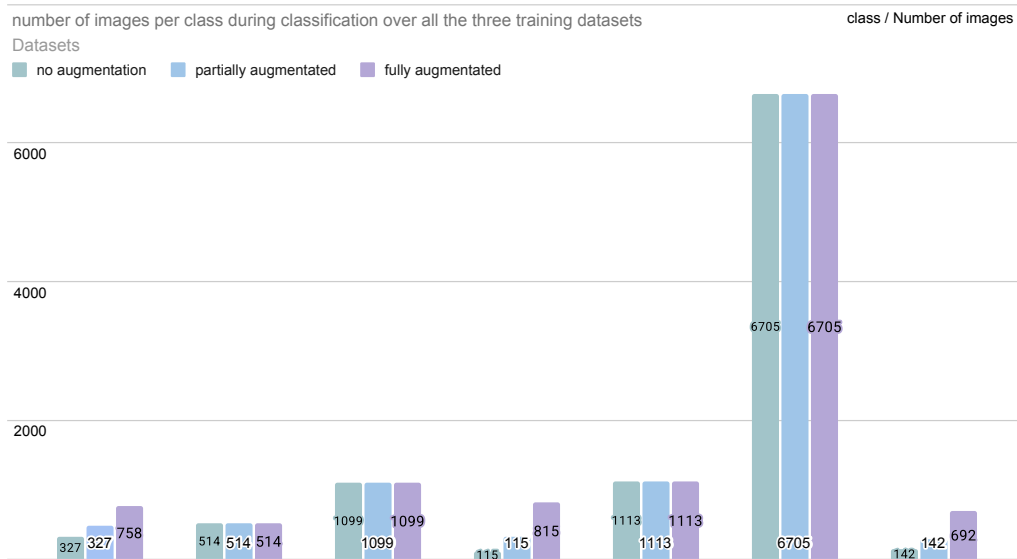


Figure 5.15: Column Chart to show Number of Images per Class during Classification over all three training datasets

Upon comparing the results from these three individual training, we found out that the validation accuracy was able to increase on the full augmented dataset compared to the original one.

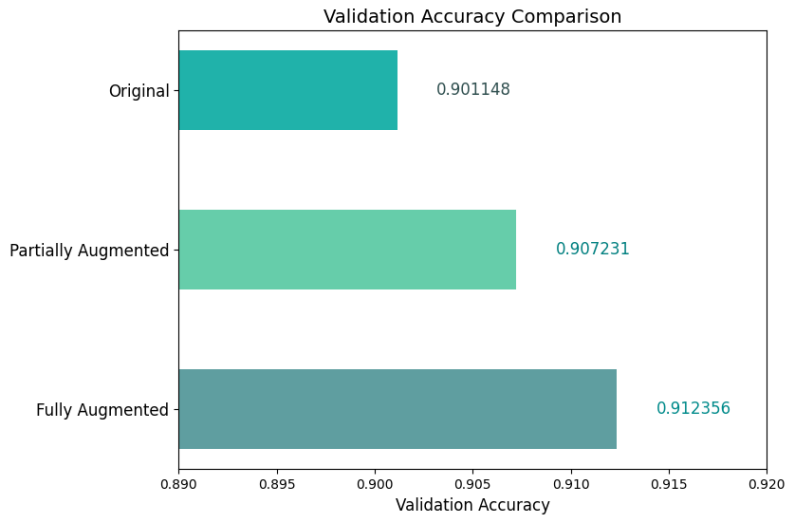


Figure 5.16: Bar Chart to show Validation Accuracy among three training datasets for all the three classes

All required values here: 150 epoch class wise folder accuracy files

The validation accuracy, which was found to be 90.11% during the classification on the original dataset, was found to be partially improved, at 90.72% when added with best images from the three classes. The significant improvement was found in the accuracy on the third model, which was run on a full augmented dataset. The accuracy increased to 91.23%, that is 1.12% in increment from the original one.

In a medical disease classification, Recall values for the model is the most important factor for making a good judgement on them. This, also called Sensitivity, denotes the proportion of true positives in classification. In the context of our case, we do not want our model to say the patient is safe, when, in reality, he does have skin cancer. So, usually, we want this value to be as high as possible denoting the effectiveness of the model.

In our training we found out that Dermatofibroma(df), which had a recall value of 0.95 in the original dataset, decreased to 0.94 once the best generated images were added, and eventually came to be 0.947 for the full augmented dataset. This was not ideally what we hoped for. But the improvement came in for the other two classes. Vascular Lesions(vasc) had a recall of 0.90 for the original dataset that increased to 0.91 and then to 0.98 for the partially augmented and fully augmented one respectively. Actinic keratoses(akiec) saw an increase from 0.61 to 0.69 and then to 0.82 eventually. This was considerably a huge improvement which is shown on Figure 5.17

We can see from the Figure 5.18, even for other metrics, like precision, the value increased when applied on the original dataset compared to on the fully augmented one. But, unlike the recall, we saw improvement for the Dermatofibroma as well here. It had earlier a precision value of 0.87 which increased to 0.94 and then 0.98

when the images were augmented onto the dataset. On the other hand, vascular lesions got to 1.00 from 0.87 when partial augmentation was done. Although, it decreased slightly when applied on the fully augmented dataset, as the precision came to 0.99. This is still negligible. The similar up and down is seen for actinic class as well - as the precision value went up to 0.98 from 0.86 when partial augmentation was performed. Then it went down to 0.93 for the fully augmented dataset, although it still had a better score than the original one.

For specificity as well, we can see some changes in the score on Figure 5.19, although these changes are very microscopic. With still very hard to comparable values, we can still see that the partially augmented dataset was slightly ahead of the other, if there was one.

For, F1 Score as shown on the Figure 5.20, the results are the most satisfying among the other scores. For each of the three classes we can see that the F1 score increased when partially some images were augmented. And then it increased further when all generated images were augmented to the original dataset. The F1 score can be termed a fine balance between the precision and recall values. So, having a good F1 score for all three classes can be a really good outcome.

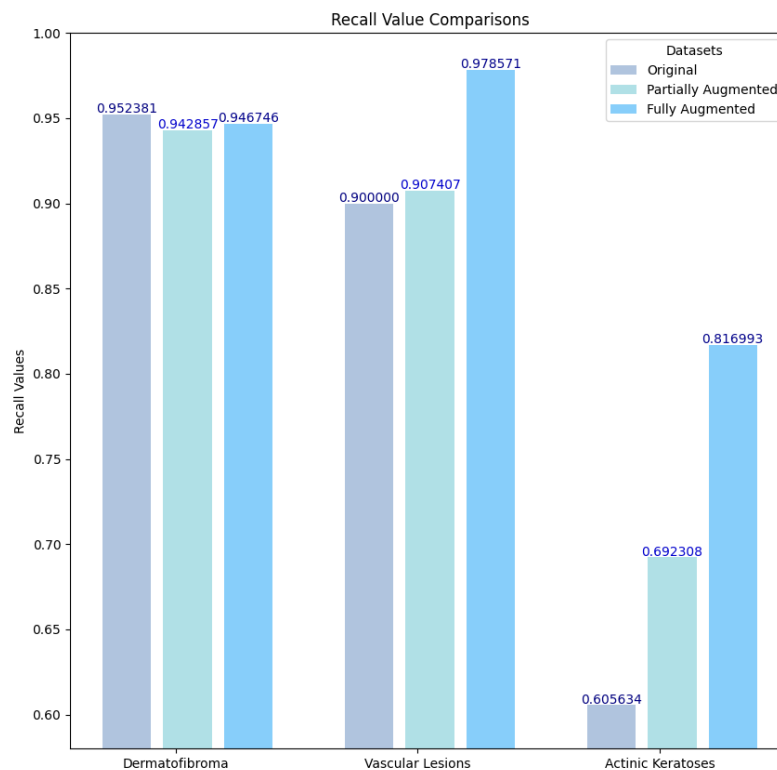


Figure 5.17: Bar Chart to show Recall among three training datasets for all the three classes

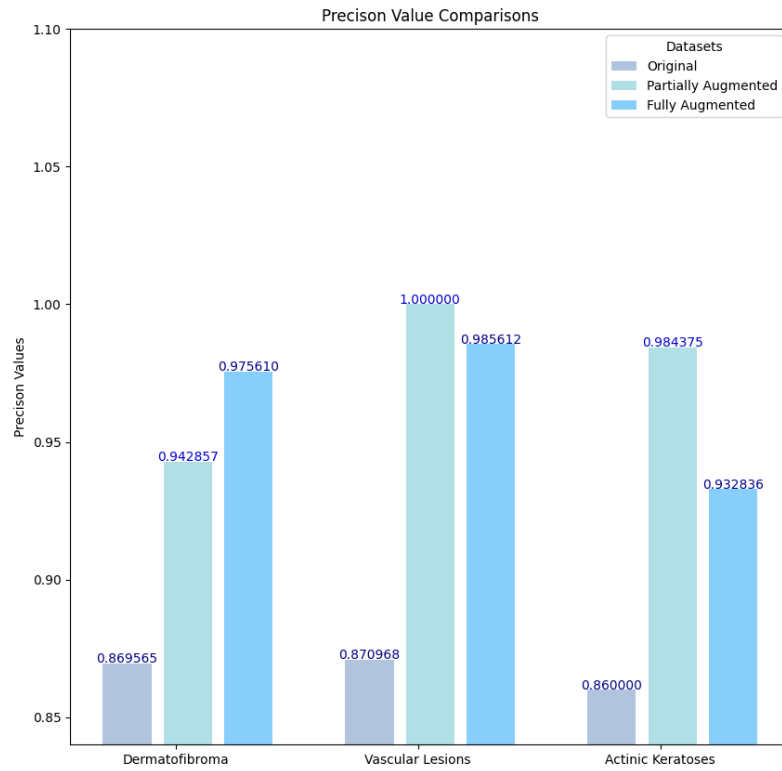


Figure 5.18: Bar Chart to show Precision among three training datasets for all the three classes

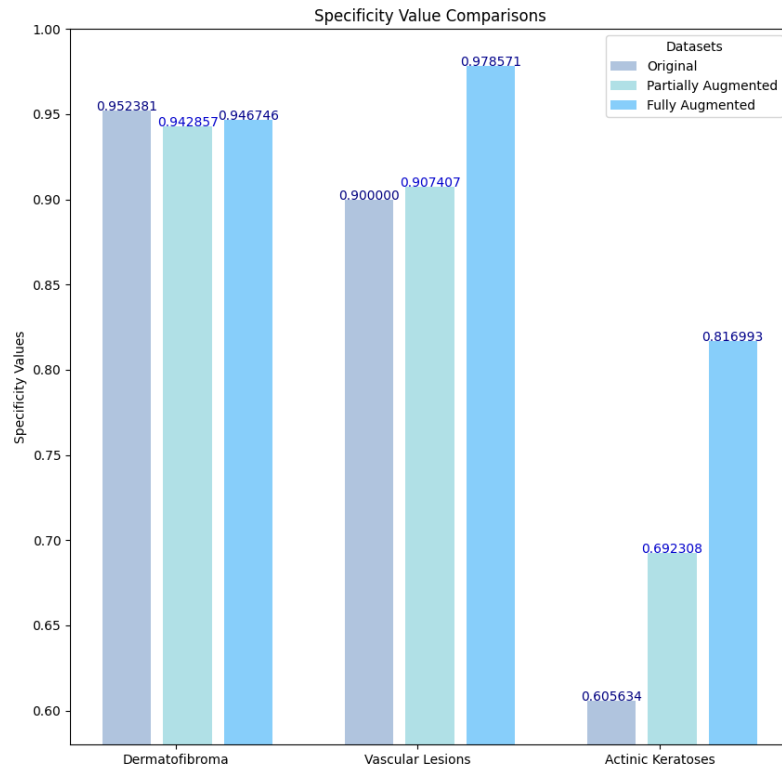


Figure 5.19: Bar Chart to show Specificity among three training datasets for all the three classes

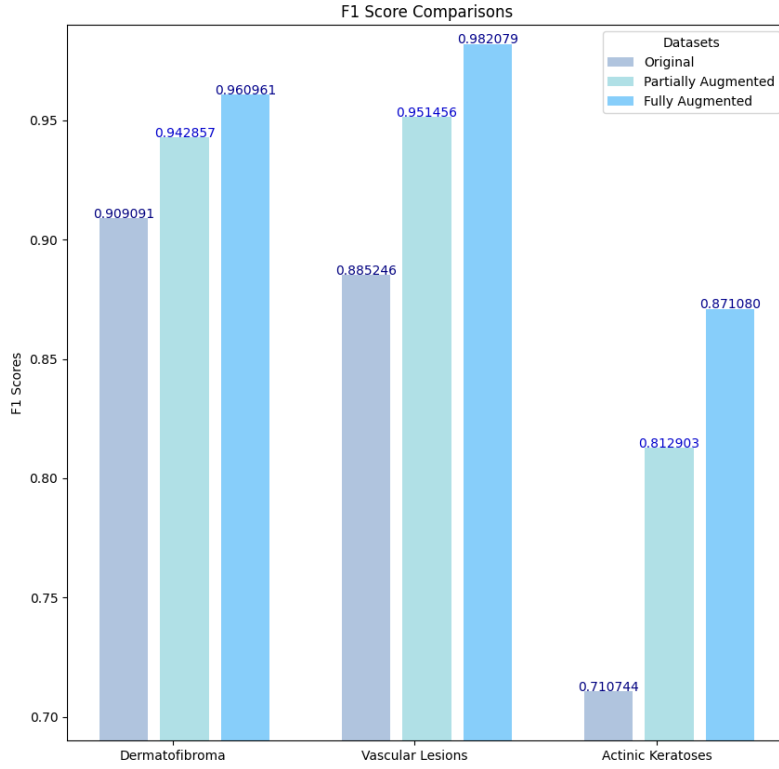


Figure 5.20: Bar Chart to show F1 Score among three training datasets for all the three classes

5.2.2 Validation using Grad-CAM

Other than these classification accuracies to judge our model, we also used Grad-CAM to validate whether the important features or artifacts inside the generated images are intact or not. We are using the same classification model that we previously trained on the fully augmented dataset for this step..

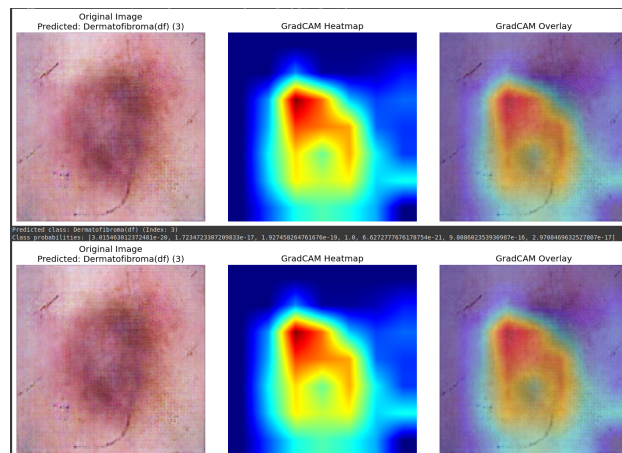


Figure 5.21: Grad-CAM applied to df class

Here the heatmap for the Actinic Keratosis (akiec) shows that it does not have any special artifact in the region it should have. This is a concern that requires improvement. Although the model does predict the image to be of akiec class, the heatmap is not as per the expectation.

5.2.3 Validation using the Official Test Set

We took a step further and also tested our fully augmented dataset on the official test set available on the ISIC archive for the 2018 challenge.

We first retrained our model on the entire original dataset using the same model as previous classification tasks - ResNet50. We used the entire 100% original dataset for this training purpose. Parallely, we also trained two other ResNet50s on the fully augmented and partially augmented dataset. For this as well, we used the entire extended dataset for this training.

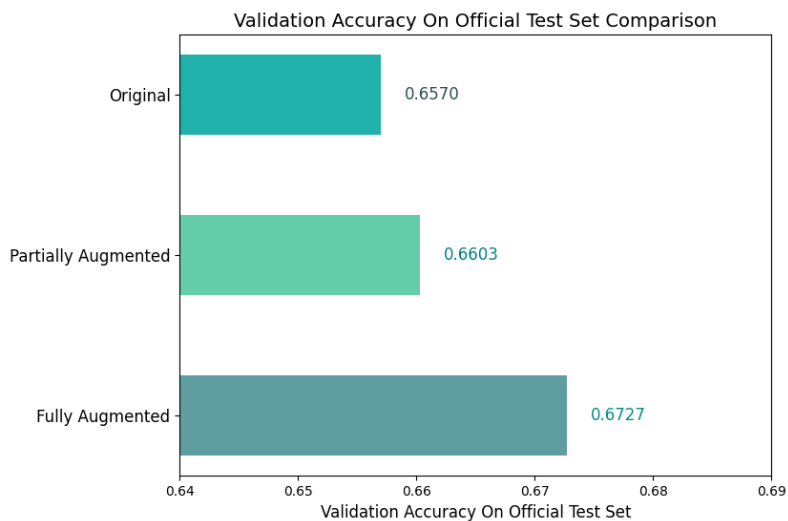


Figure 5.24: Bar Chart to show Validation Accuracy among three official test datasets for all the three classes

Now, we used these three classification models to predict classes for the test set as shown on Figure 5.24. The model trained on the original dataset showed an accuracy of 65.70% on the test set. On the other hand, the model trained on the partial augmented dataset showed an increased accuracy of 66.03%. Then finally the fully augmented dataset showed an accuracy of 67.27%, beating the previous two models. That is an increment of 1.57% of accuracy from the original one.

Chapter 6

Discussion

Looking at the results obtained from both DCGAN and Wasserstein GAN training for 450×450 sized images tells us that although Wasserstein GAN tries to combat and solve all the mode collapse, vanishing gradient and instability issues faced in DCGAN, it still is not a good model to choose for dealing with bigger size images. Medical Imaging is a critical field and it is required to generate images that are of actual original size and of the same quality as the original. Our model although, generated images that are ultimately increasing the classification accuracy, still these images are not the kind of images that should be used in the real world for such critical tasks involving life and death. But, back to the technical aspect, we believe the choice of DCGAN in such a bigger size image generation is not a good choice. Although it works just as great for generating 64×64 size images. We saw that in the prior training period. Even the 128×128 images, we believe, could have been generated better with this model. But the instability of the model is certainly not only due to the bigger size, but also the training size. We are generating images on a very small set of images. 115,142 or even 327 are a very small number for training such image generation models, that too in big size. Although we did geometric augmentation in these classes to increase the class size, they still are not comparable to having more unique images. So, the model's infamous instability and the low number of instances are the key reasons for not getting the up to the mark result. But the kind of techniques we used in the DCGAN, such as label smoothing and adding extra noises to the generator - proved to be really helpful techniques for improving stability in the model. Similar things can be done with Wasserstein GAN as well, we believe, that may have brought better results. The losses for both the generator and the critic for the Wasserstein GAN training shows that the model did not perform per the expectation. Although we did get some okay images, we saw the images lacked variety a lot. That is a mode collapse issue in general. In a perfect and ideal training setting, the generator should see a decrease in its loss value, which is not something we saw in our generator loss graph. Instead, our generator saw an increase in its loss value over time, which is concerning. The critic also seemed to have stayed stable. But again for this model as well, the low count of training data could be a factor for the model performing badly. But such a mode collapse that we saw can be due to many factors, including our use of excess geometric augmentation. We had to do it because the training size was small, but the overdoing of that might have also made things worse for the model.

On the classification part, we saw that the fully augmented dataset showed some really good scores in classification when the model was trained on it, compared to training done on the original dataset only. Not just the accuracy went up overall, some of the important metrics for all the classes also went up. The classes with the lowest amount of images, when augmented both partially and fully, showed an increase in Recall, Precision and F1 Scores. Having high recall value was important in our context of medical imaging. The improvement was not only limited to the fully augmented one only, the partially augmented training also had some good improvements. In some class wise validation metrics, not only does it beat the original training but the full augmented training. For example, in terms of precision, for both the vascular lesion and actinic class the score for partial augmented training was better than the fully augmented one. Even for the specificity score, although the improvements and changes were very microscopic, the partial augmentation training was slightly ahead compared to the other. The F1 score, on the other end, which brings an overall balance of precision and recall through its score, showed that both the partially augmented and fully augmented dataset training beats the original training for all three classes. Here, the improvement between training with the partial and fully augmented training is also clearly distinguishable, where the full augmentation beats the partial one.

Grad-CAM performed in these classes also tells the same story. The heatmap for both the lowest two classes showed that they had the artifacts required. The classification model also, with full confidence, predicted the right class for the images belonging to these two classes. Although the heatmap result was not par when performed on the actinic class of images. But the prediction was correct for this class too, just like the other two. But all in all, the final thought on the training could be that we should have chosen a more advanced model to deal with the 450x450 image generation. The number of instances in the majority of the classes were also the factor that we should have looked at the earlier stage. And geometric augmentation is not a good solution for this kind of cases and it might lead to other problems such as mode collapse that we experienced. But still, it could possibly be combated with a more powerful and state of the art model used for image generation.

Chapter 7

Future Work

In our paper, we have implemented the Wasserstein Generative Adversarial Network (WGAN) on a skin cancer dataset of 600×450 pixel images to generate realistic synthetic images of 450×450 pixels. Our main objective was to extend the original dataset with the generated synthetic images by WGAN which will enhance the training and prediction abilities of deep learning models. During our work, computational cost and high quality image generation were kept among our top priorities.

As we progress with our current work, the generated images by the WGAN, which are then fed into our pretrained ResNet50 model for classification purpose and evaluated for explainability using Grad-CAM, are targeted to be evaluated by domain specific experts for annotations and validations. This evaluation by the experts is necessary to ensure the validation of the realism of the synthetic generated images by the GAN model. As a result, we can ensure the reliability of our model on the skin cancer dataset through the expert assessment which is needed to validate the explainability provided by the Grad-CAM on the generated image part from where correct detection by the ResNet50 leads to accurate prediction.

Apart from skin cancer detection, we are willing to apply our proposed model on other rare disease datasets in order to check the generalizability of our model. Data scarcity is a problem in training models on other rare disease datasets as well. Our model can solve this problem and generate more realistic images which can further be used to extend the datasets of the rare diseases as well; which will be effective for the deep learning model training and prediction. If our approach proves to be a success, then our work can be extended further in a variety of rare disease detection tasks which can make GAN-based augmentation more useful in the medical domain.

Furthermore, we are determined to develop a hybrid GAN architecture together with the implementation of transfer learning model and explainable artificial intelligence models. This can lead to the overall architecture more scalability, robustness and interpretability. Thus, making diagnosis using AI-driven models more clinically acceptable.

By applying all these advancements into the betterment of our work, we are confident enough to come up with an architecture that can lead to better data augmentation,

more data efficiency, better model interpretability and better rare disease detection in the medical fields.

Chapter 8

Conclusion

In conclusion, current rare disease detection techniques face many difficulties such as data scarcity which decelerates the speed of detection due to the low volume of data that is used in training of the Deep Learning models. Also, patient privacy is a serious concern, which also affects the availability of more datasets in medical research. Our work on this thesis has shown that these challenges can be solved in the real world by using Generative Adversarial models effectively. We have used an unsupervised DCGAN and a Wasserstein GAN model to solve this problem by generating more synthetic images in a Skin Cancer dataset which had data imbalance issues among the classes. We proposed effective techniques and tweaks that can help immensely in bringing stability in these generative models, while also keeping the size of the original images intact. We compared and evaluated the performance of these models on various grounds using classification metrics and explainable ai. Our concluding thought on this work is that Wasserstein GAN is a good improvement over DCGAN and both of them are a good model of choice for generating small sized images. But when the requirement is generating larger sized images and the quality of the image is critical, like in the medical field, it is important to go for more advanced and state of the art generative models.

Bibliography

- [1] M. L. Nickerson et al., “Mutations in a novel gene lead to kidney tumors, lung wall defects, and benign tumors of the hair follicle in patients with the birt-hogg-dube syndrome,” *Cancer cell*, vol. 2, no. 2, pp. 157–164, 2002.
- [2] A. B. de Gonzalez and S. Darby, “Risk of cancer from diagnostic x-rays: Estimates for the uk and 14 other countries,” *The lancet*, vol. 363, no. 9406, pp. 345–351, 2004.
- [3] E. Bertino, B. C. Ooi, Y. Yang, and R. H. Deng, “Privacy and ownership preserving of outsourced medical data,” in *21st International Conference on Data Engineering (ICDE’05)*, IEEE, 2005, pp. 521–532.
- [4] D. J. Brenner and E. J. Hall, “Computed tomography—an increasing source of radiation exposure,” *New England journal of medicine*, vol. 357, no. 22, pp. 2277–2284, 2007.
- [5] D.-C. Li, C.-W. Liu, and S. C. Hu, “A learning method for the class imbalance problem with medical data sets,” *Computers in biology and medicine*, vol. 40, no. 5, pp. 509–518, 2010.
- [6] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.
- [7] P. Domingos, “A few useful things to know about machine learning,” *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.
- [8] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al., “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. icml*, Atlanta, GA, vol. 30, 2013, p. 3.
- [9] I. Goodfellow et al., “Generative adversarial nets,” *Advances in neural information processing systems*, vol. 27, 2014.
- [10] S. Ioffe, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [11] D. Kinga, J. B. Adam, et al., “A method for stochastic optimization,” in *International conference on learning representations (ICLR)*, San Diego, California; vol. 5, 2015, p. 6.
- [12] A. Radford, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

- [14] H.-C. Shin et al., “Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning,” *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.
- [15] M. Arjovsky, S. Chintala, and L. Bottou, *Wasserstein gan*, 2017. arXiv: 1701.07875 [stat.ML]. [Online]. Available: <https://arxiv.org/abs/1701.07875>.
- [16] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.
- [17] M. Rezaei et al., “A conditional adversarial network for semantic segmentation of brain tumor,” in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: Third International Workshop, BrainLes 2017, Held in Conjunction with MICCAI 2017, Quebec City, QC, Canada, September 14, 2017, Revised Selected Papers 3*, Springer, 2018, pp. 241–252.
- [18] H. Salehinejad, S. Valaee, T. Dowdell, and J. Barfett, “Image augmentation using radial transform for training deep neural networks,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 3016–3020.
- [19] H. Salehinejad, S. Valaee, T. Dowdell, E. Colak, and J. Barfett, “Generalization of deep neural networks for chest pathology classification in x-rays using generative adversarial networks,” in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, pp. 990–994.
- [20] H.-C. Shin et al., “Medical image synthesis for data augmentation and anonymization using generative adversarial networks,” in *Simulation and Synthesis in Medical Imaging: Third International Workshop, SASHIMI 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Proceedings 3*, Springer, 2018, pp. 1–11.
- [21] P. Tschandl, C. Rosendahl, and H. Kittler, “The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions,” *Scientific data*, vol. 5, no. 1, pp. 1–9, 2018.
- [22] Q. Yang et al., “Low-dose ct image denoising using a generative adversarial network with wasserstein distance and perceptual loss,” *IEEE transactions on medical imaging*, vol. 37, no. 6, pp. 1348–1357, 2018.
- [23] H. Rashid, M. A. Tanveer, and H. A. Khan, “Skin lesion classification using gan based data augmentation,” in *2019 41st annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, IEEE, 2019, pp. 916–919.
- [24] M. M. R. Siddiquee et al., “Learning fixed points in generative adversarial networks: From image-to-image translation to disease detection and localization,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 191–200.
- [25] X. Yan, B. Cui, Y. Xu, P. Shi, and Z. Wang, “A method of information protection for collaborative deep learning under gan model attack,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 871–881, 2019.

- [26] Y. Yang et al., “Gan-based semi-supervised learning approach for clinical decision support in health-iot platform,” *Ieee Access*, vol. 7, pp. 8048–8057, 2019.
- [27] X. Yi, E. Walia, and P. Babyn, “Generative adversarial network in medical imaging: A review,” *Medical image analysis*, vol. 58, p. 101552, 2019.
- [28] R. I. Bendjillali, M. Beladgham, K. Merit, and A. Taleb-Ahmed, “Illumination-robust face recognition based on deep convolutional neural networks architectures,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 2, pp. 1015–1027, 2020.
- [29] K. T. Chui, M. D. Lytras, and P. Vasant, “Combined generative adversarial network and fuzzy c-means clustering for multi-class voice disorder detection with an imbalanced dataset,” *Applied Sciences*, vol. 10, no. 13, p. 4571, 2020.
- [30] J. Islam and Y. Zhang, “Gan-based synthetic brain pet image generation,” *Brain informatics*, vol. 7, pp. 1–12, 2020.
- [31] D. Jha, M. A. Riegler, D. Johansen, P. Halvorsen, and H. D. Johansen, *Doubleu-net: A deep convolutional neural network for medical image segmentation*, 2020. arXiv: 2006.04868 [eess.IV].
- [32] H. Kang, J.-S. Park, K. Cho, and D.-Y. Kang, “Visual and quantitative evaluation of amyloid brain pet image synthesis with generative adversarial network,” *Applied Sciences*, vol. 10, no. 7, p. 2628, 2020.
- [33] A. B. Levine et al., “Synthesis of diagnostic quality cancer pathology images by generative adversarial networks,” *The Journal of pathology*, vol. 252, no. 2, pp. 178–188, 2020.
- [34] B. Liu, C. Tan, S. Li, J. He, and H. Wang, “A data augmentation method based on generative adversarial networks for grape leaf disease identification,” *IEEE Access*, vol. 8, pp. 102188–102198, 2020.
- [35] Y. Liu et al., “Prediction of oct images of short-term response to anti-veg treatment for neovascular age-related macular degeneration using generative adversarial network,” *British Journal of Ophthalmology*, 2020.
- [36] Z. Qin, Z. Liu, P. Zhu, and Y. Xue, “A gan-based image synthesis method for skin lesion classification,” *Computer Methods and Programs in Biomedicine*, vol. 195, p. 105568, 2020.
- [37] Z.-J. Xu, R.-F. Wang, J. Wang, and D.-H. Yu, “Parkinson’s disease detection based on spectrogram-deep convolutional generative adversarial network sample augmentation,” *IEEE Access*, vol. 8, pp. 206888–206900, 2020.
- [38] C. Zheng et al., “Assessment of generative adversarial networks model for synthetic optical coherence tomography images of retinal disorders,” *Translational Vision Science & Technology*, vol. 9, no. 2, pp. 29–29, 2020.
- [39] M. K. Ahuja, S. Sahil, and H. Spieker, “Mistake-driven image classification with fastgan and spinalnet,” 2021.
- [40] E. Yildiz et al., “Generative adversarial network based automatic segmentation of corneal subbasal nerves on in vivo confocal microscopy images,” *Translational Vision Science & Technology*, vol. 10, no. 6, pp. 33–33, 2021.

- [41] T. K. Yoo, J. Y. Choi, and H. K. Kim, “Feasibility study to improve deep learning in oct diagnosis of rare retinal diseases with few-shot classification,” *Medical & Biological Engineering & Computing*, vol. 59, pp. 401–415, 2021.
- [42] H. H. N. Alrashedy, A. F. Almansour, D. M. Ibrahim, and M. A. A. Hammoudeh, “Braingan: Brain mri image generation and classification framework using gan architectures and cnn models,” *Sensors*, vol. 22, no. 11, p. 4297, 2022.
- [43] M. Baygin, T. Tuncer, and S. Dogan, “New pyramidal hybrid textural and deep features based automatic skin cancer classification model: Ensemble darknet and textural feature extractor,” *arXiv preprint arXiv:2203.15090*, 2022.
- [44] D. S. Charan, H. Nadipineni, S. Sahayam, and U. Jayaraman, *Method to classify skin lesions using dermoscopic images*, 2022. arXiv: 2008.09418 [eess.IV].
- [45] S. Agrawal, V. Rewaskar, R. Agrawal, S. Chaudhari, Y. Patil, and N. Agrawal, “International journal of intelligent systems and applications in engineering advancements in nsfw content detection: A comprehensive review of resnet-50 based approaches,” *vol*, vol. 11, pp. 41–45, 2023.