HERALD COLLEGE KATHMANDU

UNIVERSITY OF WOLVERHAMPTON

Project and Professionalism

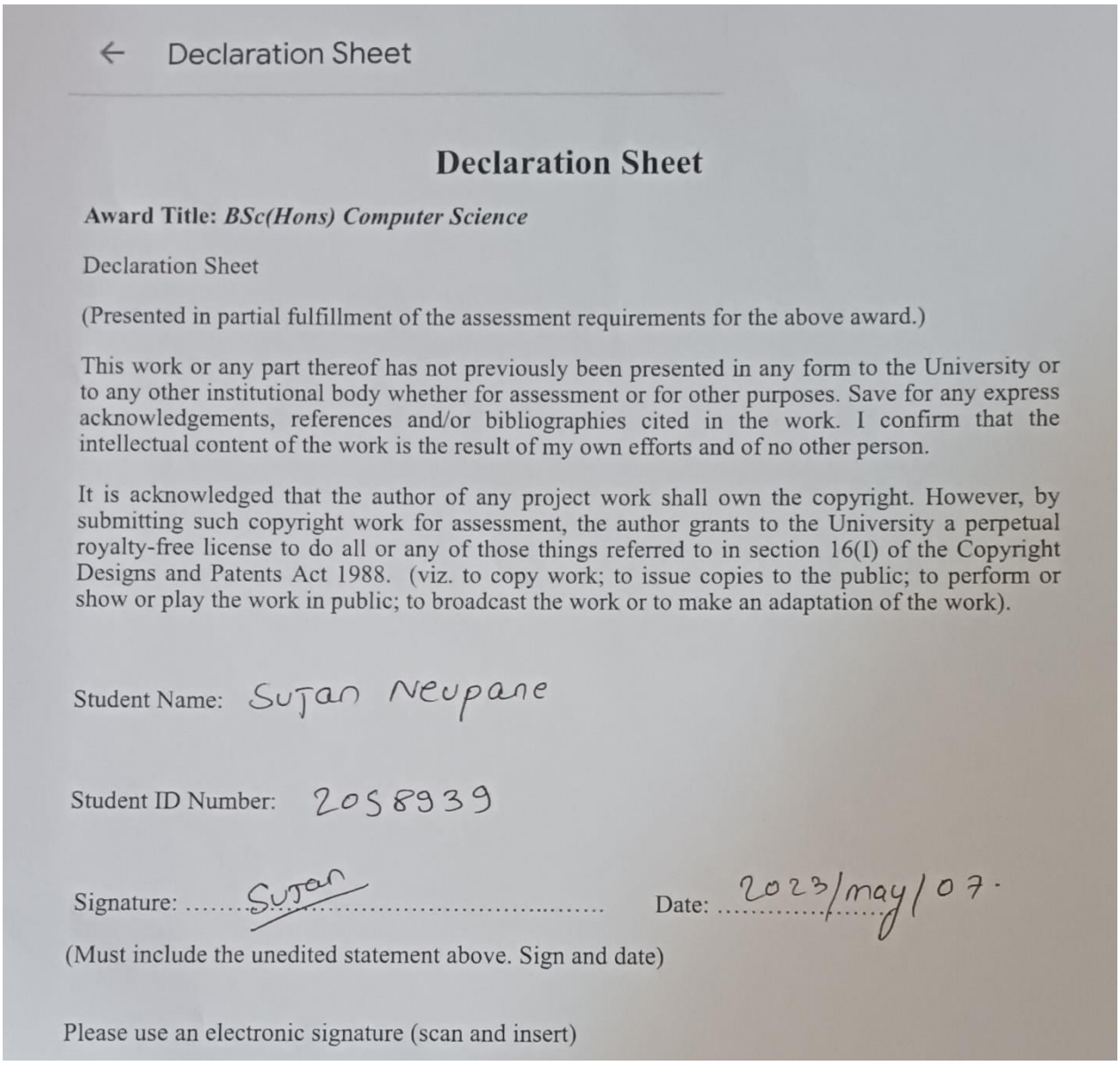
(6CS007)

Project Report

MedicareAI

Student Id	: 2058939
Student Name	: Sujan Neupane
Group	: Group 2
Supervisor	: Mr. Dinesh Saud
Reader	: Mr. Siman Giri
Module Leader	: Mr. Biraj Dulal
Submitted on	: May 15, 2023

Title and Declaration sheet



← Declaration Sheet

Declaration Sheet

Award Title: ***BSc(Hons) Computer Science***

Declaration Sheet

(Presented in partial fulfillment of the assessment requirements for the above award.)

This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. Save for any express acknowledgements, references and/or bibliographies cited in the work. I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

It is acknowledged that the author of any project work shall own the copyright. However, by submitting such copyright work for assessment, the author grants to the University a perpetual royalty-free license to do all or any of those things referred to in section 16(I) of the Copyright Designs and Patents Act 1988. (viz. to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make an adaptation of the work).

Student Name: Sujan Neupane

Student ID Number: 2058939

Signature: Sujan Date: 2023/may/07.

(Must include the unedited statement above. Sign and date)

Please use an electronic signature (scan and insert)

Abstract

With the advancement of machine learning, there has been an increase in the application of ML to enhance disease diagnosis in the healthcare industry, where ML-based solutions have proven to outperform clinician diagnosis in terms of precision and accuracy. By identifying patterns in medical images, commonly known as medical Imagining, ML solutions are extensively used to diagnose diseases from MRI and X-rays. In many medical datasets, there exists a class imbalance problem where the number of samples for the negative class far exceeds the number of samples for the positive class. Furthermore, Training Deep learning models require large amounts of data for optimal performance. Data with class imbalance will have a detrimental impact on classification model's performance and lead to model bias in favor of one class over another. Quality data collection is further constrained by concerns about patient privacy. As a result, this project conducts in-depth research on several techniques for augmenting image data by utilizing both conventional techniques like **Cropping, Flipping, Padding, Rotation,** and a variety of **Generative Adversarial Networks(GANs)** architectures. The Quality of synthetic data generated will be properly accessed. The performance of various classification algorithms will then be assessed using a variety of metrics after they have been trained on the augmented data. Therefore, this system will implement several GAN and CNN architectures, and select the best-performing model, which will be integrated into a web application using Django to enhance covid-19 diagnosis effectively and quickly.

Acknowledgement

I would personally like to thank my supervisor and reader Mr. Dinesh Saud and Mr. Siman Giri for their help and support by providing several ideas regarding the project during the final year timeframe. I could not have completed the work done for this report without your support. During the proposal defense, literature review and artifact design, my reader and supervisors provided me with ideas that could further improve my project. They also provided me the feedback about the things that I could incorporate into my report that could make my project and report stand out. I immensely thank my supervisor and reader for the support and help they have provided me because, without their help, it would have been very hard for me to effectively and efficiently complete my final report. Additionally, I would also like to thank my father and mother for all the support they have provided to me. Without them, I could not have achieved anything meaningful as well. Finally, I would also like to thank my friends as well who guided me when I did something wrong and put me in the right path for completing this project report.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

1.1 Project briefing

1.1. 1 Project Introduction

MedicareAI is a web-based application that allows users to diagnose covid-19 with a high level of accuracy and precision by selecting the best performing model from a collection of ML models that are enhanced by the capability of Generative Adversarial Networks (GANs). The system will make use of 3 different state of the art CNN and GAN architectures for detecting covid-19 from chest X-ray. The users will upload a chest x-ray which will be used to determine whether they are covid infected or not. The dataset **COVID-19 Radiography Database** from **Kaggle** is utilized in this project. The selected dataset has 3 classes: Covid, Normal, and Viral Pneumonia. Sixty-four randomly selected samples for each class are shown below.

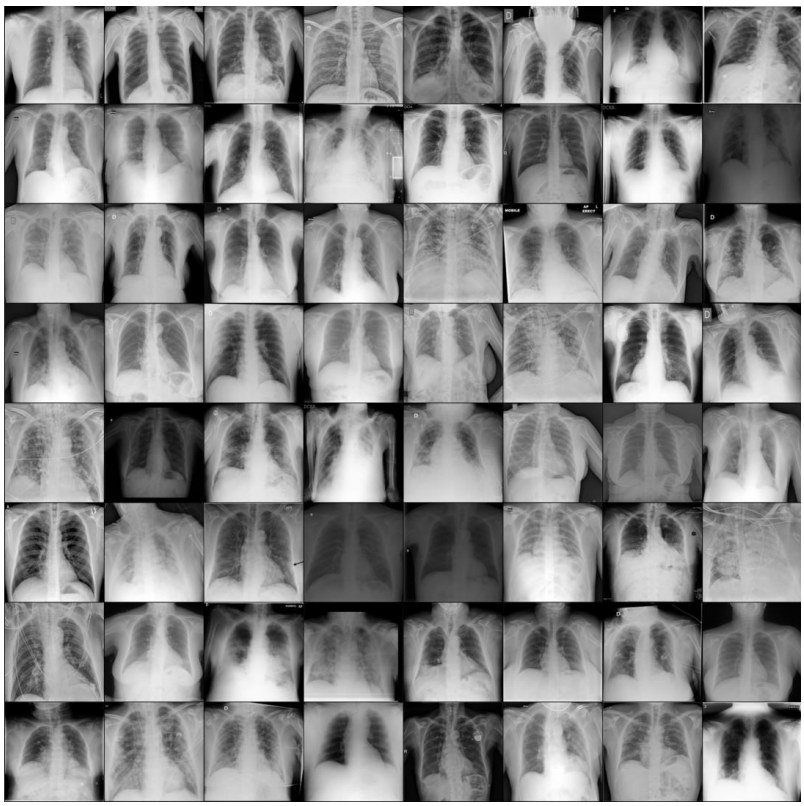


Figure 1 64 random covid positive x-ray samples

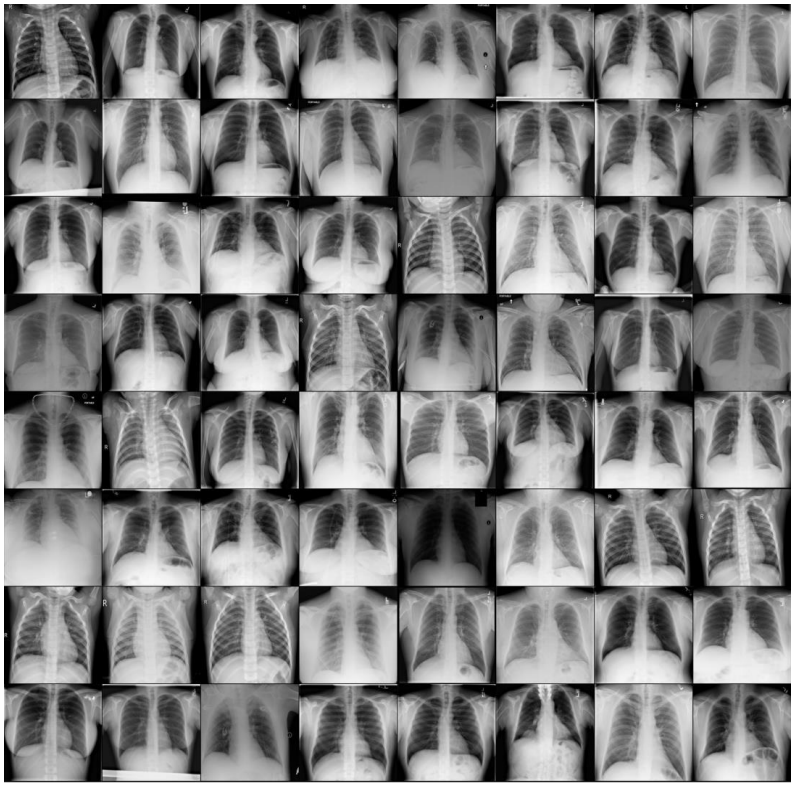


Figure 2 64 random normal x-ray samples

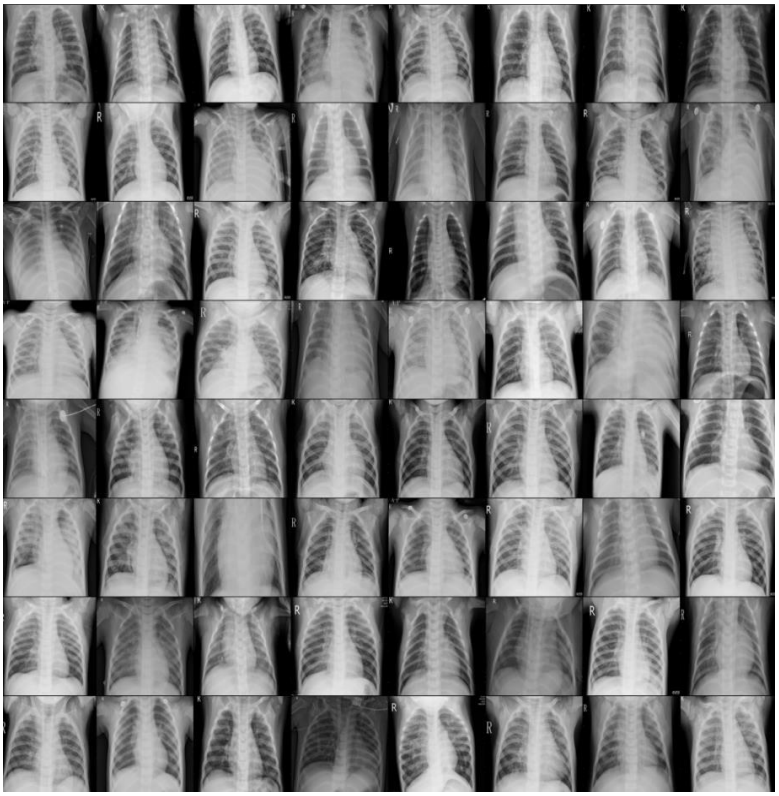


Figure 3 64 random viral pneumonia samples

1.1.2 Problem Statement

In most medical datasets, there is a challenge where the number of samples for the negative class is significantly higher than the number of samples for the positive class. It is called class imbalance. Data with class imbalance will have a detrimental impact on classification model's performance and lead to model bias in favor of one class over another. Collection of quality data is also restricted due to the concerns regarding individual's privacy, as healthcare data is sensitive and very private. Annotating large number of medical images like X-ray will also take huge amounts of time and resources, which can be better used. Traditional augmentation methods like horizontal and vertical flipping, rotations etc. don't produce diverse and varied images. In the selected dataset, three classes are Covid, Normal, and Viral Pneumonia and their data count is **3616**, **10192**, and **1345** respectively. A pie chart below showcases the class imbalance problem present in the selected dataset.

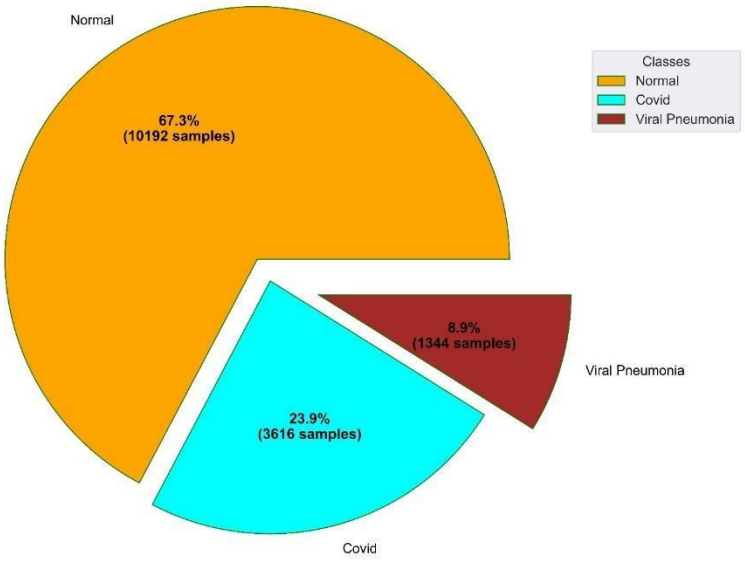


Figure 4 A pie chart showcasing the class imbalance problem

1.1.3 The Project as a Solution

Traditional image augmentation methods like horizontal and vertical flipping, and rotations don't produce diverse images. This problem can be solved by GANs as they can learn the distribution of original data and create new samples that mimic the original distribution. The synthetic data can also be used for research, which limits the ethical issue of sharing confidential patient information. By balancing the number of observations across minority and majority classes, GAN-generated synthetic data may be utilized to train ML models, thereby resolving the issue of class imbalance. In the selected dataset, the ratio of samples among the classes is **0.238**, **0.672**, and **0**.**0887**. Using three GAN architectures, synthetic x-ray images will be generated across imbalanced classes such that the number of samples count across all classes will be equal while training deep learning classification models. The bar diagram below showcases the number of samples for each class after the problem of class imbalance has been solved through GAN based data augmentation.

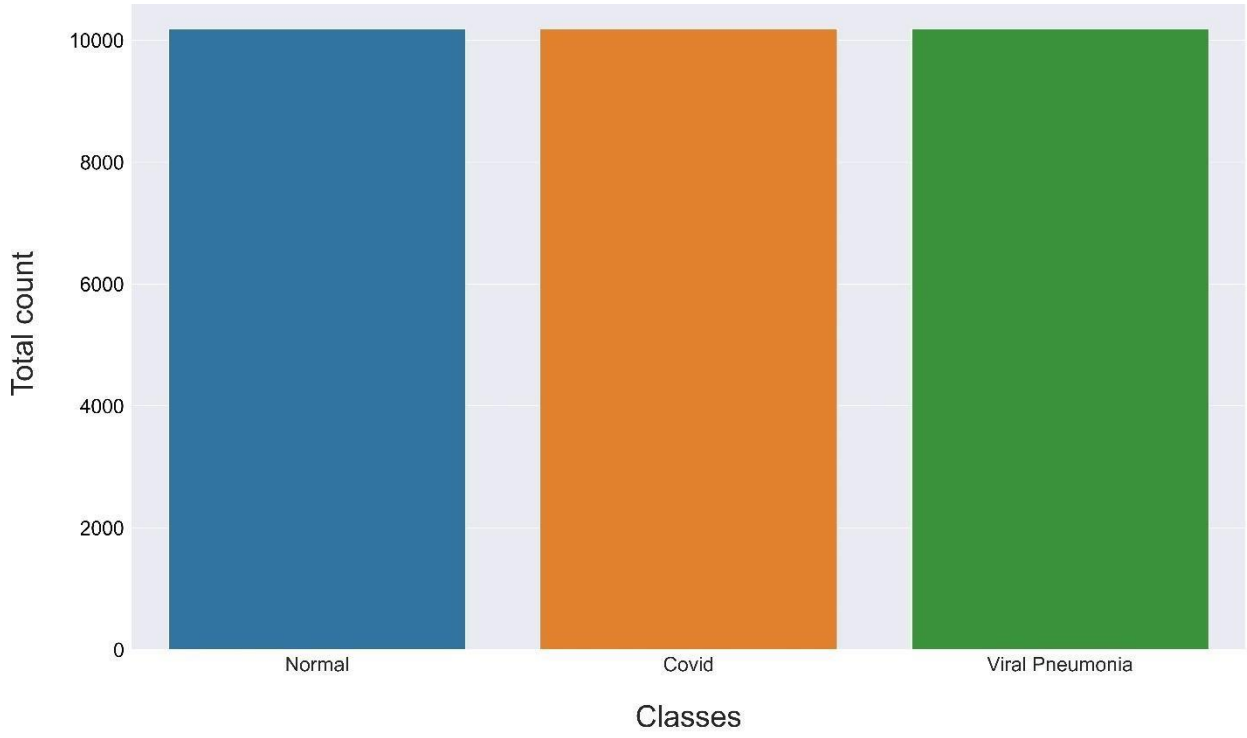


Figure 5 Class imbalance solved using GANs

1.1.4 Aspects of AI

This project implements multiclass classification, which is a component of computer vision. Three CNN architectures - ResNet50, GoogLeNet, and EfficientNet - are implemented to perform supervised classification. To solve the problem of class imbalance in the original dataset, three GAN architectures - DCGAN, WGAN, and WGANGP - are also implemented. GANs are a class of unsupervised generative algorithms that implement two rival neural networks: the Generator and the Discriminator. The generator neural network takes in random noise and maps it to an output, which undergoes binary classification as either fake or real by the discriminator. The neural networks are locked in a zero-sum game until the generator can generate samples that are indistinguishable from real samples.

1.1.4.1 Reasons for using a CNN

Unlike a traditional artificial neural network like a multilayer perceptron, CNN performs better for image data primarily due to following reasons:

- CNN can make use of stride and different pooling techniques to reduce the dimensions of image tensor. For example, if an input image tensor is 3*224*224 in dimensions, CNN can extract features and reduce this tensor possibly to a dimension of 1 * 10 * 10. Then, this tensor can be flattened to a 1d array which can be passed onto a linear layer. However, if we had used MLP, the number of input neurons in input layer of MLP would equal to 3*224*224 which would be more than 150000.
- CNN can tolerate small shifts in an image unlike MLP.
- CNN is also able to take advantage of extracting correlated features from a complex image, which is highly unlikely in an MLP.

1.1.4.2 EfficientNet

EfficientNet is a convolutional neural network proposed by Mingxing Tan and Quoc V. Le from google research. These researchers investigated how a CNN's performance might be enhanced by carefully balancing its resolution, width, and depth. The smallest EfficientNet, which has 5,330,564 parameters compared to the ResNet-50's 23,534,592, nevertheless, outperforms the ResNet-50 despite having a far smaller number of parameters. In EfficientNet, a compound coefficient is used to scale all depth, breadth, and resolution dimensions consistently. This compound scaling method is effective because a larger input image requires more layers to expand the network's receptive field and capture patterns in the larger image (Papers with code, 2023).

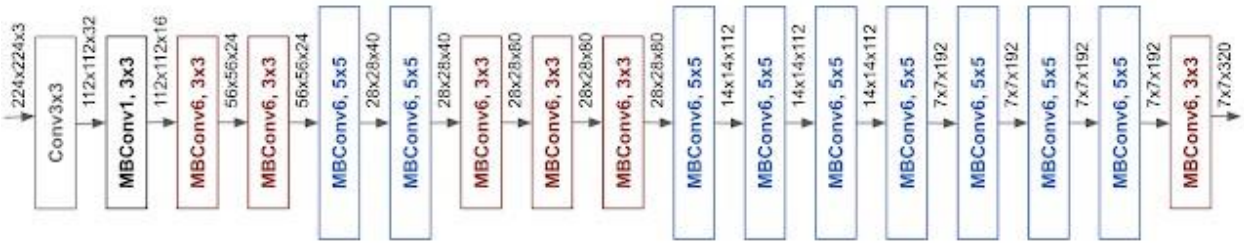


Figure 6 EfficientNet architecture (Google)

1.1.4.4 ResNet50

ResNet architecture was introduced in 2015 and stands for Residual Network. In a traditional convolutional neural network, the architecture begins with an input layer for an image which undergoes convolutions followed by a max pooling layer. Then, it might follow a fully connected layer with a Softmax or sigmoid activation function at the end depending upon whether the task is binary or multiclass classification. Increasing the layers in this fashion does not necessarily increase a model’s performance, rather the performance also depends upon the structure of the arrangement of these layers making efficient models. As we keep increasing the depth of the network, the loss keeps decreasing up to a certain point and then, keeps on increasing. This is due to exploding and vanishing gradient problems. Here, Residual Networks solve this problem by using a skip or shortcut connection between layers which enables us to take the output of one

layer and add it to another layer (Ma, 2019). The image below showcases the architecture of ResNet-50.

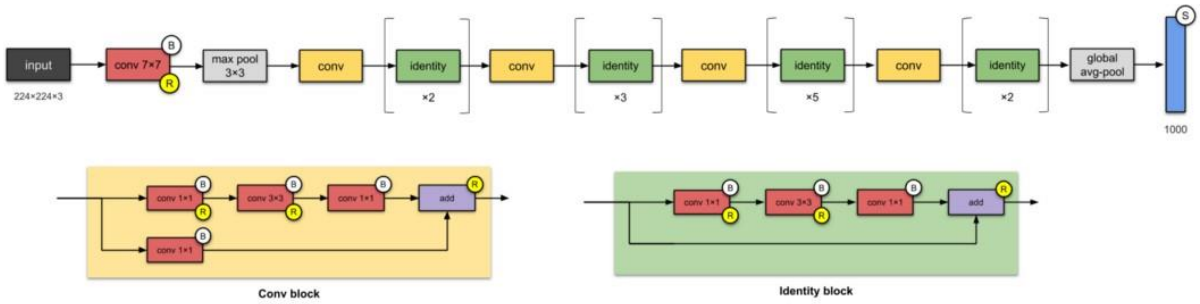


Figure 7 ResNet-50 architecture (Raimi Karim)

1.1.4.5 GoogLeNet

GoogLeNet is a convolutional neural network that is 22 layers deep. It was proposed at Google in 2014 with collaboration with various universities in the research paper with title as “Going deeper with convolutions”. This CNN architecture makes use of several different kinds of methods like global pooling and 1 * 1 convolution operation. This results in GoogLeNet having deeper network architecture. This architecture has 4 million parameters. There are 27 pooling layers which are used to reduce the dimensions of feature maps. Some other features of this architecture include global average pooling, inception module auxiliary classifier for training and 1*1 convolution operations (pawangfg, 2018). The pictorial representation of this CNN architecture has been given below.

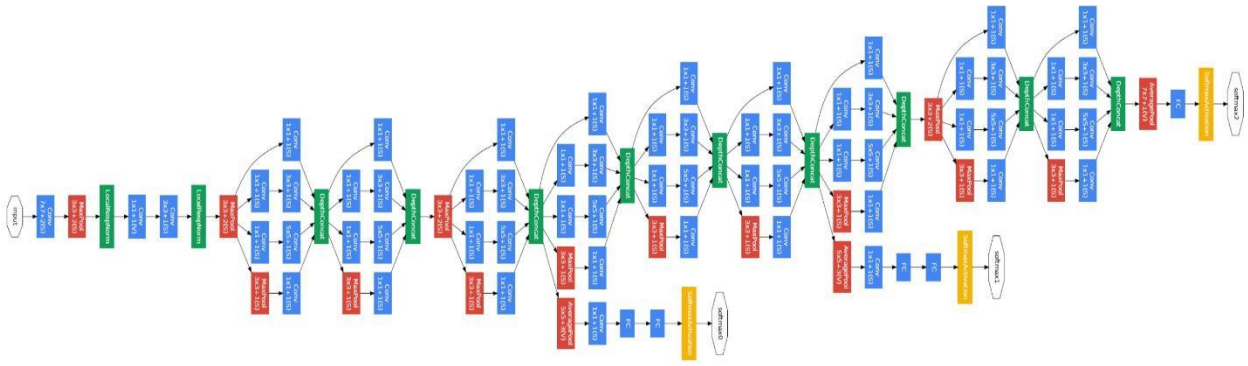


Figure 8 GoogLeNet architecture (Geeks for Geeks)

1.1.4.6 DCGAN, WGAN and WGAN-GP

DCGAN stands for deep convolutional generative adversarial network. DCGAN uses convolution and transposed convolutions-based layers in the discriminator and generator, as opposed to the multilayer perceptron used by the vanilla GAN. In the discriminator's architecture, Leaky ReLu activation is used to ensure that gradients actively pass through the architecture during generator training. The following images below highlight the architecture of the generator and discriminator in DCGAN.

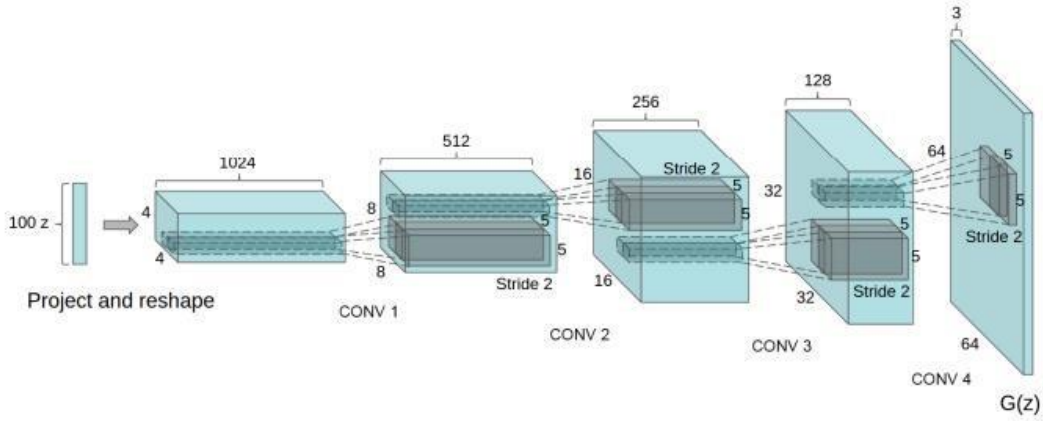


Figure 9 DCGAN Generator architecture (Original Paper)

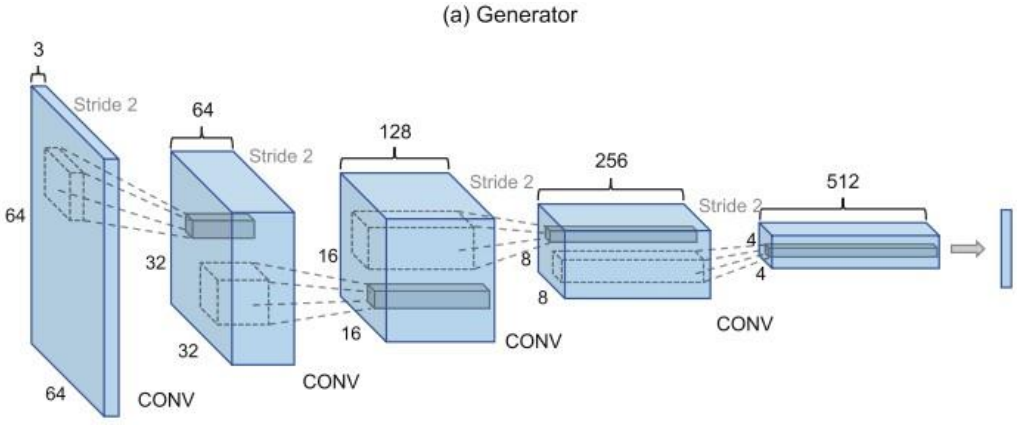


Figure 10 Generator architecture (Original Paper)

The DCGAN architecture suffers from problems like vanishing gradients and mode collapse. To solve this, WGAN architecture replaces the binary cross entropy loss function in the DCGAN with W-Loss. Additionally, the weights of the discriminator model are also clipped. In WGAN-GP, instead of weight clipping, a gradient penalty is implemented.

1.2 Aims

The aims of this project are given below:

1. To automate and fasten covid-19 diagnosis.
2. To solve the problem of class imbalance and data scarcity in covid-19 dataset using GANs.

1.3 Objectives

The objectives of this project are given below:

1. Synthetic image data for covid-19 chest x-ray dataset will be generated to solve problems like patient privacy concerns, class imbalance, and data scarcity by implementing the following GAN architectures:
    - Conditional DCGAN
    - Conditional WGAN
    - Conditional WGAN-GP
2. Quality of synthetic image data generated will be properly accessed using **Fréchet inception distance (FID).**

3. Covid-19 diagnosis will be boosted and automated by implementing various classification algorithms like ResNet 50, EfficientNet and GoogLeNet.

4. A web application using Django, HTML, CSS, and JavaScript will be developed for making real time predictions.

1.4 Artefact

The functional decomposition diagram (FDD) of this project is given below.

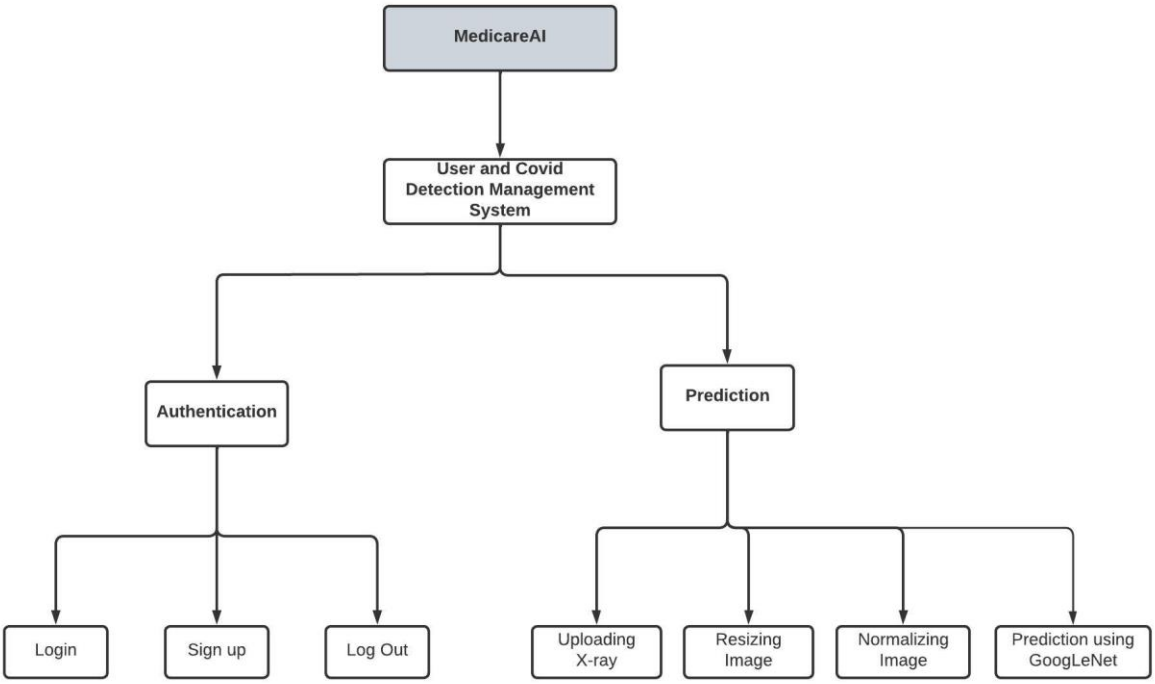


Figure 11 Functional decomposition diagram for MedicareAI

MedicareAI is a web application developed to enhance COVID-19 diagnosis using generative adversarial networks. The web application will be built using HTML, CSS, JavaScript, and Django. An index page will contain a form allowing the user to select a chest X-ray from their file system and upload it. The uploaded X-ray will be processed by the selected CNN architecture, which will return predicted probabilities for each class. The output displayed to the user will be the class with the highest probability, along with the probabilities for the other classes to show the confidence of the selected model for those classes.

To access the COVID-19 diagnosis feature, users must first create an account or log in with an existing MedicareAI account. If the user does not have an account, they will be

directed to a page to create one. Only after successful login will the user be able to upload an X-ray for analysis.

Since the primary goal of this project is to automate and accelerate COVID-19 diagnosis, only **one** subsystem - **the User and COVID Management System** - will be developed. This subsystem will enable users to log in and make predictions by uploading chest X-rays.

1.5 Academic Questions

1.5.1 What is the main purpose of this project?

The primary goal of the project MedicareAI is to help doctors swiftly evaluate whether a patient has a covid infection or not using the predicted probability. People in Nepal were getting infected left and right, and the number of infections was much higher than what the testing kits could handle. Therefore, the project MedicareAI has been created as an effort to speed up accurate covid testing.

1.5.2 Does this project implement ML algorithms?

Three CNN architectures, ResNet50, GoogLeNet, and EfficientNet, are trained on the imbalanced data, data balanced using traditional augmentation and. Additionally, the selected CNN models are also trained on 3 Generative Adversarial Network architectures: DCGAN, WGAN and WGAN-GP.

1.5.3 How will GANs help solve class imbalance?

Three GAN architectures, after training, will be passed an n-sized random noise vector for the imbalanced class. Here, n refers to the number of images for a particular class we want to generate. The GAN model will return the n number image tensors that are concatenated to the original dataset, which solves class imbalance.

1.5.4 How will the model be used to classify covid chest X-ray?

Since 3 CNN architectures are trained with a variety of augmented data, the best-performing model will be selected and used along with Django to create a web application that has the model integrated. Then, the user will upload an image from the front end, which is passed onto the backend using HTML form the image is passed onto the model class and predictions are made.

1.7 Scope and Limitation of the project

The scopes of this project are given below as bullet points.

- This project will implement three CNN architectures: ResNet-50, EfficientNet, and GoogLeNet.
- This project will implement three GAN architectures: DCGAN, WGAN, and WGANGP.
- This project will also incorporate several traditional image augmentation techniques to solve the class imbalance problem.
- This project will fully deploy trained models into a web application.

The limitations of this project are given as follows.

- GANs are only conditional but not controllable.
- Weak hardware limiting efficient training of models.
- Time constraints for implementing complex GAN architecture.
- Limited CNN architectures studied.

1.8 Report Structure

The structure of the report is briefly explained in the table below.

Section	Brief Explanation
Introduction	This section will provide a brief overview of the system proposed, along with its aims, objectives, scope, limitations, academic questions, problem statement, and solutions.
Literature Review	This section will include a detailed study of relevant kinds of literature and systems that also implement similar architecture like the one proposed in this project.
Project Methodology	This section explains the reasons for choosing the scrum methodology as compared to other methodologies.
Tools and Technologies	This section explains the reasons for which select tools and technologies were used for completing this project.
Artifact Design	This section includes the system's SRS, UML diagrams, model training, results, testing, and wireframes.
Conclusion	This section summarizes the major results and outcomes of this project.
Critical Evaluation	This section explains the major findings of the project as well as self-reflection on what things were learned and what can be improved.

Evidence of Project Management	This section contains the Gantt Chart and log sheets of each meeting organized by the project supervisor
References	This section contains references to the sources that are cited in this report.

Table 1 Report Structure

2. Literature Review

2.1 Research Papers

2.1.1 Evaluation of Deep Convolutional Generative Adversarial Networks for Data Augmentation of Chest X-ray Images

The authors of this literature acknowledge that training deep learning models with largely imbalanced data will result in model overfitting on the majority class sample's data. DCGAN-based data augmentation for minority class is proposed with binary cross entropy (BCE) loss as GAN's loss function. The authors also trained the classifier with traditional augmentation methods like horizontal and vertical flips, random crop, and rotations. CNN classifier is used for comparing the performance of the GAN and traditional augmentation. The classifier achieved the highest accuracy of 94.71 % on data balanced using Random crop with a precision 96.39 % and recall of 96 %. Similarly, the maximum AUC score achieved with traditional augmentation was 93.4 %. When the classifier was trained with synthetic data generated using DCGAN, it achieved the highest accuracy of 95.5 %, a precision of 96.2 %, and a recall of 97.7 %. Similarly, DCGAN based data augmentation also helped the model achieve the highest AUC score of 93.6 % respectively. The authors conclude that synthetic images generated using DCGAN are more diverse and varied and contain more information as compared to traditionally augmented images, which resulted in the classifier trained with DCGAN synthesized data outperforming the classifier trained with traditionally augmented data (Venu & Ravula, 2021).

My project will implement three CNN architectures: ResNet50, EfficientNet and GoogLeNet. A general CNN architecture followed by a fully connected layer with a SoftMax activation is used in this research work. The literature also compares performance of CNN classifier with 5 traditional augmentation methods: Randomflip-leftright, RandomCrop, ClipByValue, AdjustBrightness, and AdjustContrast. The results show that these traditional augmentation methods can improve a model's performance, which is why my project will also incorporate Random Horizontal Flip, Random Vertical

Flip, and Random Rotation. The performance of three classifiers trained on augmented data using these methods will also be compared using metrics used in the literature above like AUC score, accuracy, precision, recall, f1 score, false and true positive rate respectively. My project will also implement a conditional DCGAN because the results in this literature show that traditional augmentation methods don't produce diverse and varied images which can result in model overfitting on the training data. The literature's generator model generates images with 1 * 128 * 128 dimensions whereas my project's generator will generate images with 1 * 64 * 64 dimensions due to hardware limitations. The authors conclude that DCGAN is prone to mode collapse and vanishing gradient problem and wish to include the use of Wasserstein loss and Wasserstein loss with gradient penalty in future. Therefore, my project will also implement conditional WGAN and WGAN-GP to enhance the quality of generated images and make training GANs more stable. Following images below show the architecture of generator and discriminator, and performance of CNN classifier used in this literature.

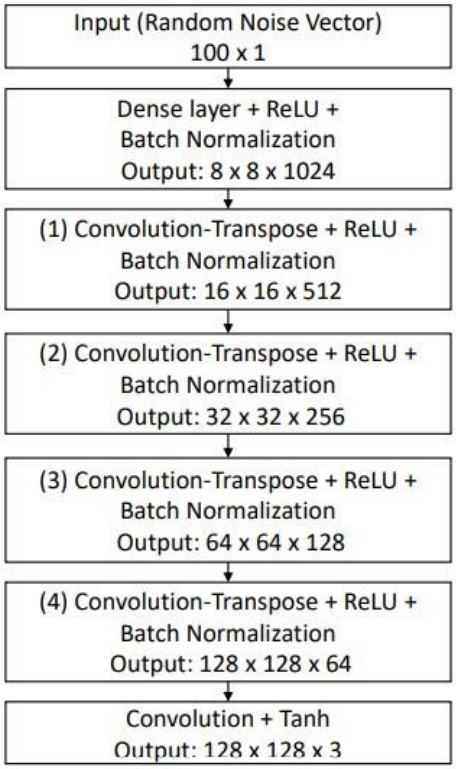


Figure 12 Generator Architecture (Original Paper)

Model/ Method	Accuracy	Precission	Recall	F1 Score	AUC	FPR	TPR
Randomflip-leftright	94.9	96.39	96.6	96.5	93.4	0.09	0.97
RandomCrop	94.71	95.31	97.54	96.42	92.31	0.13	0.98
ClipByValue	94.11	95.17	96.84	96	91.8	0.13	0.97
AdjustBrightness	92.91	94.37	96.02	95.18	90.28	0.16	0.96
AdjustContrast	92.58	97.41	92.29	94.77	92.83	0.07	0.92
GAN	95.5	96.2	97.7	97	93.6	0.1	0.98

Figure 13 CNN classifier's performance (Original Paper)

2.1.2 Comparison of Deep Learning Models AlexNet and GoogLeNet in Detection of Pneumonia and Covid19

The literature acknowledges the challenge doctors face in diagnosing COVID-19 due to the exponential growth in positive cases, as well as the role deep learning models have in easing this difficulty by being able to identify COVID positive cases with high precision and accuracy. A dataset with a total of 6357 chest x-rays was utilized. Two CNN architectures, AlexNet and GoogLeNet, were implemented for performing multiclass classification on three classes: Covid, Normal, and Pneumonia. The dataset was randomly split into a 70 % training set and a 30 % testing set. Each model was trained up to 30 epochs. The dataset used was imbalanced as the majority of the 6357 samples belonged to the Normal class. The problem of class imbalance was not addressed in this work, and the models were trained on the imbalanced dataset. The **Normal**, **Pneumonia**, and **Covid** classes had a total number of 1508, 4273, and 576 samples respectively. The AlexNet model achieved an accuracy of 95.23 % and a precision of 0.9046 whereas the GoogLeNet model achieved an accuracy of 95.64 % with a precision of 0.921 approximately. The GoogLeNet model outperformed the AlexNet model by a slight margin. The authors conclude that the primary limitation of their study was data scarcity and class imbalance (Yaren & Deniz, 2021).

This literature trains two CNN architectures, AlexNet and GoogLeNet, on the imbalanced data without solving the problem of class imbalance. My project will implement three traditional augmentation methods and three conditional generative adversarial network

architectures for addressing this problem. Additionally, my project will also implement the GoogLeNet model along with other two models: EfficientNet and ResNet50. This literature also compares the model's performance using 4 metrics, whereas my project will compare each of the three model's performances using accuracy, precision, recall, f1 score, true positive rate, true negative rate, false positive rate, AUC score, etc.

2.1.3 CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection

This literature presents a new type of Auxiliary Classifier Generative Adversarial Network (ACGAN) called CovidGAN for synthesizing covid positive x-rays. The training set can be quickly expanded using conventional augmentation techniques like resizing, rotating, and flipping images. The change to new images, however, is fairly limited in this sort of augmentation because the original sample is only minimally altered to produce a new sample, which yields fewer diversified and varied images. To prevent this, the literature aims to implement a new type of GAN, CovidGAN with binary cross-entropy, for performing data augmentation. The original dataset had 1124 chest x-ray samples out of which 403 samples belonged to covid positive class and 721 images belonged to the negative class. A VGG16 CNN classifier is used to compare the performance of original imbalanced data and GAN-based augmented data. CovidGAN with conditional generation is implemented. The generator model is passed class information by concatenating the noise vector with one hot encoded class vector, whereas the discriminator model is passed class information in form of channels. The height and width of generated images are 112 * 112. In the original imbalanced dataset, VGG16 achieved a maximum accuracy of 85 %, sensitivity of 69 %, and specificity of 95 %. After training VGG16 with CovidGAN based data augmentation, the model achieved an accuracy of 95 %, sensitivity of 90 %, and specificity of 97 %. In the future, the authors wish to implement progressive growing GAN to enhance the quality of generated images (Waheed, et al., 2020). The following screenshots showcase the working mechanism of CovidGAN and the performance of VGG16 model on original data and GAN based augmented data.

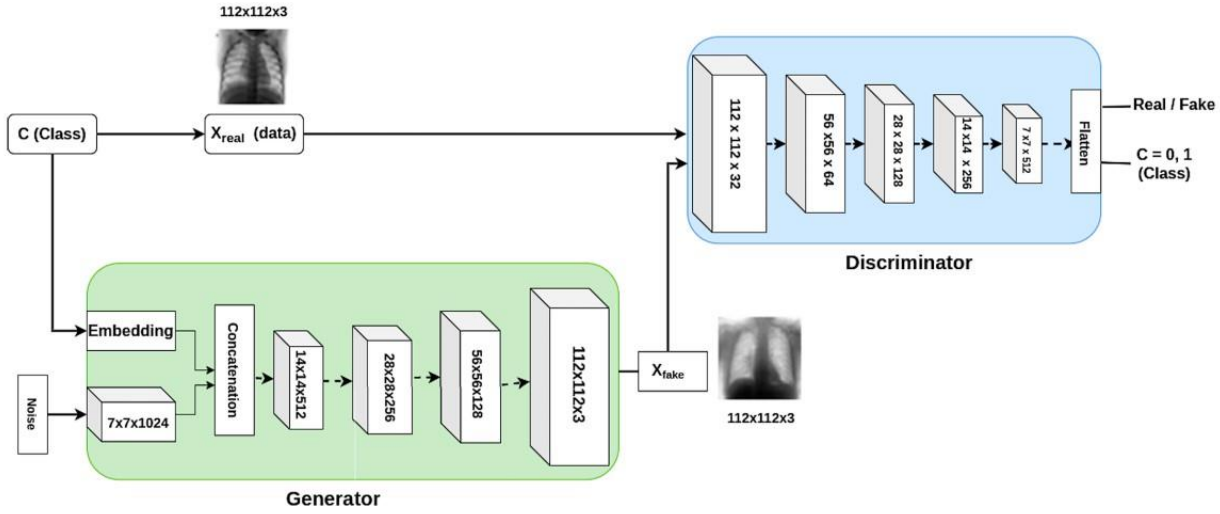


Figure 14 Working mechanism of CovidGAN (Original paper)

Dataset	Class	Precision	Recall	F1-score	Support	Accuracy (%)	Sensitivity(%)	Specificity(%)
Actual Data (CNN-AD)	Covid-CXR	0.89	0.69	0.78	72	**85**	**69**	**95**
	Normal-CXR	0.84	0.95	0.89	120			
	Macro-average	0.87	0.82	0.84	192			
	weighted-average	0.86	0.85	0.85	192			
Actual data + Synthetic Augmentation (CNN-SA)	Covid-CXR	0.96	0.90	0.93	72	**95**	**90**	**97**
	Normal-CXR	0.94	0.97	0.96	120			
	Macro-average	0.95	0.94	0.94	192			
	weighted-average	0.95	0.95	0.95	192			

AD stands for actual data, SA stands for synthetic augmentation, and Support means the total number of samples.

Figure 15 Performance of VGG15 model (Original paper)

This literature only makes use of the VGG16 model along with one conditional GAN architecture. Furthermore, the effectiveness of GAN-based data augmentation is not evaluated against that of conventional augmentation. My project will implement three CNN architectures and three GAN architectures. The performance of CNN models trained on both imbalanced data, data augmented using traditional augmentation, and data augmented with three GAN architectures will also be compared in my project.

2.1.4 Enhancing Automated COVID-19 Chest X-ray Diagnosis by Image-to-Image GAN Translation

In this study, a conditional generative adversarial network called Pix2Pix GAN was implemented for translating non-covid-19 chest x-rays to covid-19 positive chest x-rays. The original dataset used in this literature had extreme class imbalance as only 219 covid19 samples were present, but 1341 and 1345 samples were present for normal and viral pneumonia classes respectively. All images were resized to a dimension of 224 * 224. Random rotation and random horizontal flip techniques were also used to expand the variety of training samples. Image-to-image translation between covid-positive and non-covid-positive x-rays was implemented using a U-Net-based generator and discriminator architecture. The GAN model was trained for 100 epochs. After training, a total of 1100 new synthetic covid-19 x-rays were generated to balance the number of classes. A ResNet50 CNN architecture was used for performing multiclass classification using the SoftMax activation function. Transfer learning with the pre-trained weights of ResNet50 was also implemented for performing multiclass classification. Additionally, the ResNet50 model was also trained from scratch for comparing the performance. The model trained with GAN-based augmented data achieved an accuracy of 97.8 % when trained as compared to 96.1 % accuracy in the transfer learning mode. The model trained with pretrained weights achieved an accuracy of 96.1 % when trained on imbalanced data. However, the same model achieved an accuracy of 95.6 % when trained from scratch. The authors conclude that training the ResNebt50 model was more stable with the GAN based augmented data and its performance also was enhanced in terms of precision, recall, and f1 score (Liang, et al., 2020).

The screenshot below showcases the ResNet50 model's performance thoroughly.

Measure	Class		
	Normal	*COVID-19*	*Viral*
Original imbalance dataset (transfer learning)			
Accuracy	0.961		
Precision	0.935	1.000	0.950
Recall	0.967	0.967	0.950
F1 score	0.951	0.983	0.950
Original imbalance dataset (train from scratch)			
Accuracy	0.956		
Precision	0.967	1.000	0.906
Recall	0.967	0.933	0.967
F1 score	0.967	0.966	0.935
cGAN enhanced dataset (transfer learning)			
Accuracy	0.978		
Precision	0.952	1.000	0.983
Recall	0.983	1.000	0.950
F1 score	0.967	1.000	0.966
cGAN enhanced dataset (transfer from scratch)			
Accuracy	0.961		
Precision	0.935	0.984	0.965
Recall	0.967	1.000	0.917
F1 score	0.951	0.992	0.940

Figure 16 Performance of ResNet50 model (Original paper)

This literature implements the ResNet50 model for performing multiclass classification. Additionally, a Pix2Pix GAN is implemented for performing image to image translation.
The scope of my project is to synthesize images using DCGAN, WGAN and WGAN-GP. Therefore, image to image translation related techniques will not be integrated in my project. However, unlike this paper which only makes use of one CNN model, my project will implement three models: ResNet50, GoogLeNet, and EfficientNet.

2.1.5 Wasserstein GAN based Chest X-Ray Dataset Augmentation for Deep Learning Models: COVID-19 Detection Use-Case

The authors acknowledge that generating and annotating medical images like chest x-rays is a time and resource-consuming process. Additionally, the scarcity of such high quality data is a notable challenge with deep learning in medical imaging. Therefore, this literature proposes a Wasserstein Generative Adversarial Network (WGAN) to synthesize high-quality x-rays to enhance covid-19 classification performance of a model. The COVID-19 Radiography dataset from an open-access benchmark dataset containing a total of 3615 covid-19 chest x-rays is utilized. Traditional GANs like DCGAN that use binary cross entropy as the loss function in discriminator and generator models are prone to mode collapse and vanishing gradient problems. So, WGAN architecture is used to equalize the number of samples across all three classes: Normal, Covid, and Viral Pneumonia. The WGAN was trained for 20,000 epochs, with the generator model generating images of 96 * 96 dimensions. A lightweight CNN classifier was trained on imbalanced data and data with WGAN-based augmentation for 50 epochs. Each class had a maximum number of 10,192 samples after WGAN-based augmentation. The augmented data was divided into a 20 % testing set and an 80 % training set. Additionally, the imbalanced data also underwent regular non-GAN-based augmentation. The classifier trained with non-GAN-based augmentation achieved a maximum accuracy of 93.04 %, a precision of 92.81 %, a specificity of 99.82 %, and an AUC score of 98.39 %. However, the classifier trained with WGAN-based augmented data outperformed the previous classifier with an accuracy of **95.34** %, a precision of **94.2** %, a specificity of **99.91** %, and an AUC score of **99.4** % respectively. The literature concludes that WGAN can be a highly precise technique for augmenting sparse medical datasets by generating high-quality synthetic medical images (Hussain, et al., 2022).

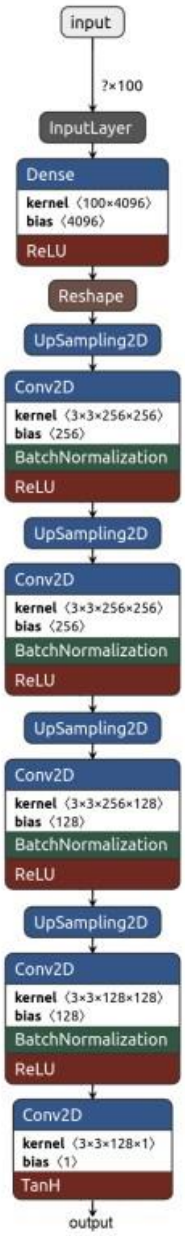


Figure 17 WGAN discriminator model architecture [(original paper)]

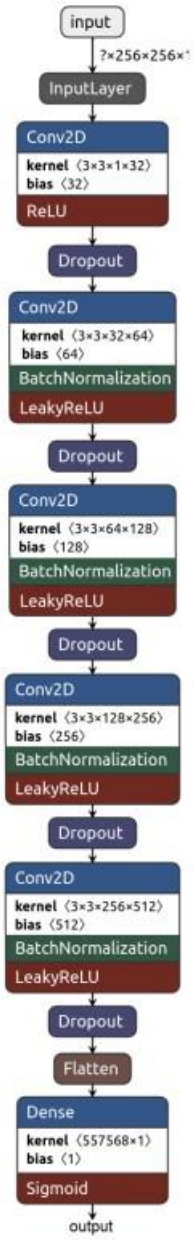


Figure 18 WGAN generator model architecture [(original paper)]

My project will use ResNet50, GooLeNet, and EfficientNet instead of the single lightweight CNN classifier utilized in this literature. The findings of this literature demonstrate that WGAN can produce chest x-rays of high quality. As a result, I'll also use WGAN and WGANGP-based data augmentation in my project. To understand how GAN-based augmentation outperforms traditional augmentation, the performance of classifiers in my project will be compared to GAN-based augmentation versus traditional augmentation like in the following literature.

2.1.6 Automated COVID-19 diagnosis using Deep Multiple Instance Learning with CycleGAN

The authors of this literature acknowledge that manual examination of CT scans for covid19 diagnosis is slow, and this process should be automated using deep learning. The literature proposes a new conditional cycle generative adversarial network(CycleGAN) that can synthesize CT scans of resolutions 512 * 512. The CycleGAN architecture used in this work is used to map the distribution of one CT scan to another by transferring the characteristic of the first image via image-to-image translation. The GAN architecture used consists of two generator models and two discriminator models. Initially, normal data is passed into the first generator model which generates abnormal data. This new synthetic data is passed into the second generator model which synthesizes the CT scans. The corresponding synthesized images are also passed into two discriminator models to improve the training of generator models. Various other GAN architectures like CGAN, DCGAN, Clustering + GAN, DetectorGAN, and CovidGAN are also implemented to compare performance with CycleGAN. CycleGAN outperformed all other GANs with maximum accuracy and specificity of **97** % and **96.5** %. Two classifiers, DeCovNet and DMIL with SoftMax activation, were implemented for performing multiclass classification on CT scans. The performance of the implemented classifiers was also compared with existing techniques like VGG-19, COVIDX-Net, DenseNet-121, and Shallow CNN. DMIL classifier outperformed all other existing techniques with a maximum accuracy of **97** %, specificity of **96.5** %, and f1 score of **96.3** %. Additionally, the DMIL classifier also achieved an accuracy of **97** % on the original

imbalanced dataset, and **98.96** accuracy with CycleGAN-based augmentation. The literature concludes that CycleGAN is a better approach for augmenting sparse medical datasets like covid CT scans as compared to other GAN architectures (Suganya & Kalpana, 2022).

The following image below showcases the CycleGAN architecture implemented in this literature.

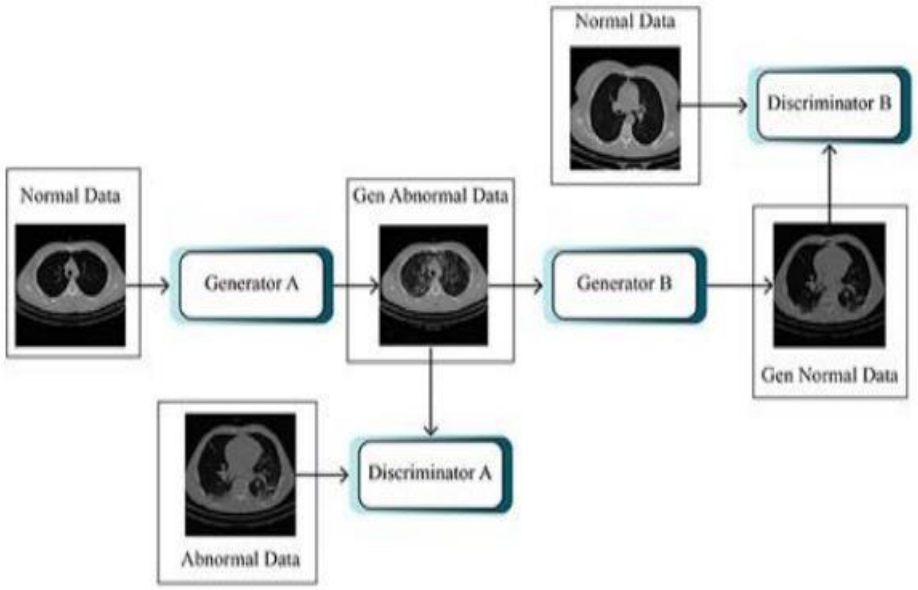


Figure 19 Working mechanism of CycleGAN (Original paper)

This literature implemented several GAN architectures along with two classifiers. The performance of classifiers is also compared with existing techniques. Like this literature, my project will implement three GAN architectures: cDCGAN, cWGAN, and cWGAN-GP. The performance of three classifiers, GoogLeNet, ResNet50, and EfficientNet, will be compared with imbalanced data, traditionally augmented data, and GAN-based augmented data. The scope of my project is not the image-to-image translation, which is why CycleGAN will not be implemented in my project. Like in this literature, the performance of the classifiers used in my project will be compared in terms of accuracy, specificity, f1 score, precision, false positive rate, true positive rate, AUC score, etc.

2.1.7 A Survey on Deep Learning Advances and Emerging Issues in Pneumonia and COVID19 Prediction

The literature acknowledges that deep learning algorithms have been a tremendous success in early identifying and automating covid-19 detection using medical images like X-rays and CT scans. The main goal of this paper is to address new advances in deep learning like Transformers, GANs, and LSTMs and cover issues like class imbalance in sparse covid-19 medical datasets. A total of ten medical datasets relating to covid-19 and viral pneumonia are described in this literature. This work describes three solutions that can be achieved via chest x-rays or CT scans: Image classification, object detection, and image segmentation. Different deep learning architectures based on CNN, LSTM, and Transformers can be applied to perform binary or multiclass classification on a covid-19 medical dataset. For localizing covid-19 pneumonia in x-rays, R-CNN and YOLO can be applied as these models can classify and locate covid-19 with high precision and accuracy. Finally, for segmenting covid-19, Mask R-CNN and UNet can be applied as they are highly precise and accurate while classifying each pixel in an x-ray or CT scan. The literature discusses two major problems in covid-19 datasets: data privacy and scarcity. The authors explain that x-rays and CT scans are sensitive data that are subjected to patient privacy, which limits their use ultimately resulting in a scarcity of high quality data and class imbalance while training deep learning models. Models trained without addressing class imbalance will result in model biasness towards the majority class due to the skewness in the distribution of the number of samples of classes. The literature concludes that GAN-based data augmentation can help enhance classification model's performance while diagnosing covid-19 from chest x-rays and CT scans and address the problem of class imbalance and patient privacy (Makhanov, et al., 2022).

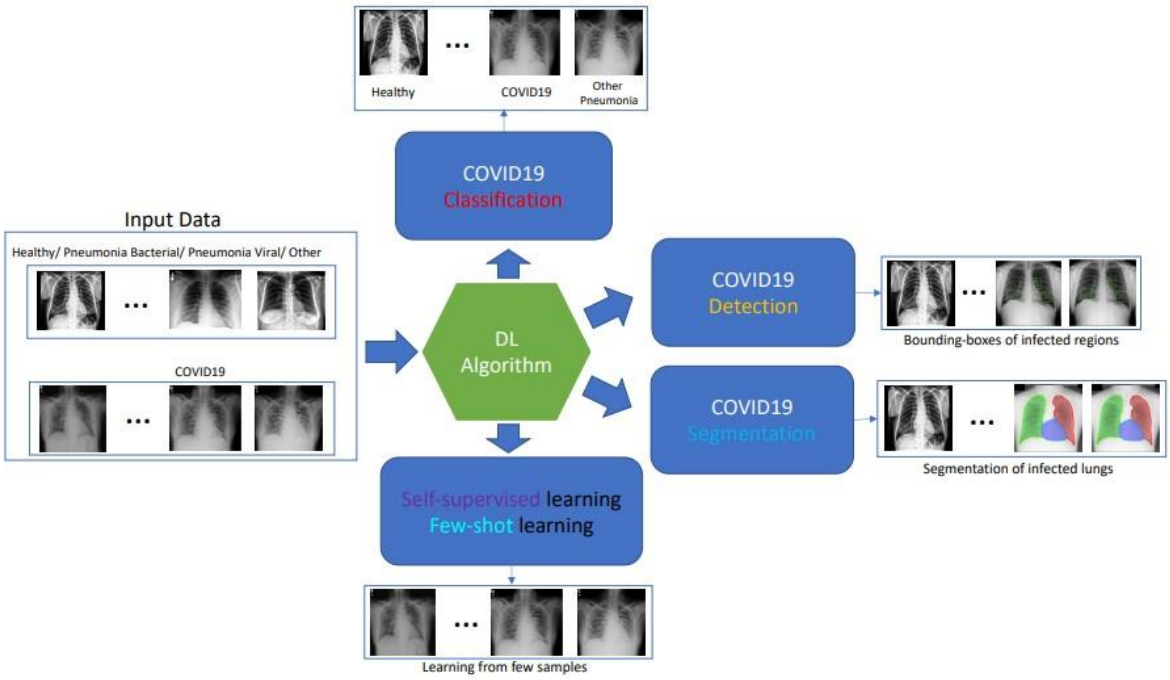


Figure 20 Applications of chest x-rays using deep learning (Original paper)

The results of this literature explain that GAN-based data augmentation can solve two major problems: class imbalance and patient privacy. Models trained with GAN-based augmented data can help solve the problem of class imbalance as the number of samples for each class will be equal. Additionally, using synthetic images to equalize the number of samples of minority classes will address and reduce the issues related to patient privacy. Therefore, my project will incorporate three GAN architectures whose performance will be compared with imbalanced data and traditionally augmented data. The performance of three CNN-based classifiers, GoogLeNet, EfficientNet, and ResNet50, will also be compared with respect to various metrics like accuracy, precision, recall, false positive rate, false negative rate, specificity, sensitivity, f1 score, and AUC score respectively.

2.2 Review of similar systems

The following similar systems to this project are explained briefly below.

2.2.1 Mirry

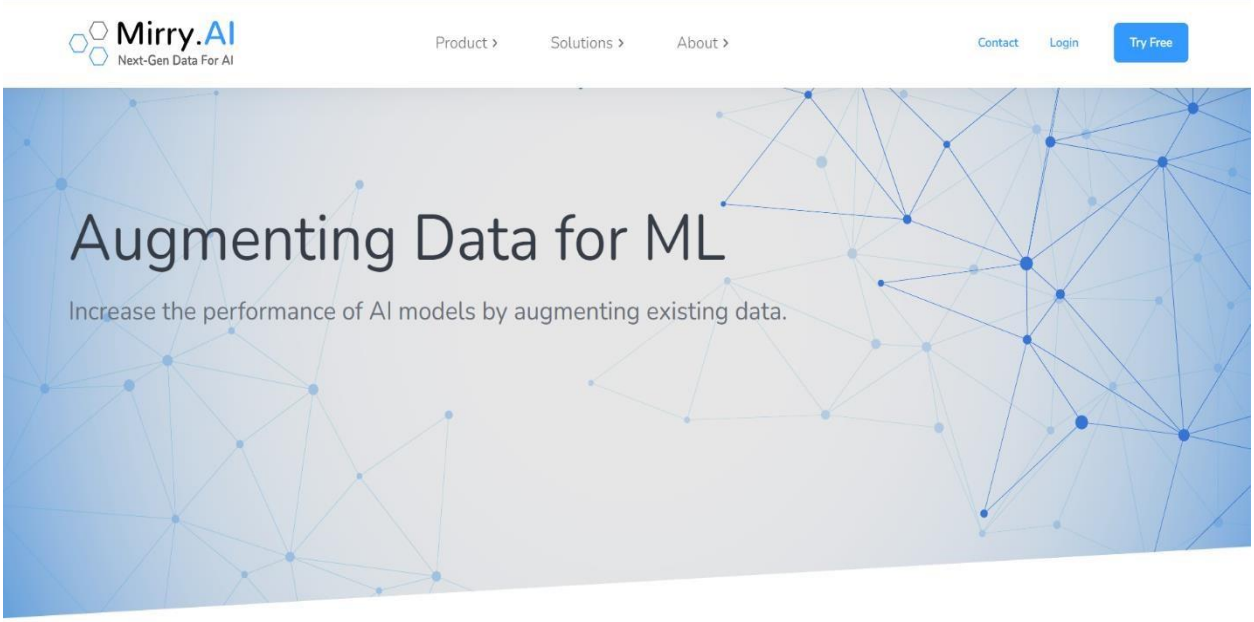


Figure 21 Mirri.AI homepage (Mirri.AI)

Mirri.AI is a platform that allows users to access synthetic data for training deep learning models. The platform makes use of conditional generative adversarial networks for generating synthetic data. Additionally, the platform enables the creation of synthetic data across a range of industries, including finance, telecommunications, healthcare, and retail. The screenshot below showcases the working mechanism of this platform for augmenting synthetic data (Mirri.AI, 2022).

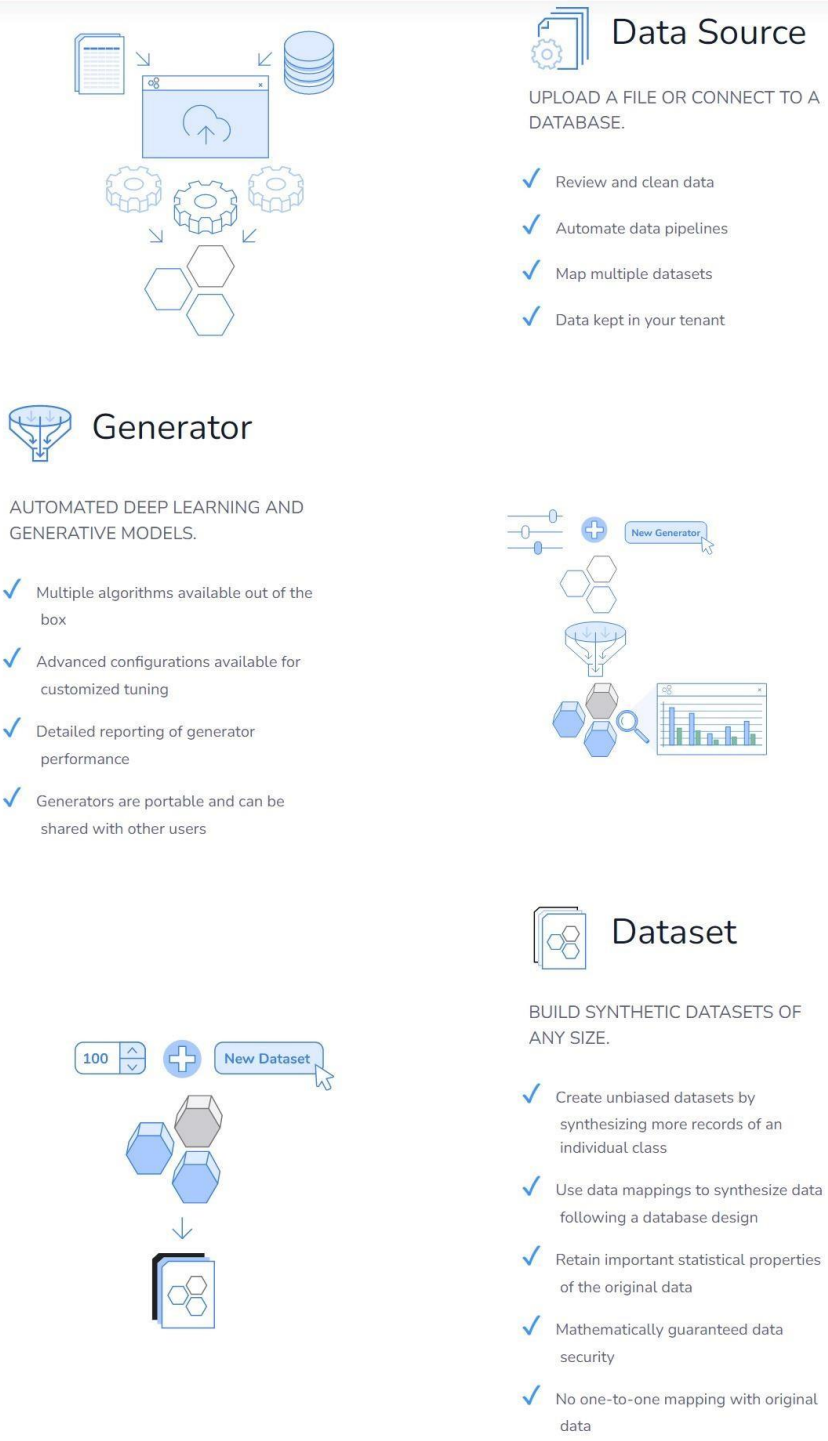


Figure 22 Mirri.AI system's working mechanism (Source)

2.2.2 Tonic

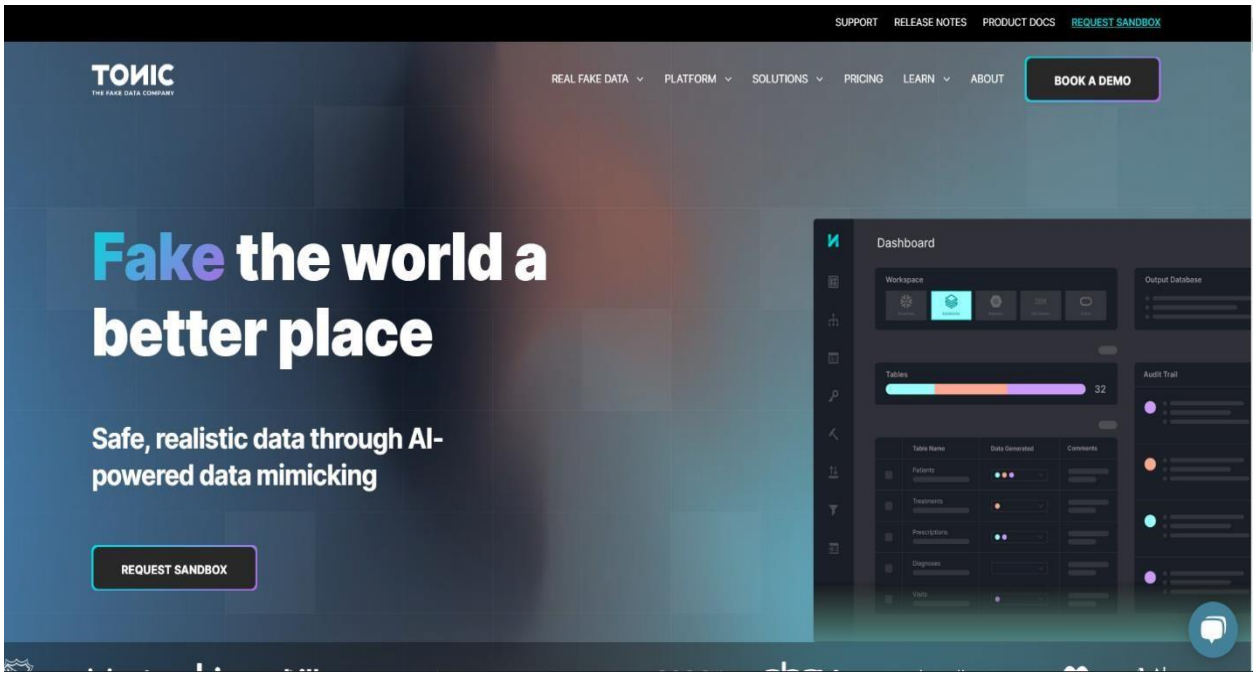


Figure 23 Tonic.AI homepage (Toni.AI)

Tonic's AI synthesizer uses a Variational Autoencoder (VAE) as its main processing engine. Businesses in a wide range of industries, including finance, healthcare, retail, tech, and insurance, can improve their services by using the synthetic data offered by tonic. Tonic believes in protecting patient privacy and abiding by policies put in place by the Centers for Disease Control and Prevention(CDC), particularly in the context of healthcare (Tonic, 2022).

2.2.3 TripleBlind

TripleBlind is a healthcare-specialized data synthesizing platform that supports data collaboration through analytics and training ML models. The synthetic healthcare data generated through TripleBlind is Health Insurance Portability and Accountability Act of 1996(HIPAA) approved (TripleBlind, 2022). The platform developed a method for encrypting data for increasing collaboration and sharing the data without being decrypted (Fierce Healthcare, 2021).

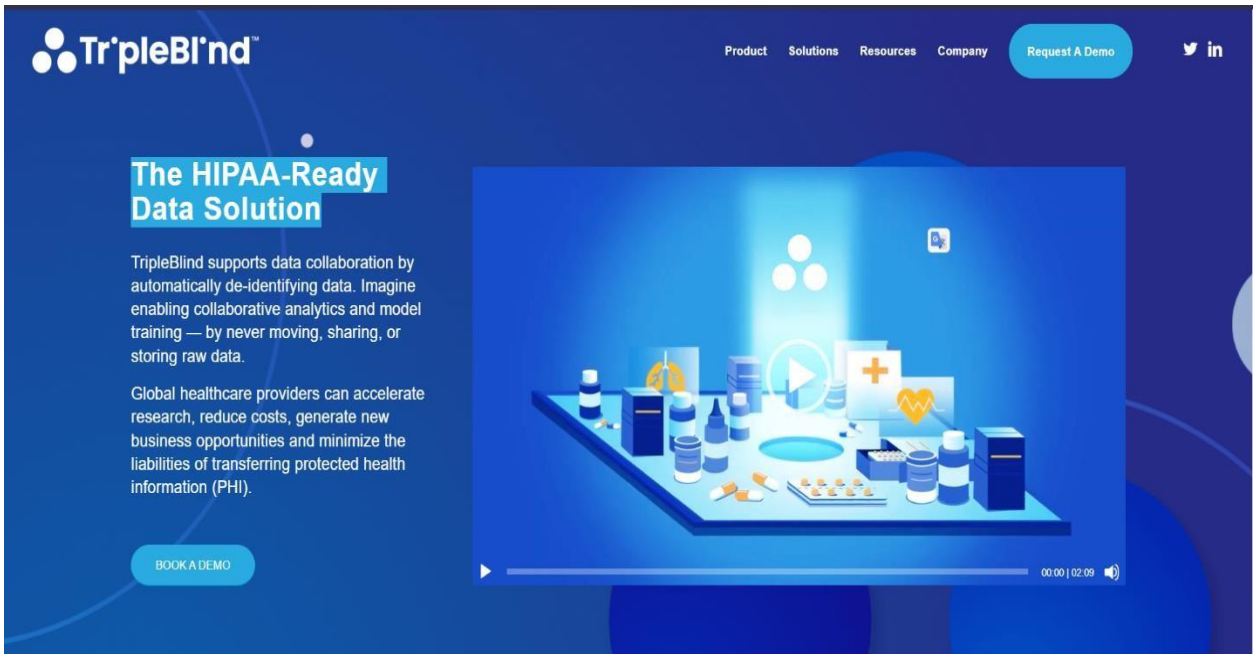


Figure 24 TripleBlind homepage (Tripleblind)

2.2.4 TrailTwin

TrailTwin is a major software platform that assists the life sciences industries by generating synthetic healthcare data. The synthetic healthcare data generated by TrailTwin is completely compliant with the Health Insurance Portability and Accountability Act of 1996(HIPAA) and the General Data Protection Regulation(GDPR) (TrailTwin, 2022).

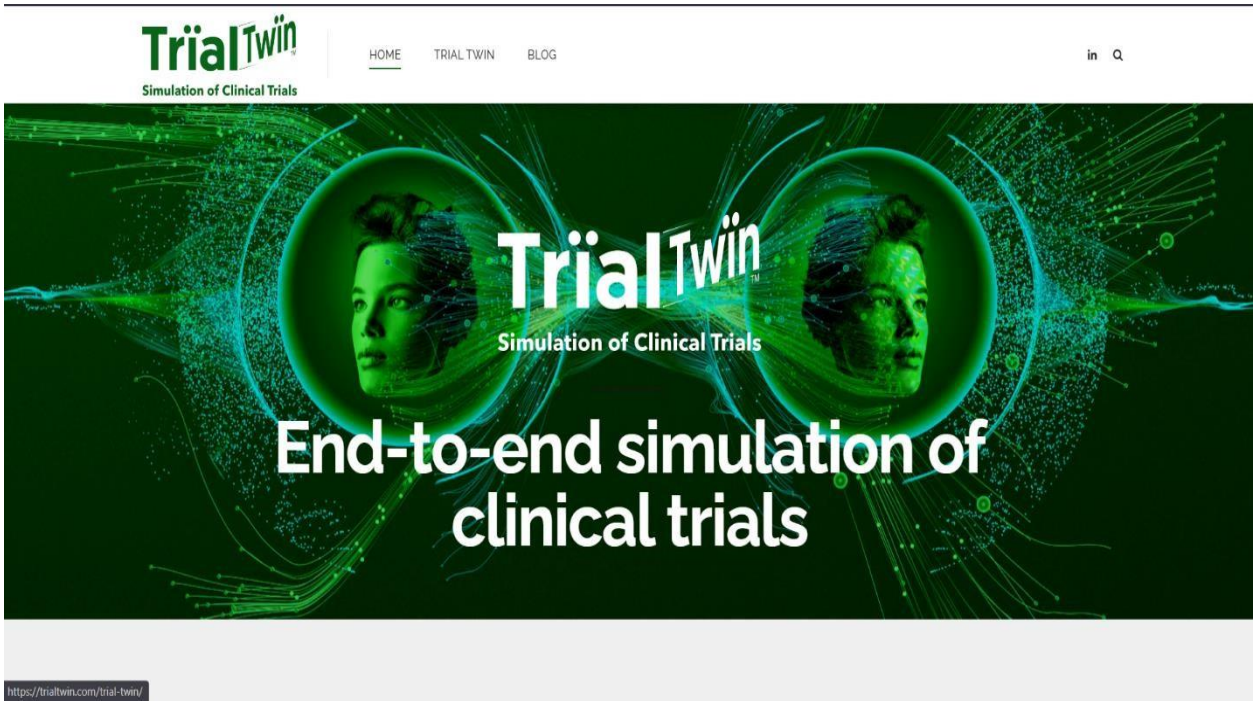


Figure 25 TrailTwin homepage (TrainTwin)

2.2.5 Veil

Viel.ai is a leading European life science and healthcare data driven company that specializes in data anonymization and synthetic data generation. The platform has a patented technology, the VEIL.AI Anonymization Engine, that is responsible for synthesizing real-like high quality data. Some of the key features offered by Veil include secure multiparty anonymization, adaptive anonymization, and AI-based risk management (Viel, 2022).

Figure 26 Viel.AI homepage (Viel)

The complete explanations of CNN and related architectures, and GANs are given in the **Appendix** section.

3. Project Methodology

3.1 The Waterfall methodology

The **waterfall** approach divides a project into logical stages, each of which is dependent upon the successful completion of the stage before it. Since the core of my project is the implementation of several GAN and CNN models, I may need to explore different online GPU services and adjust as necessary when training my models on a given platform proves challenging. Additionally, it is challenging to gather requirements up front for my project because I will be experimenting with various deep learning architectures that require training with various hyperparameters that will change as more research is needed.

3.2 The Agile Methodology

Unlike waterfall which isn't flexible, **Agile methodology** solves such shortcomings for complex projects like mine by providing flexibility and being adaptable to change. I have compared two frameworks of agile methodology: Kanban and Scrum.

3.2.1 Kanban

Kanban is an agile framework that promotes visual mechanisms for managing work as it works through a process. My project will need multiple weeks for researching and training each model such that there will be newer subtasks for each model as I would need to perform hyperparameter tuning. Coding models might also halt for many days as I would need to do sufficient research before starting to code. Furthermore, the Kanban approach believes that requirements and plans are stable and consistent. This does not fit with my project as many changes will need to be made to each model for improving accuracy while training, validating, and testing the model (WisdomPlexus, 2022).

3.2.2 Scrum

Scrum is an agile framework that gives flexibility and is adaptive to change, in contrast to the waterfall methodology. Scrum would benefit me since I could focus on the architecture one sprint at a time because my project involves the implementation of several GAN and CNN architectures. Additional needs introduced by the reader or supervisor during the sprint review meeting would not be a problem because scrum is adaptable. The subsequent sprint can also include newly added requirements. Therefore, I'll be able to effectively complete the research and training of one deep learning model throughout each sprint. Then, I can proceed to complete training of another model in another sprint giving me efficiency as well as time to do necessary research.

3.3 Work Breakdown Structure

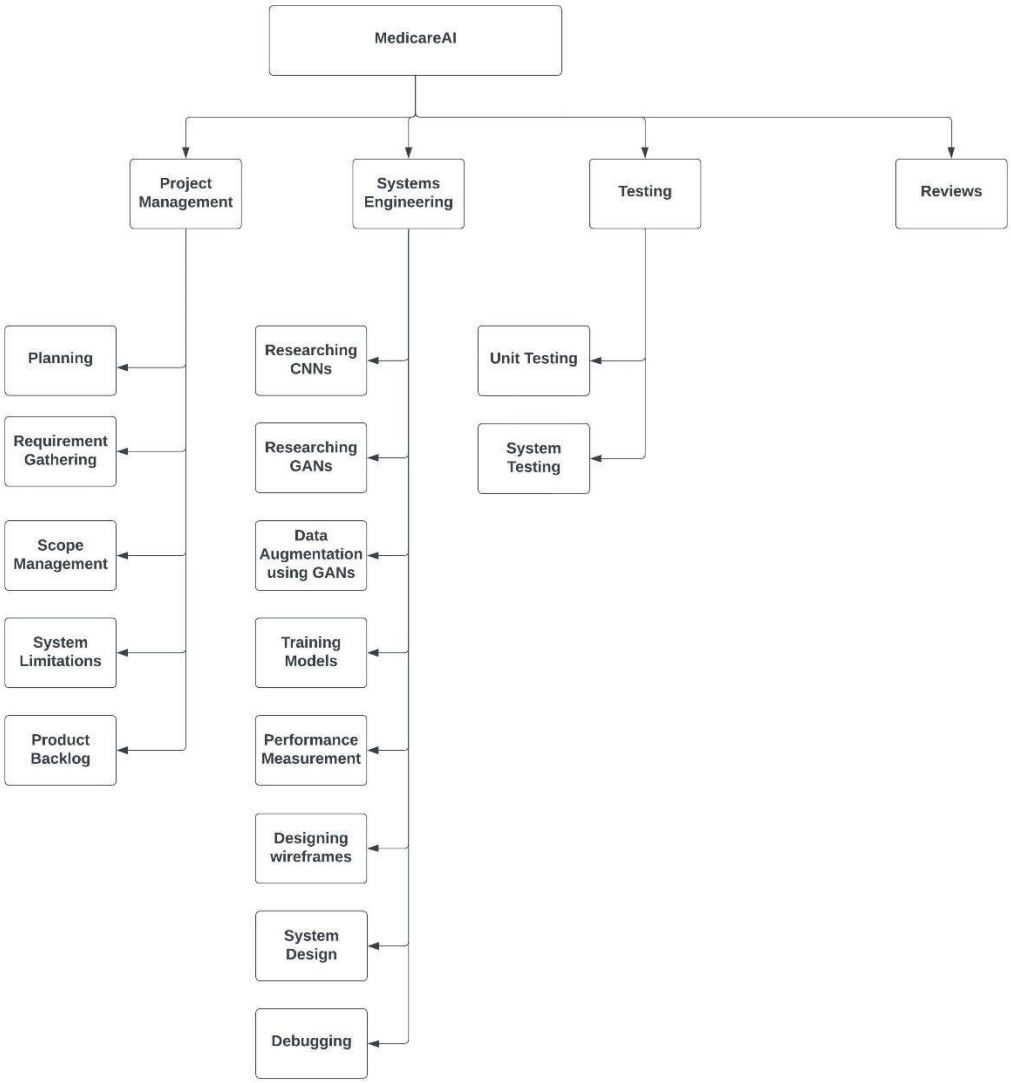


Figure 27 Work Breakdown Structure for MedicareAI

4. Different Technology and Tools used for the project

The tools and technologies used in this project are explained in the table below.

Tools and Technologies	**Brief introduction of selected tools and technologies**
Python	Python is used as the primary language due to its flexibility. It is easy to learn and provides simple yet effective deep learning frameworks such as PyTorch, which is why I chose Python as the primary language for this project.
PyTorch	PyTorch is a python based open-source machine learning framework. I chose PyTorch for this project due to its ease of use, and extensive community support.
Django	Django is an open-source python framework which is used to create web applications. I used Django for this project because it makes it easy to create powerful and user friendly web applications without having to write a lot of code.
HTML, CSS and JavaScript	These tools will be used for created front end portion of the project. I implemented HTML, CSS, and JavaScript to develop the front-end section of my project as they are simple and easy to use and navigate.
MySQL	MySQL database will be used in this project to store data in the database because it has simple queries and easier to learn.
Visual Studio Code	This tool will be used to write code for ML models and web application because it is versatile and provides

	powerful features and extensions to help write productive code.
Jupyter Notebook	Jupyter notebook will be used for training models because it provides me with interactive web-based environment where I can prototype and experiment with ML models.
NumPy, Pandas, Matplotlib, and Seaborn	These python libraries are used to perform array and data manipulation, and visualization because of their flexibility and simplicity.
Git and GitHub	GitHub is a open source version control software that is used to store a git repository. Git is an open-source software that is used to track the changes made to a set of files over a period. They are used in this project because they provide an efficient method for easy collaboration, tracking the changes, and back up of the code in GitHub's online repository.

Table 2 Tools and Technologies

5. Artifact Design

5.1 Software Requirement Specification (SRS)

The SRS is a document that helps capture and explain the complete details and description of the system and it's functionalities (Tutorials Point, 2023).

5.1.1 User and Covid Detection Management System

Requirement code	Description	Moscow prioritization
UCDMS-F-1.0	New users should be allowed to register or create accounts in the system. Existing users should be allowed to log into the system	Must have
UCDMS-F-1.0.1	Username for every user should be unique.	Could have
UCDMS-F-1.0.2	Password must be at least 8 characters long with alphanumeric characters.	Could have
UCDMS-F1.0.3	Email address of a user should be verified.	Could have
UCDMS-F-1.0.4	Proper error alert messages should be displayed in case user makes mistakes while creating an account or logging into the system	could have
UCDMS-F-1.0.5	Verify email address entered by the user.	would have
UCDMS-F-1.1	User should be able to upload x-ray images after logging into the system	Must have

UCDMS-F-1.2	User should be able to log out of the system	Must have
UCDMS-F-1.3	Predicted probabilities for each class should be presented to the user for each prediction.	Should have
UCDMS-F-1.4	An alert window should be displayed to ensure user's request to log out of the system didn't occur by mistake.	Could have
UCDMS-NF-1.5	The web app should be online 24 hours a day and should make accurate predictions.	Could have

Table 3 MedicareAI SRS

5.1.2 Use case diagram

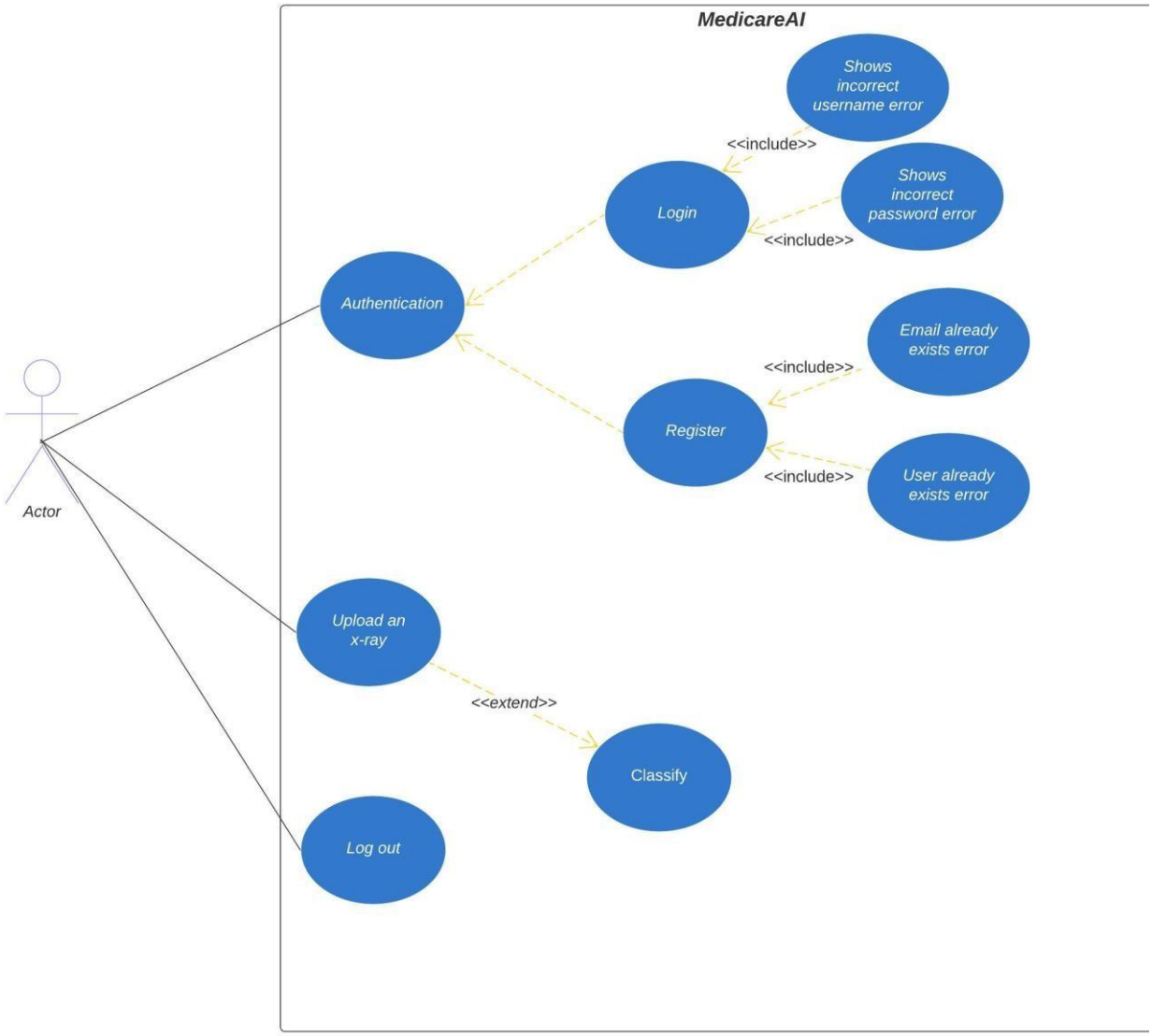


Figure 28 Use case diagram for User and Covid Detection Management System

5.1.3 Activity Diagram

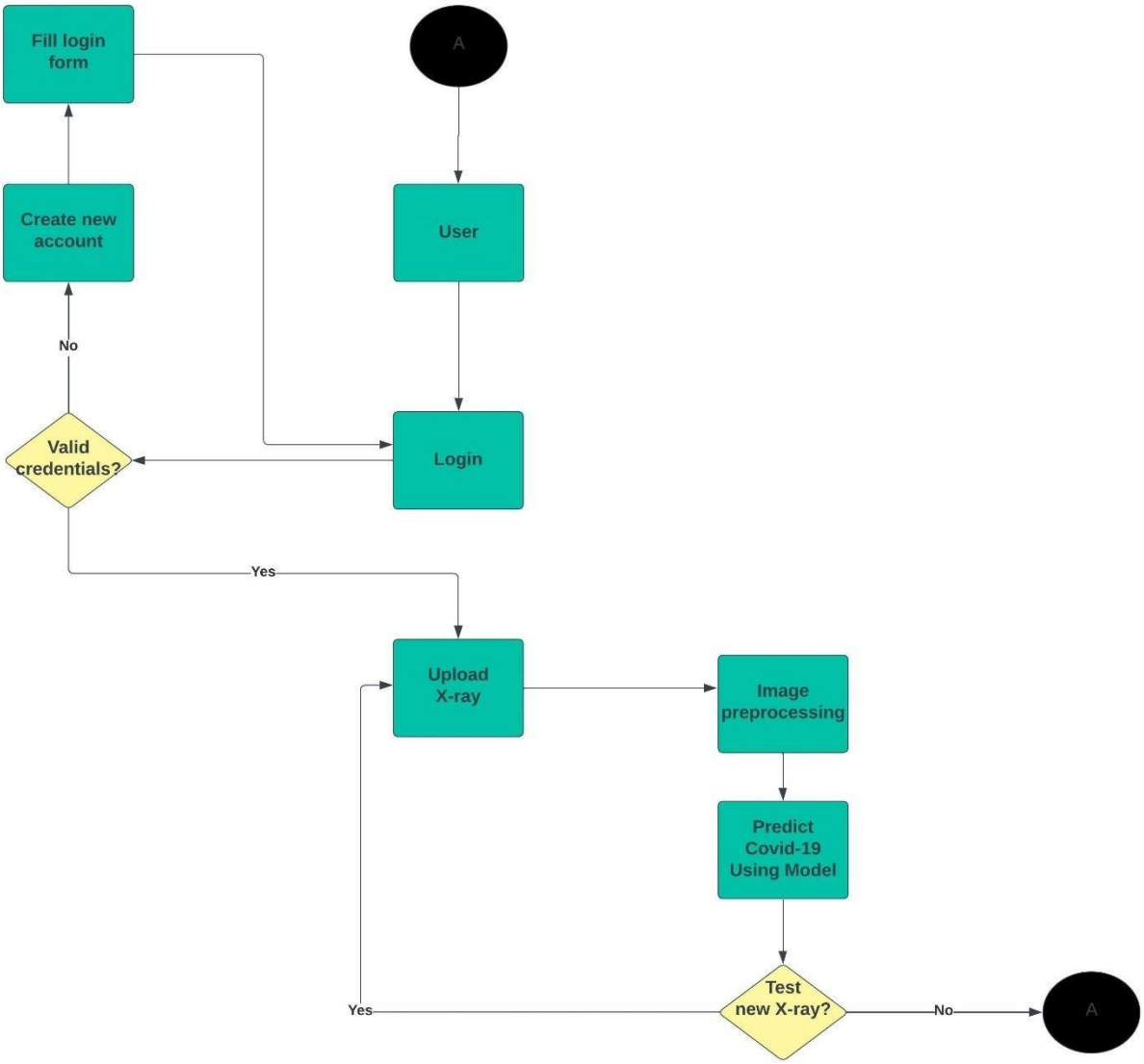


Figure 29 Activity diagram for User and Covid Management System

5.1.4 Sequence Diagram

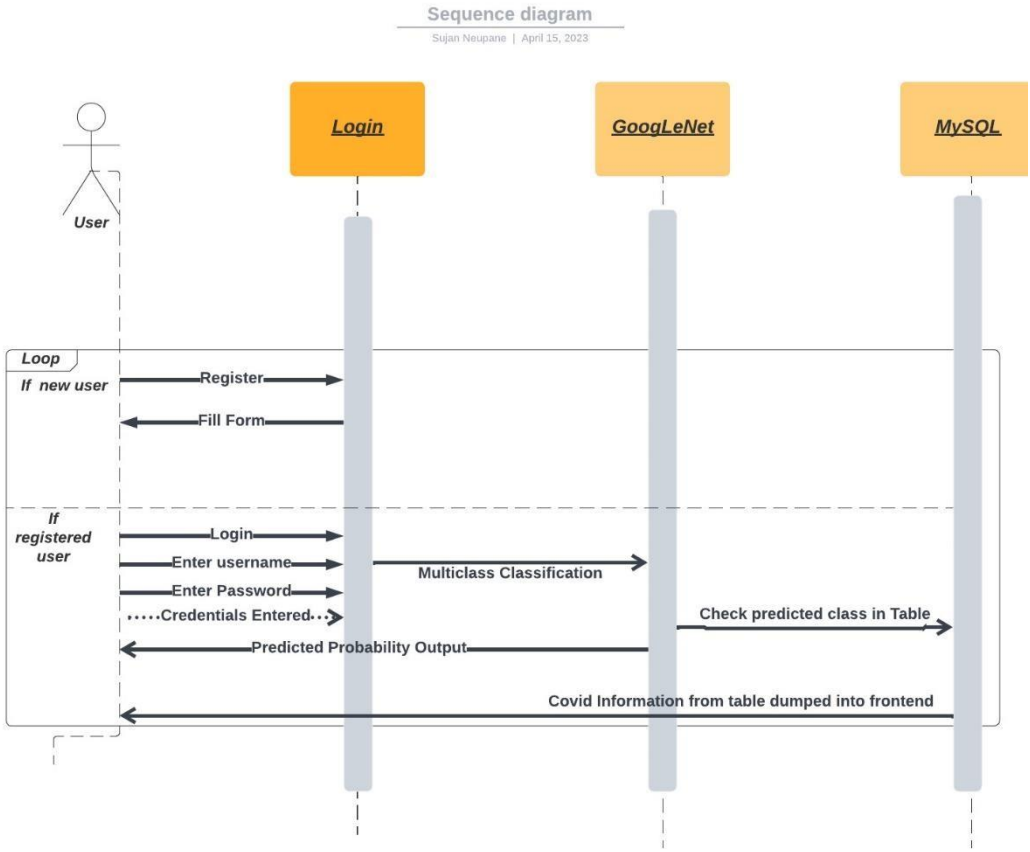


Figure 30 Sequence Diagram for User and Covid Management System

5.1.5 Entity Relationship Diagram (ERD)

The ERD diagram for this project is given below.

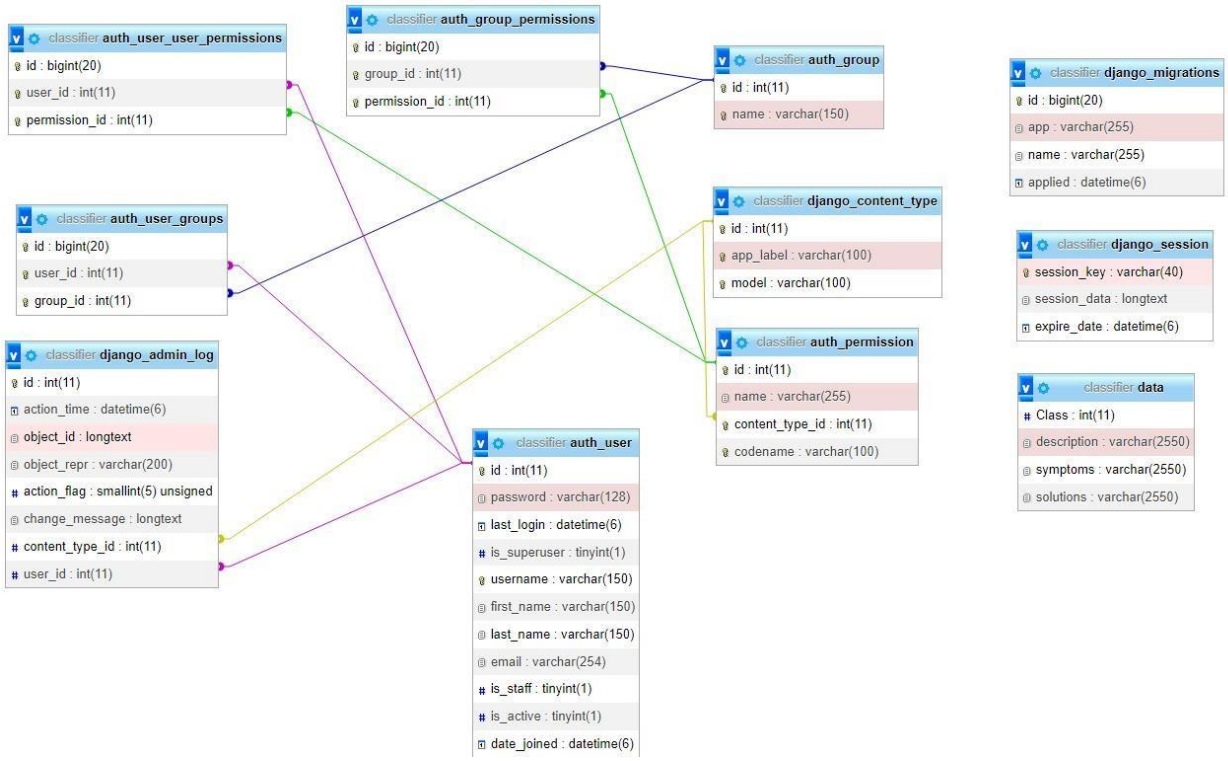


Figure 31 ERD Diagram for MedicareAI

5.1.6 Wireframes

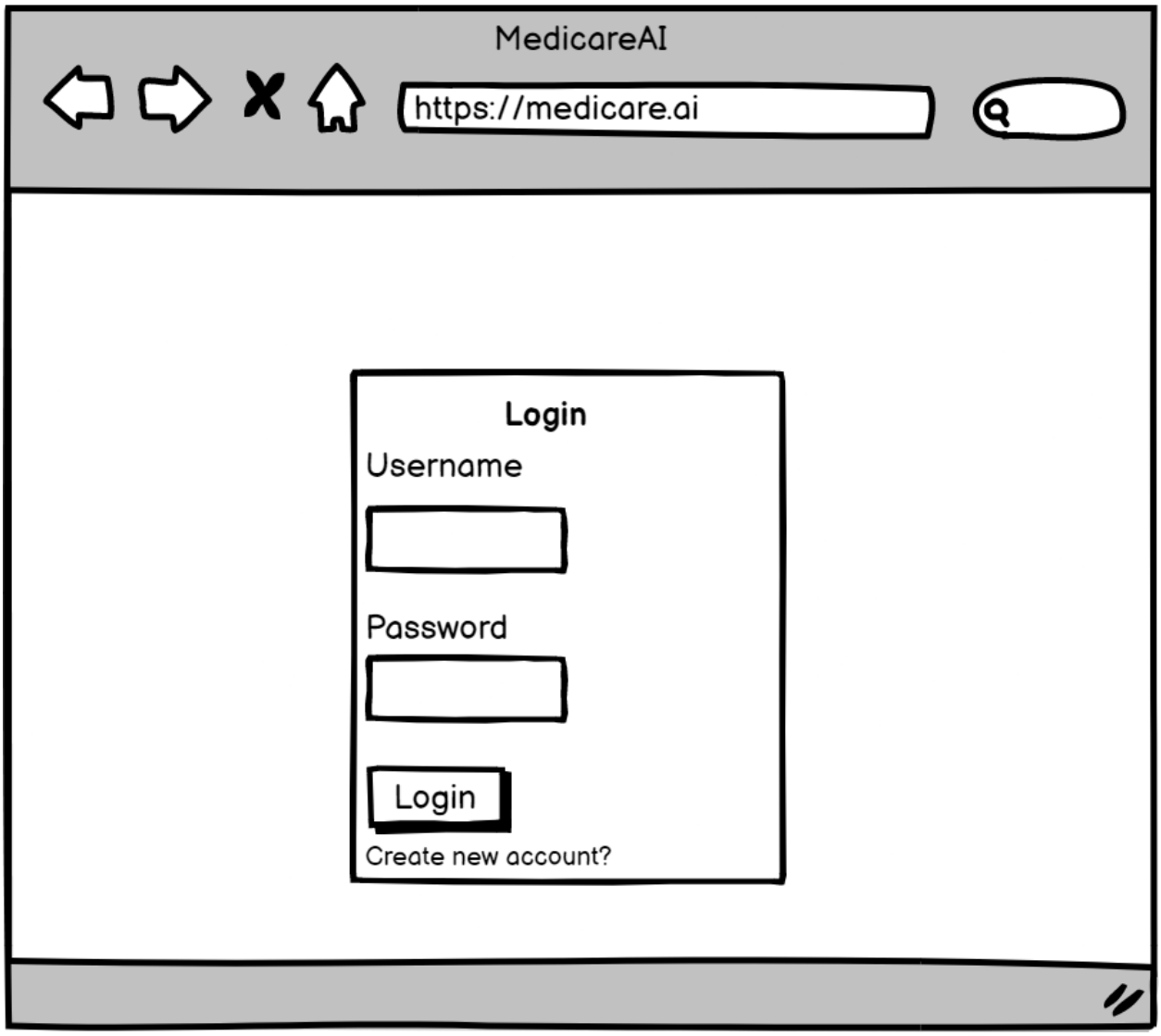


Figure 32 Wireframe number 1 for login

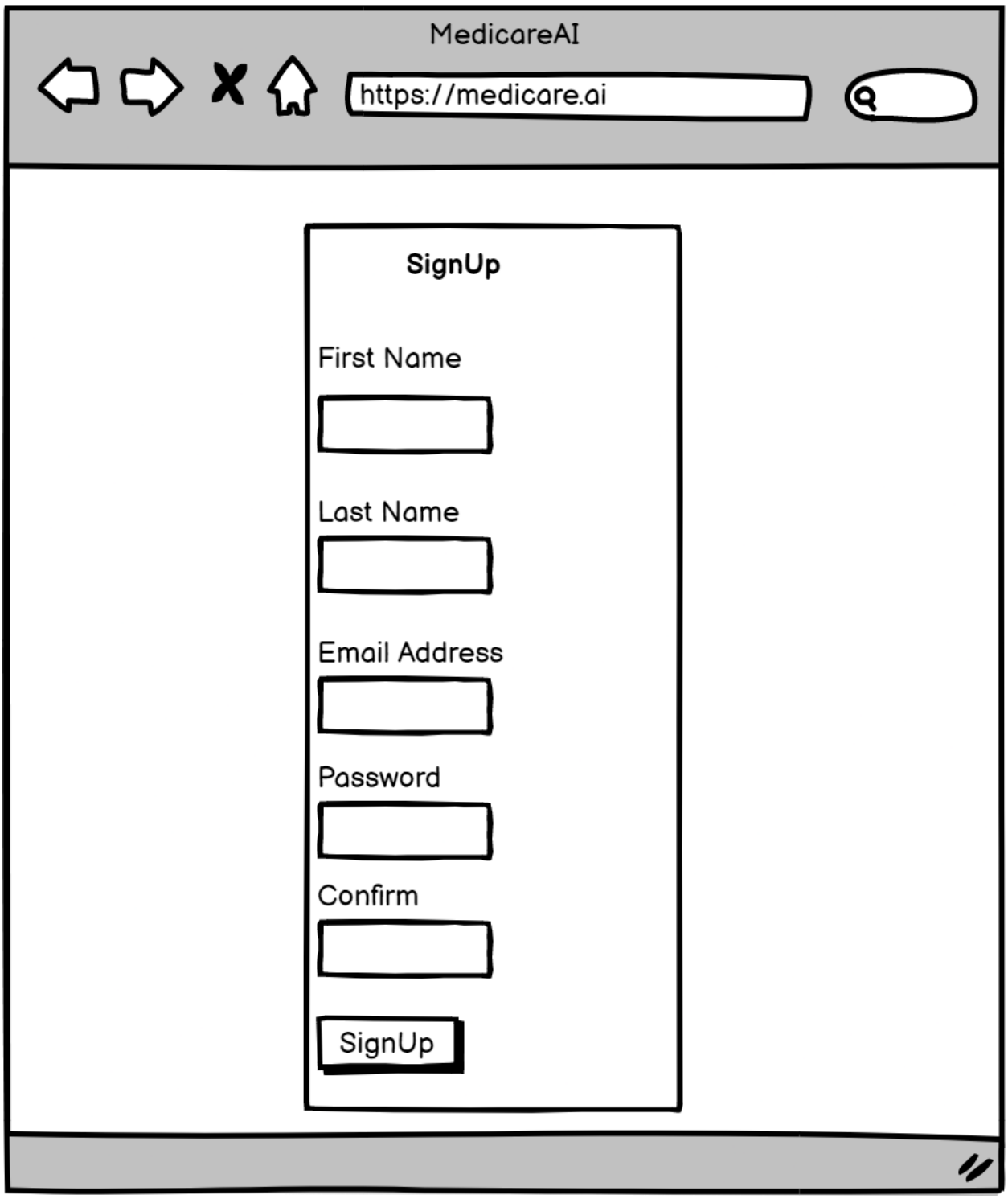


Figure 33 Wireframe number 2 for signup

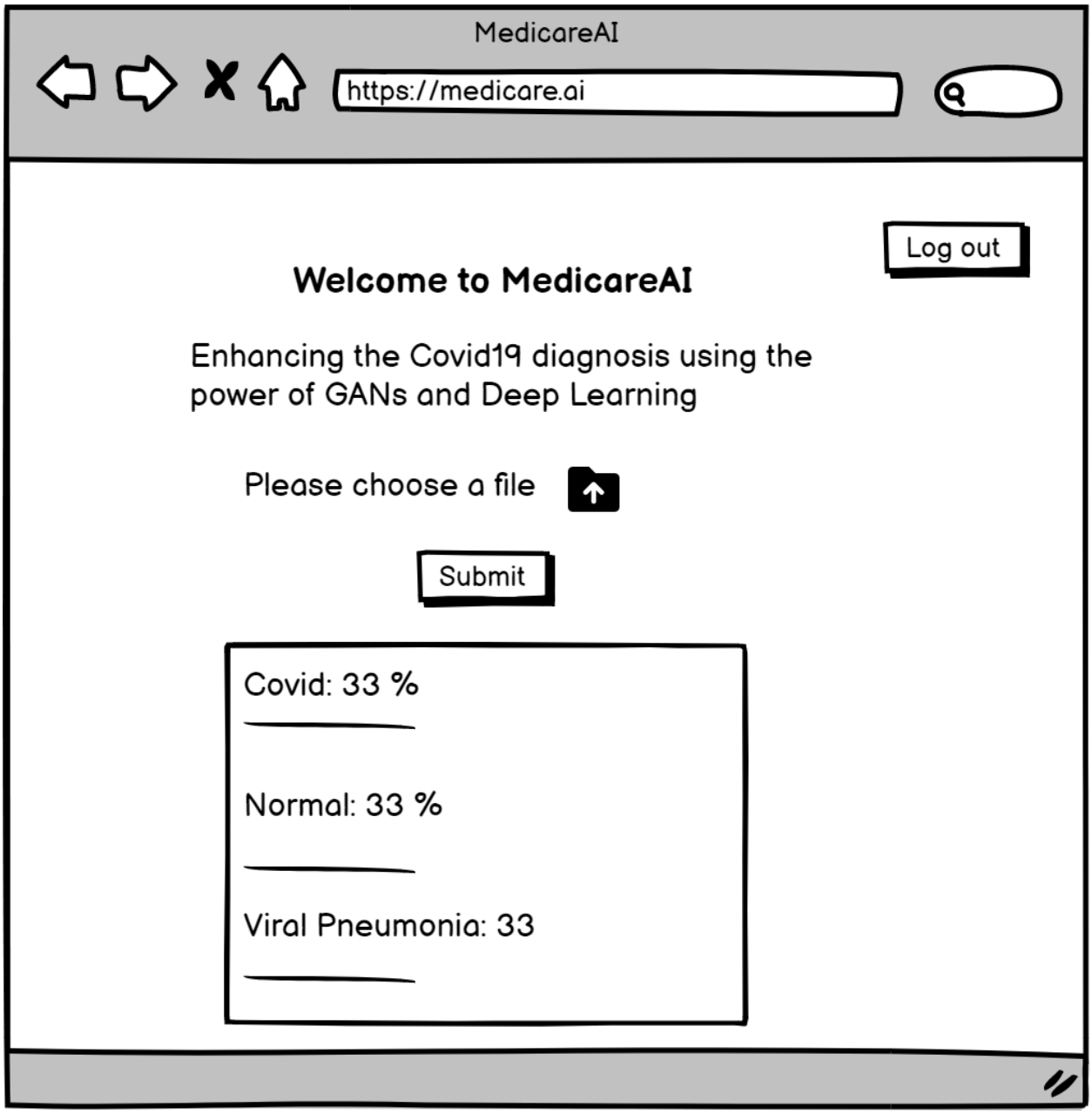
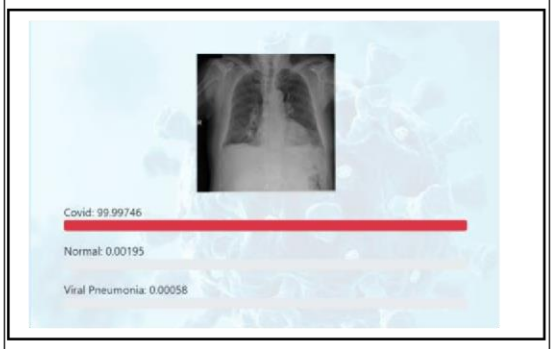
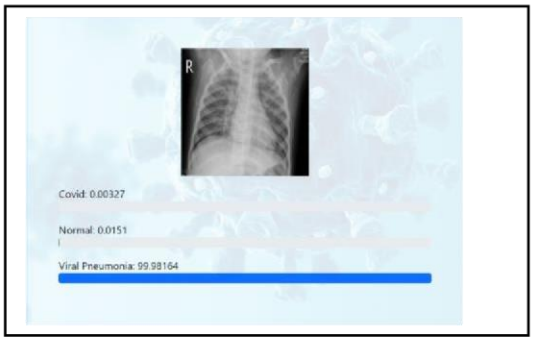


Figure 34 Wireframe number 3 for index.html

5.2 Testing

5.2.1 Black Box Testing

Test Case ID	Description	Test Input	Predicted Output	Actual Output	Pass / Fail
TC001	Testing if system correctly classifies covid or not	 COVID-3218.png	 Prediction: Covid	Covid	**Pass**
TC002	Testing if system correctly classifies covid or not	 COVID-3616.png	 Prediction: Covid	Covid	**Pass**
TC003	Testing if system correctly classifies viral Pneumonia or not	 Viral pneumonia-1.png	 Prediction: Viral Pneumonia	Viral Pneumonia	**Pass**

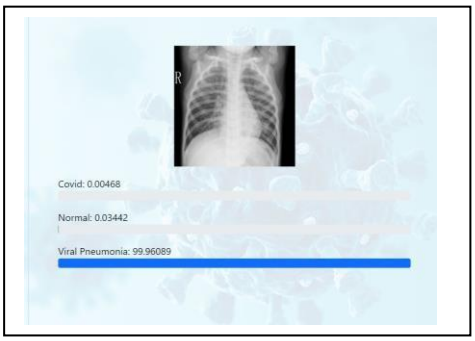
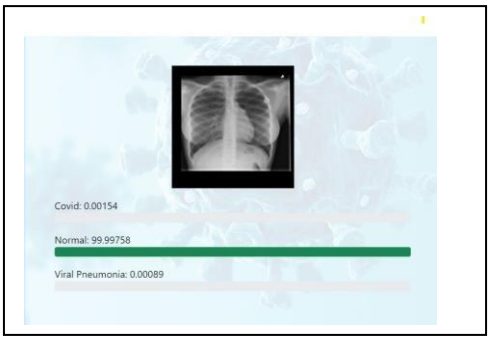
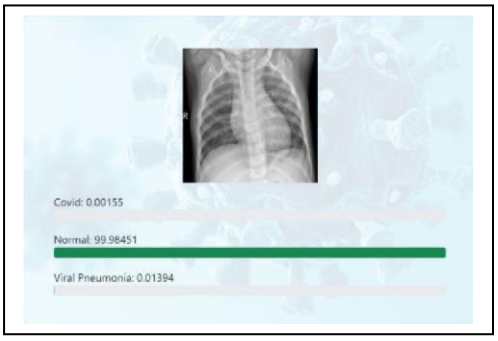
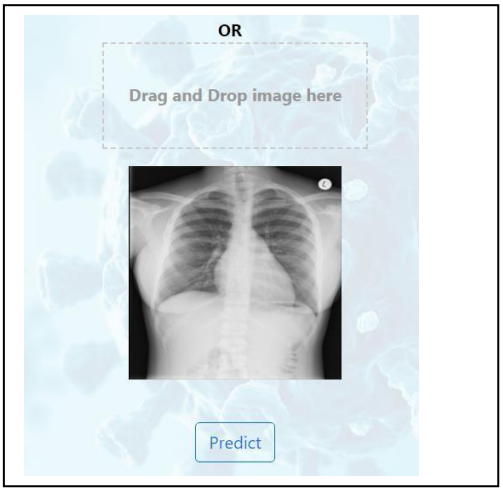
TC004	Testing if system correctly classifies viral Pneumonia or not	 Viral Pneumonia-400.png	 Prediction: Viral Pneumonia	Viral Pneumonia	**Pass**
TC005	Testing if system correctly classifies Normal or not	 Normal-1.png	 Prediction: Normal	Normal	**Pass**
TC006	Testing if system correctly classifies Normal or not	 Normal-400.png	 Prediction: Normal	Normal	**Pass**

Table 4 Black Box Testing for MedicareAI

5.2.2 Functional Testing

In this testing, the drag and drop, and upload image feature will undergo functional testing.

Test Case ID	Description	Test Steps	Expected Results	Actual Results	Pass/ Fail
TC001	Testing if the drag and drop section is clearly visible in the index page or not	Navigating to the index.html page and check drag and drop menu	The drag and drop feature are clearly visible in the index.html page		**Pass**
TC002	Testing if the x-ray images can be dragged and dropped from file explorer or not	Navigating to the images folder through windows explorer and dragging and dropping the image	The selected x-ray image is dragged and dropped successfully.		**Pass**
TC003	Testing if the dragged and drop image is displayed properly in the index.html page or not	Selecting an image and uploading using the drag and drop feature	The selected image is correctly displayed in the index.html page		**Pass**

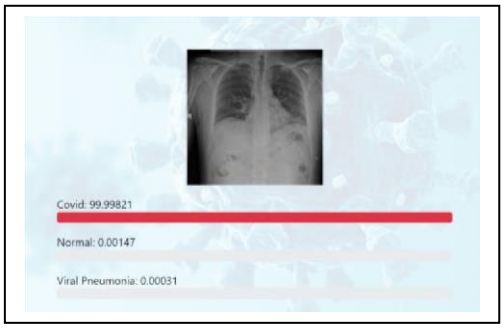
TC004	Testing If the classification model performs correct prediction for the image uploaded using drag and drop feature or not	Uploading a chest x-ray image for covid class and making prediction	The system correctly classifies the chest x-ray uploaded using the drag and drop feature		**Pass**

Table 5 Functional Testing for MedicareAI

5.2.3 Usability and Accessibility Testing

Various accessibility and usability aspects like color-contrast, performance, alternate texts, proper use of caching, and search engine optimization are tested using a chrome store extension called **Lighthouse.** The report for those tests generated by this chrome extension is given below.

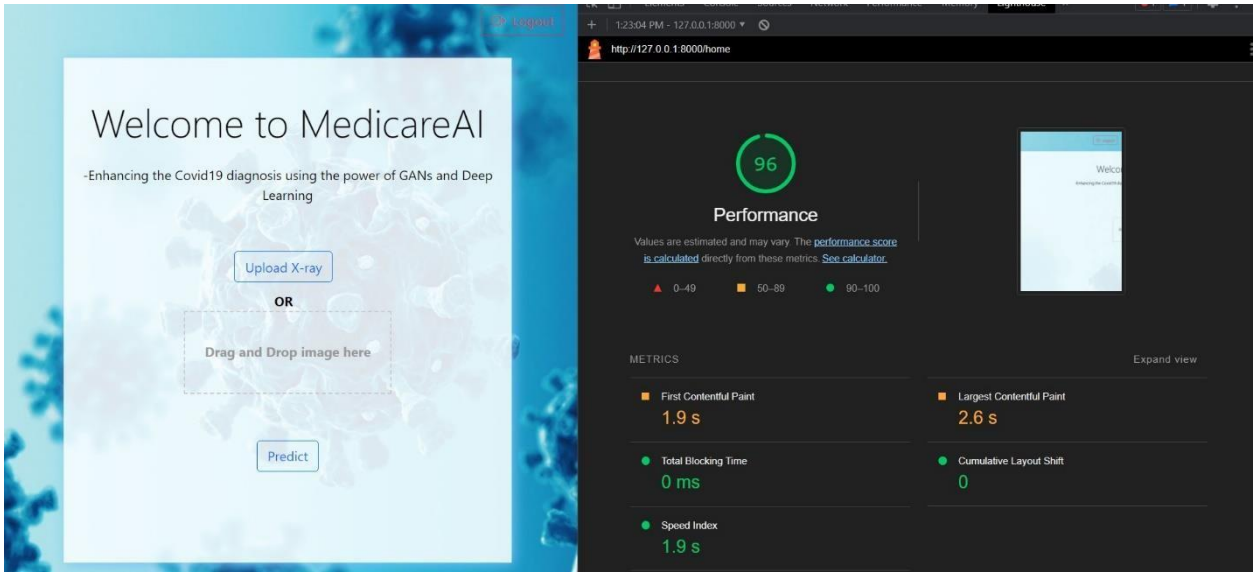


Figure 35 MedicareAI's performance testing using Lighthouse

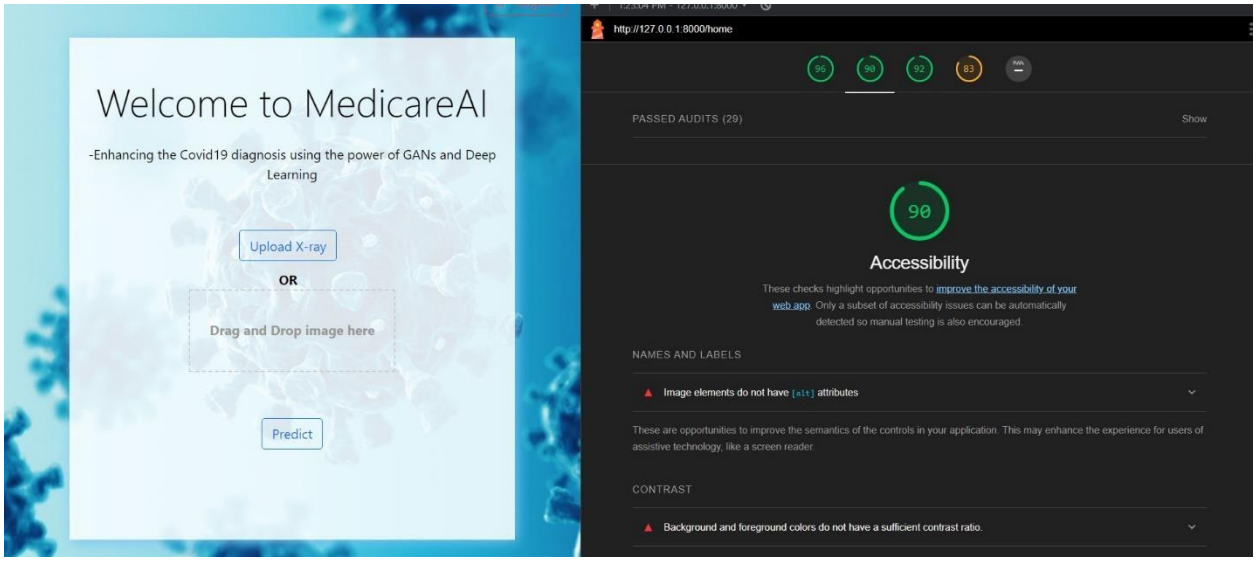


Figure 36 MedicareAI's accessibility testing using Lighthouse

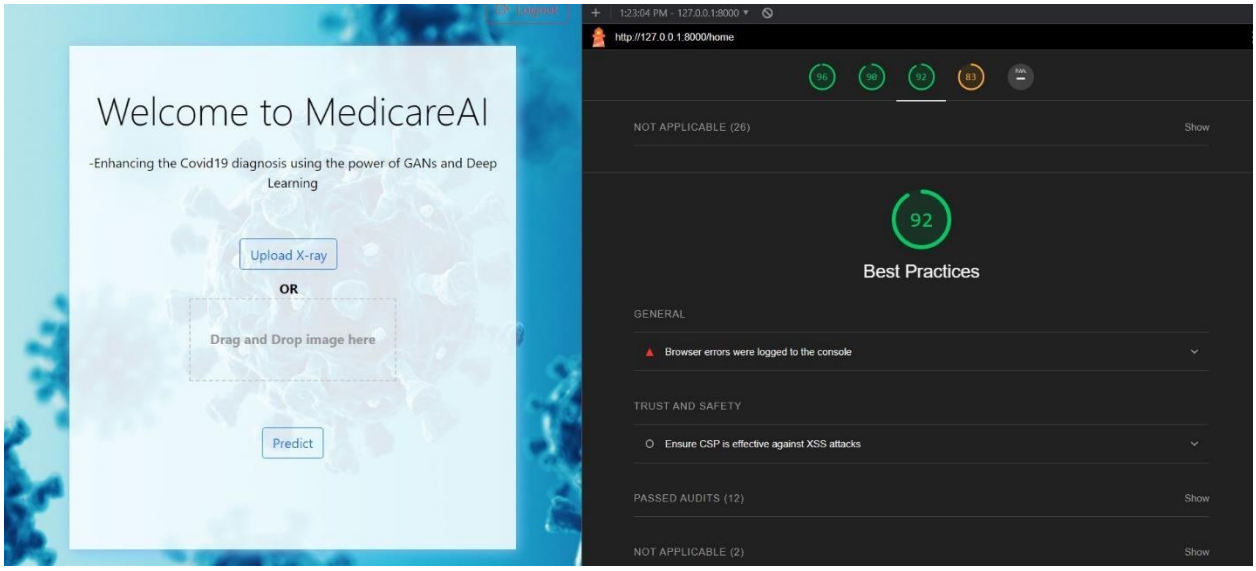


Figure 37 MedicareAI's best practices testing using Lighthouse

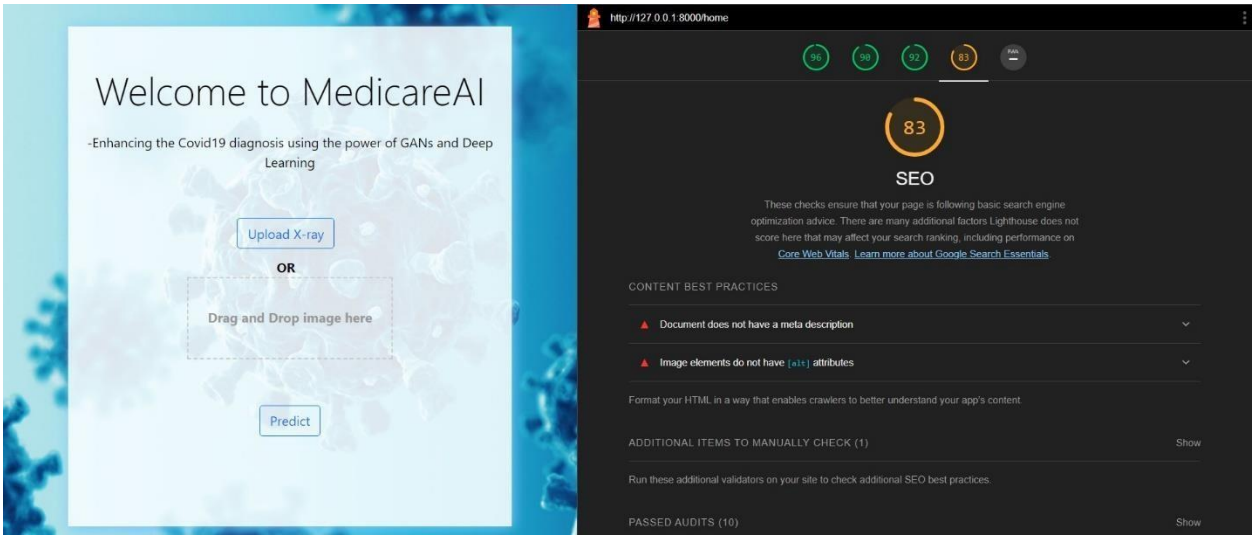


Figure 38 MedicareAI's SEO testing using Lighthouse

In terms of performance, Lighthouse reported this system to be 96 % effective. Likewise, Lighthouse reported this system to be 90 % accessible. Lighthouse also reported this system as 92 % effective in terms of following various web-related best practices. Finally, Lighthouse reported this system to be 83 % effective in terms of search engine optimization.

5.3 Data collection

For developing the EfficientNet model, a public dataset from Kaggle called **COVID-19 Radiography Database** was utilized. The data size is approximately 807 MB in total. The following screenshot contains the folder structure in this dataset.

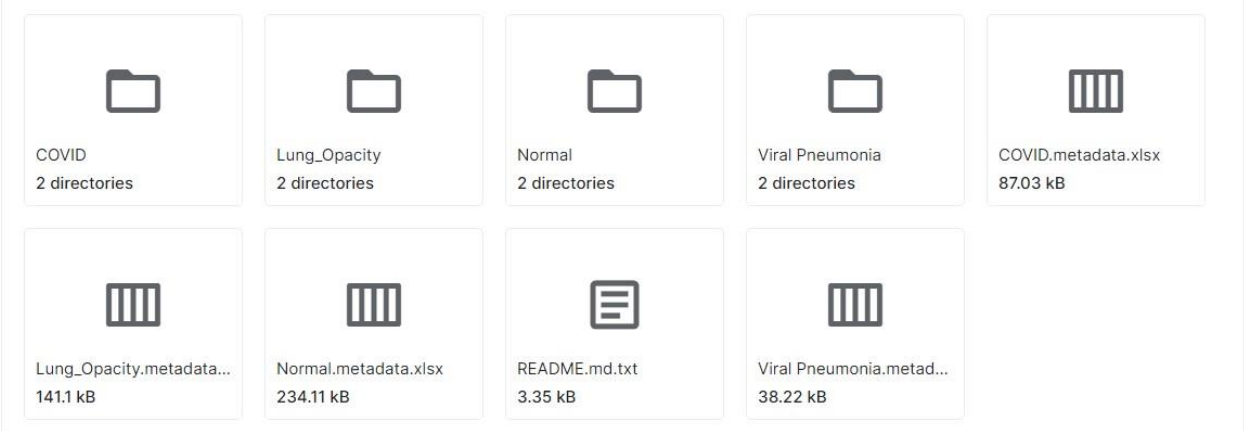


Figure 39 Dataset from Kaggle

5.4 Model development

Three CNN architectures, EfficientNet, ResNet50, and GoogLeNet are used for performing multiclass classification. Each CNN model is trained five times. First, the model is trained on the imbalanced data. Then, the training is followed on data balanced using traditional augmentation techniques like random rotation, horizontal flip and vertical flip. DCGAN, WGAN and WGAN based augmentation are implemented to solve the problem of class imbalance and the model's performance is measured in each of the different training techniques.

5.4.1 Data preparation

Each CNN model is trained five times in this project. It is trained on imbalanced data, data that is balanced using traditional augmentation techniques, data augmented using DCGAN, WGAN and WGAN-GP. The performance of each model on each of these datasets is thoroughly compared. Before feeding the data to the model, several techniques are also applied to the data. Additionally, different regularization techniques are also implemented while training the model. They are explained below.

5.4.2 Data pre-processing

The original dataset's images have dimensions of 1 * 300 * 300. DCGAN, WGAN, and WGAN-GP-based data augmentation techniques are implemented in this project which can generate images with dimensions 1 * 64 * 64. Therefore, all of the images in the original datasets are resized to 64 * 64 pixels in height and width. However, the EfficientNet model takes in an image tensor of n * 224 * 224. So, a technique called **zero padding** is applied to all image tensors. The original image tensors of size 1 * 64 * 64 are applied zero padding of 80 zero pixels around the image such that the output size converts to 224. The screenshot below showcases a custom class in PyTorch to implement zero padding.

```
class CustomDatasetWithPadding(Dataset):
    def __init__(self, data_dir, transforms):
        self.data = []
        self.transform = transforms
        self.targets = []

        class_list = os.listdir(data_dir) # ['Covid', 'Normal', 'Viral Pneumonia']
        for class_name in class_list:
            class_path = os.path.join(data_dir, class_name)
            for img_path in glob.glob(class_path + "/*.png"):
                self.data.append([img_path, class_name])

        self.available_classes = os.listdir(data_dir)

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        img_path, class_name = self.data[idx]
        img = self.transform(Image.open(img_path))
        grey_image = np.zeros([1, 224, 224], dtype = np.float64)
        grey_image[:,80:144, 80:144] = img
        grey_image = torch.from_numpy(grey_image)
        grey_image = grey_image.type(torch.FloatTensor)
        # grey_image = grey_image.type(torch.cuda.FloatTensor)

        class_id = self.available_classes.index(class_name)
        class_id = torch.tensor(class_id)


        return grey_image, class_id
```

Figure 40 Custom class to perform zero padding

5.4.3 Normalization

All image tensors underwent normalization to improve the performance of the CNN model. Normalization is done by subtracting each pixel by mean and dividing by standard deviation such that the mean becomes 0 and standard deviation becomes 1. The mean and standard deviation of entire dataset across batch, width and height is calculated and used to perform normalization.

```
In [ ]:
def get_mean_and_std(dataloader):
    channels_sum, channels_squared_sum, num_batches = 0, 0, 0
    for data, _ in dataloader:

        # Mean over batch, height and width, but not over the channels
        channels_sum += torch.mean(data, dim=[0,2,3])
        channels_squared_sum += torch.mean(data**2, dim=[0,2,3])
        num_batches += 1

    mean = channels_sum / num_batches

    # std = sqrt(E[X^2] - (E[X])^2)
    std = (channels_squared_sum / num_batches - mean ** 2) ** 0.5

    return mean, std
```

```
In [9]:
# creating the train dataset and dataloader for calculating the mean and standard deviation of our samples
image_size = 64
batch_size = 100
CHANNELS_IMG = 1
```

```
In [ ]:
train_ds = ImageFolder(train_data_dir, transform=tt.Compose([
                                        tt.Grayscale(num_output_channels=1),
                                        tt.Resize(image_size),
                                        tt.ToTensor()]))

train_dl = DataLoader(train_ds, batch_size, shuffle=True, num_workers=4, pin_memory=True)
```

```
In [ ]:
mean, std = get_mean_and_std(train_dl)
```

```
In [ ]:
print("The calculated mean is: ", mean)
print("The calculated standard deviation is: ", std)
```

```
The calculated mean is:  tensor([0.5184])
The calculated standard deviation is:  tensor([0.2531])
```

Mean = 0.5184, Std = 0.2531

Figure 41 Calculating mean and standard deviation for imbalanced training dataset

```
# following transformations will be performed on each image
transformations_to_perform = transform=tt.Compose([
                                    tt.Grayscale(num_output_channels=1),
                                    tt.Resize(image_size),
                                    tt.ToTensor(),
                                    tt.Normalize(*stats)])

```

Figure 42 Applying normalization

Similarly, the same technique is implemented while training CNN on GAN based augmented data and traditional augmentation-based data.

5.4.4 Data augmentation

In the training set, there are a total of 2816, 9392, and 545 samples for Covid, Normal and Viral Pneumonia classes. To balance the sample count across each class, three traditional augmentation techniques are also applied: Horizontal flip, vertical flip and random rotation. Additionally, DCGAN, WGAN and WGAN-GP are also used to augment data for balancing sample count across each class. 2192 and 9392 new samples are generated for Covid and Viral Pneumonia respectively.

```
''' ----------------------------CLASS FOR COVID CLASS-------------------------------- '''
class CustomDatasetForCovidClass(Dataset):
    def __init__(self, data_dir):
      self.available_classes = os.listdir(data_dir)
      self.selected_augmentations = [tt.RandomHorizontalFlip(p = 1), tt.RandomVerticalFlip(p = 1), tt.RandomRotation(degrees
      self.data = []
      self.transform = tt.Compose([tt.Grayscale(num_output_channels=1), tt.Resize(64), tt.ToTensor()])
      self.targets = []

      class_list = ['Covid']
      for class_name in class_list:
          class_path = os.path.join(data_dir, class_name)
          for img_path in glob.glob(class_path + "/*.png"):
              self.data.append([img_path, class_name, 'None'])

      self.data_horizontal_flip = random.choices(self.data, k = 2192).copy()
      self.data_horizontal_flip = [list(map(lambda x: x.replace('None', 'Horizontal_Flip'), local)) for local in self.data_h
      self.data_vertical_flip = random.choices(self.data, k = 2192).copy()
      self.data_vertical_flip = [list(map(lambda x: x.replace('None', 'Vertical_Flip'), local)) for local in self.data_verti
      self.data_random_rotations = random.choices(self.data, k = 2192).copy()
      self.data_random_rotations = [list(map(lambda x: x.replace('None', 'Random_Rotations'), local)) for local in self.data
      self.concatenated_data = [*self.data , *self.data_horizontal_flip, *self.data_vertical_flip, *self.data_random_rotatio

    def __len__(self):
        return len(self.concatenated_data)

    def __getitem__(self, idx):
        img_path, class_name, type_of_augmentation = self.concatenated_data[idx]
        img = self.transform(Image.open(img_path))
        if type_of_augmentation == 'Horizontal_Flip':
          img = self.selected_augmentations[0](img)
        elif type_of_augmentation == 'Vertical_Flip':
          img = self.selected_augmentations[1](img)
        elif type_of_augmentation == 'Random_Rotations':
          img = self.selected_augmentations[2](img)

        class_id = self.available_classes.index(class_name)
        return img, class_id
```

Figure 43 Traditional augmentation for covid class

```
''' ----------------------------CLASS FOR Viral Pneumonia CLASS------------------------------- '''

class CustomDatasetForViralPneumoniaClass(Dataset):
    def __init__(self, data_dir):
      self.available_classes = os.listdir(data_dir)
      self.selected_augmentations = [tt.RandomHorizontalFlip(p = 1), tt.RandomVerticalFlip(p = 1), tt.RandomRotation(degrees
      self.data = []
      self.transform = tt.Compose([tt.Grayscale(num_output_channels=1), tt.Resize(64), tt.ToTensor()])
      self.targets = []

      class_list = ['Viral Pneumonia']
      for class_name in class_list:
          class_path = os.path.join(data_dir, class_name)
          for img_path in glob.glob(class_path + "/*.png"):
              self.data.append([img_path, class_name, 'None'])
      self.data_horizontal_flip = random.choices(self.data, k = 2949).copy()
      self.data_horizontal_flip = [list(map(lambda x: x.replace('None', 'Horizontal_Flip'), local)) for local in self.data_h
      self.data_vertical_flip = random.choices(self.data, k = 2949).copy()
      self.data_vertical_flip = [list(map(lambda x: x.replace('None', 'Vertical_Flip'), local)) for local in self.data_verti
      self.data_random_rotations = random.choices(self.data, k = 2949).copy()
      self.data_random_rotations = [list(map(lambda x: x.replace('None', 'Random_Rotations'), local)) for local in self.data
      self.concatenated_data = [*self.data , *self.data_horizontal_flip, *self.data_vertical_flip, *self.data_random_rotatio

    def __len__(self):
        return len(self.concatenated_data)

    def __getitem__(self, idx):
        img_path, class_name, type_of_augmentation = self.concatenated_data[idx]
        img = self.transform(Image.open(img_path))
        if type_of_augmentation == 'Horizontal_Flip':
          img = self.selected_augmentations[0](img)
        elif type_of_augmentation == 'Vertical_Flip':
          img = self.selected_augmentations[1](img)
        elif type_of_augmentation == 'Random_Rotations':
          img = self.selected_augmentations[2](img)

        class_id = self.available_classes.index(class_name)
        return img, class_id
```

Figure 44 Traditional augmentation for Viral Pneumonia class

For DCGAN, WGAN and WGAN-GP, the weights of generator model of each GAN architecture are loaded first. Then, the required number of random noise vector from a normal distribution is passed onto the generator model to get the generated image tensor, which will undergo same preprocessing as original data.

```
device = 'cpu'
n_classes = 2
latent_size = 100
generator_input_dim = 103
CHANNELS_IMG = 1
stats_gen = [0.5 for _ in range(CHANNELS_IMG)], [0.5 for _ in range(CHANNELS_IMG)]

def denorm(img_tensors):
    return img_tensors * stats_gen[1][0] + stats_gen[0][0]

''' FOR 64 * 64 Image Generation '''
generator = nn.Sequential(
    nn.ConvTranspose2d(102, 512, kernel_size=4, stride=1, padding=0, bias=False),
    nn.BatchNorm2d(512),
    nn.ReLU(True),

    nn.ConvTranspose2d(512, 256, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(256),
    nn.ReLU(True),

    nn.ConvTranspose2d(256, 128, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(128),
    nn.ReLU(True),

    nn.ConvTranspose2d(128, 64, kernel_size=4, stride=2, padding=1, bias=False),
    nn.BatchNorm2d(64),
    nn.ReLU(True),

    nn.ConvTranspose2d(64, 1, kernel_size=4, stride=2, padding=1, bias=False),
    nn.Tanh()
)

generator.load_state_dict(torch.load('cDCGAN_generator.pth', map_location ='cpu'))
generator.eval()
latent_size = 100
generator_input_dim = 102
```

Figure 45 Loading DCGAN's generator model

```
def get_an_image(class_, noise):
  label = torch.tensor([class_ for i in range(noise.shape[0])])
  one_hot_labels = F.one_hot(label.to(device), n_classes)

  #
  noise_and_labels = torch.cat([noise, one_hot_labels.float()],dim=1)
  noise_and_labels = noise_and_labels.view(len(noise_and_labels), generator_input_dim, 1, 1)
  fake_images = denorm(generator(noise_and_labels))

  return fake_images.detach().to('cpu')
```

```
%%time

torch.manual_seed(42)
noise_Covid = torch.randn(6576, 100, device='cpu')
covid_generated_by_DCGAN = get_an_image(0, noise_Covid)
covid_generated_by_DCGAN = [(i, 0) for i in covid_generated_by_DCGAN]
```

```
CPU times: user 44.2 s, sys: 3.5 s, total: 47.7 s
Wall time: 53.7 s
```

```
%%time

torch.manual_seed(84)
noise_Pneumonia = torch.randn(8847, 100, device='cpu')
Pneumonia_generated_by_DCGAN = get_an_image(1, noise_Pneumonia)
Pneumonia_generated_by_DCGAN = [(i, 2) for i in Pneumonia_generated_by_DCGAN]
```

```
CPU times: user 53.4 s, sys: 1.36 s, total: 54.8 s
Wall time: 55.3 s
```

Figure 46 Generating new samples using DCGAN

It can be clearly seen that a total of 6576 new samples for Covid class and 8847 new samples for Viral Pneumonia class are generated using DCGAN. Same code is used to synthesize new data using WGAN and WGAN-GP. Only the weights file is changed with respect to the GAN architecture.

5.4.5 Regularization techniques

Regularization refers to various techniques that make slight modifications to a model such that the model generalizes well on unseen data. Regularization can prevent overfitting by adding a penalty term to the loss function while training a model. The main aim of regularization is to prevent complex models from fitting the noise in the data and make the model generalize well on the data. Various regularization techniques are implemented while training EfficientNet and they are explained briefly below.

Gradient clipping is a technique to clip gradients to prevent them from becoming too large during training. Larger gradients can lead the model to diverge and produce unstable results during training. This technique handles large gradients by capping them to a maximum value and solve the problem of **exploding gradient problem**.

Weight decay, which is similar to **L2 regularization,** is a regularization technique used to regularize the size of a model's weights. In L2 regularization, weights of a model are squared and summed and multiplied by a coefficient, which gets added to the loss function as a penalty term that is to be reduced or minimized. L2 regularization is added to the loss function whereas weight decay is directly added to the update rule in gradient descent. Weight decay helps to prevent overfitting by adding a term to the optimizer that will penalize larger weight values. Therefore, the model will be motivated to have smaller values of weights which will reduce overfitting (Bajaj, 2022) .

Dropout is another type of regularization technique to prevent overfitting which works by randomly setting the outputs of neurons to zero during each forward pass.

5.4.6 Model training and optimization

Each CNN model is trained for a total of 20 epochs. Cross entropy loss is used as the model's loss function. One-cycle learning rate scheduling was implemented with a maximum learning rate set to 0.0001. When we start training the model, the learning rate gradually increased with respect to total number of batches across all epochs. The learning rate will be maximum when the model is trained on half of all available batches. Then, the learning rate reduces and approaches zero during the end of training. The main aim of this technique is that we want to take larger steps in gradient descent when we start training our model and take smaller steps during the end of training to prevent the model from overshooting the global minimum of its loss function. The value of weight decay and gradient clip is set to 1e-4 and 0.1 respectively.

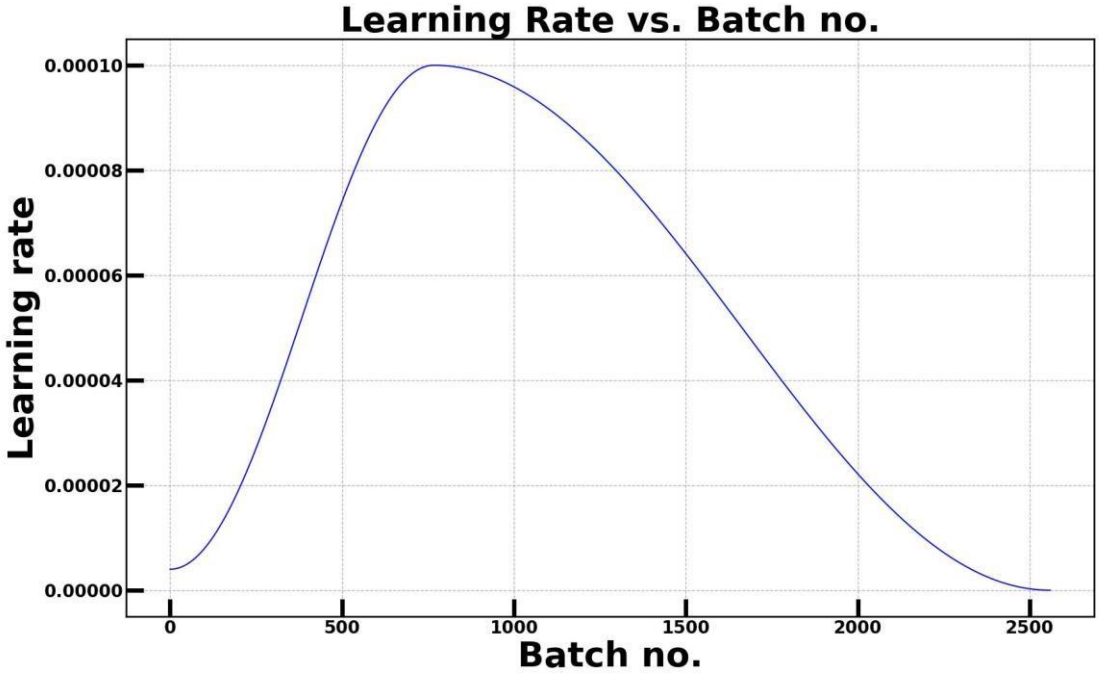


Figure 47 Learning rate scheduling

Initially, a maximum learning rate of 0.01 was implemented. But the model was making huge jumps during gradient descent that resulted in big oscillations in model's training loss and validation accuracy. Therefore, the learning rate was subsequently reduced 10 times twice resulting in new maximum learning rate to be 0.0001. The training and validation loss along with validation accuracy are recorded and plotted using matplotlib python library, which is added below. Left y-axis represents validation accuracy which is in the range of 0 and 100. Right y-axis represents training and validation loss.

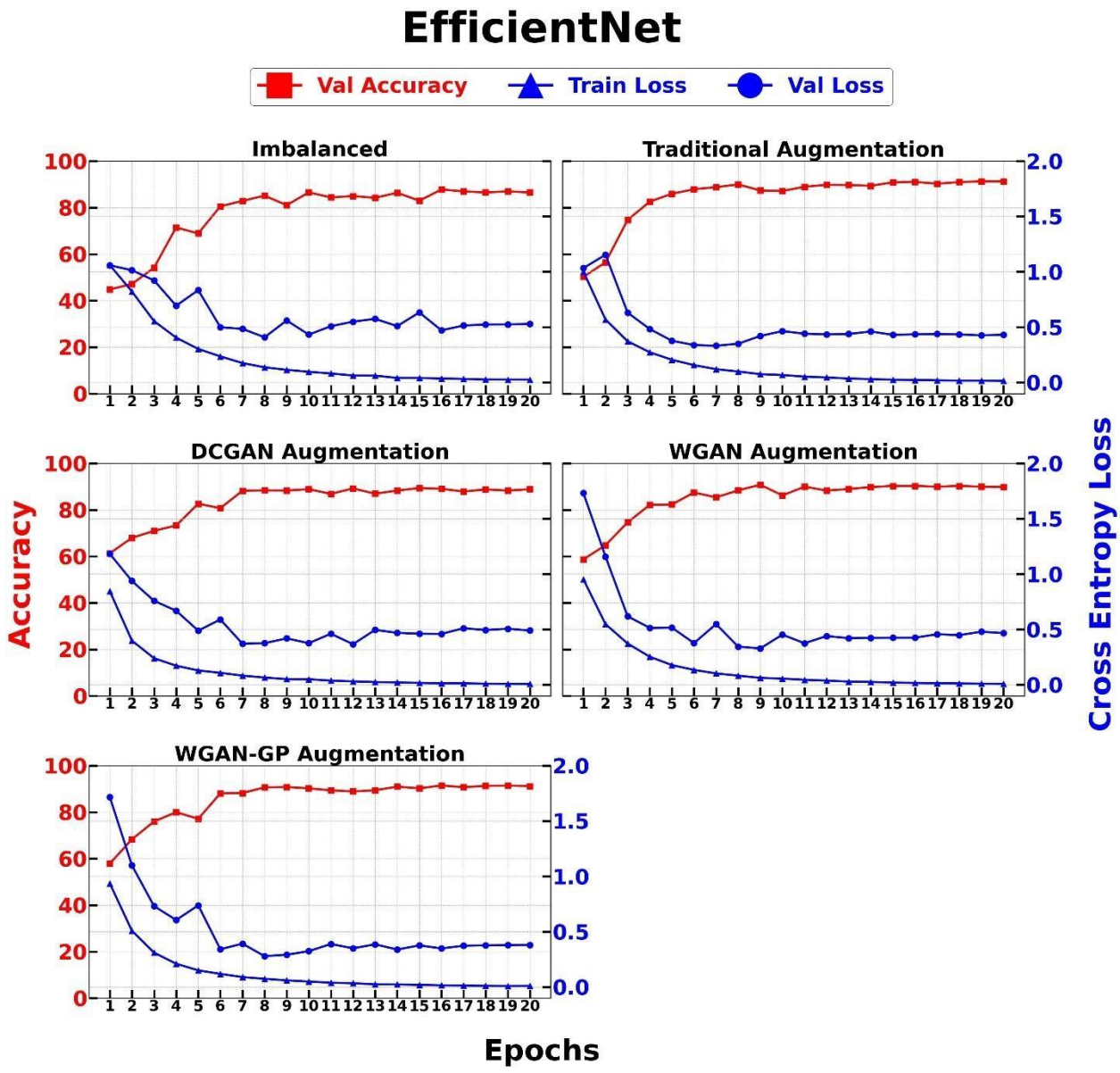


Figure 48 EfficientNet training

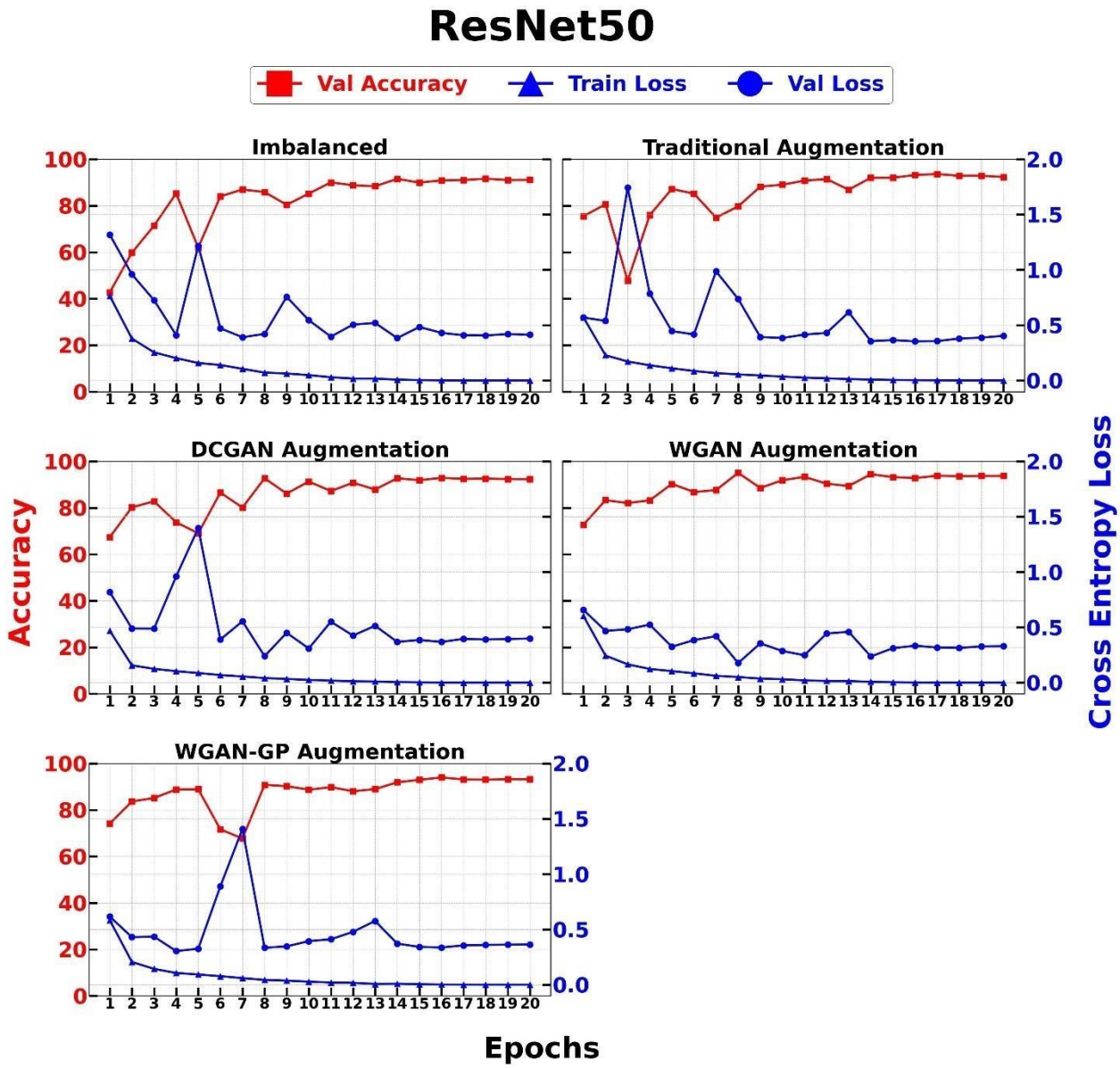


Figure 49 ResNet50 Training

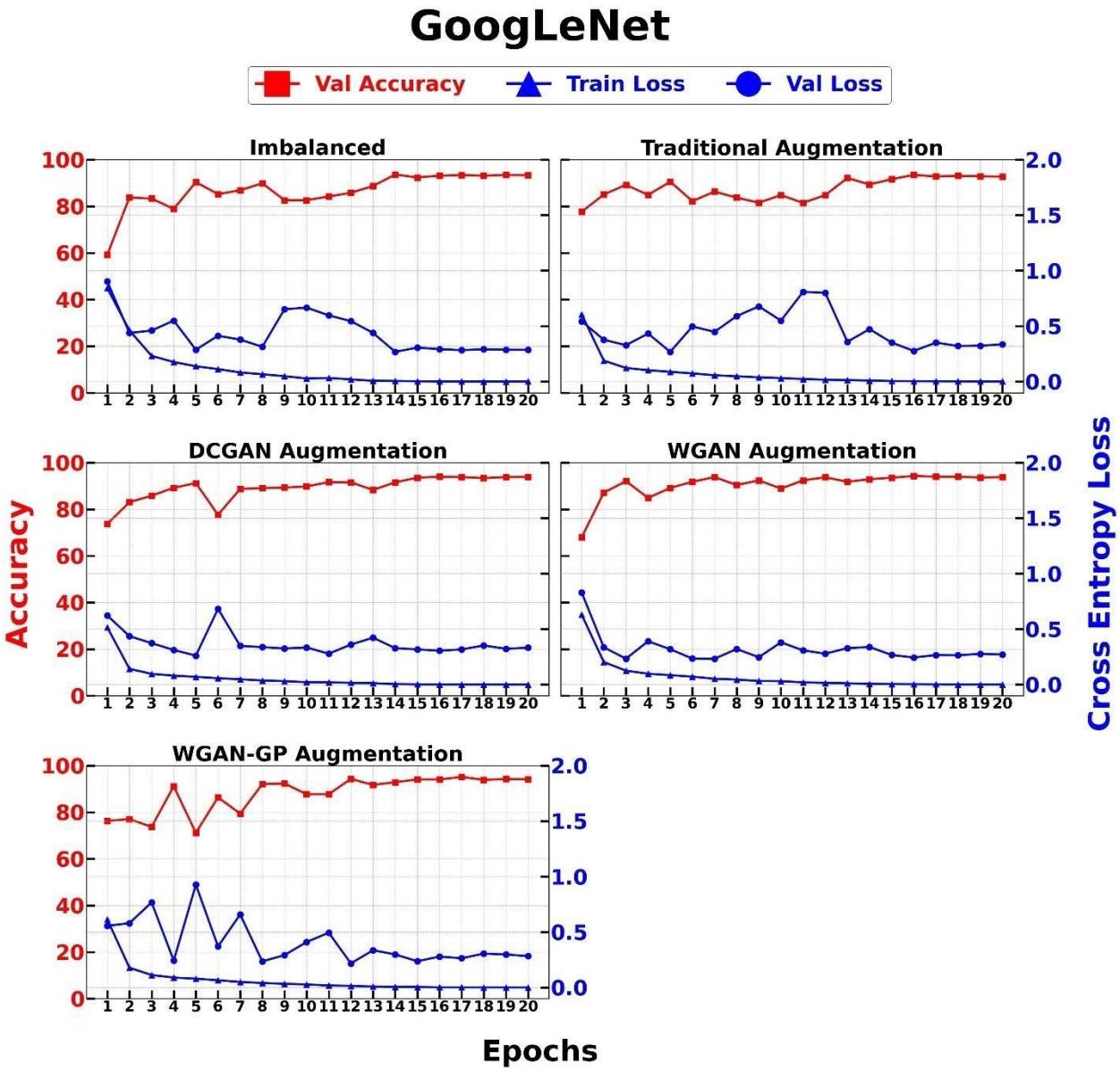


Figure 50 GoogLeNet without Auxiliary classifiers training

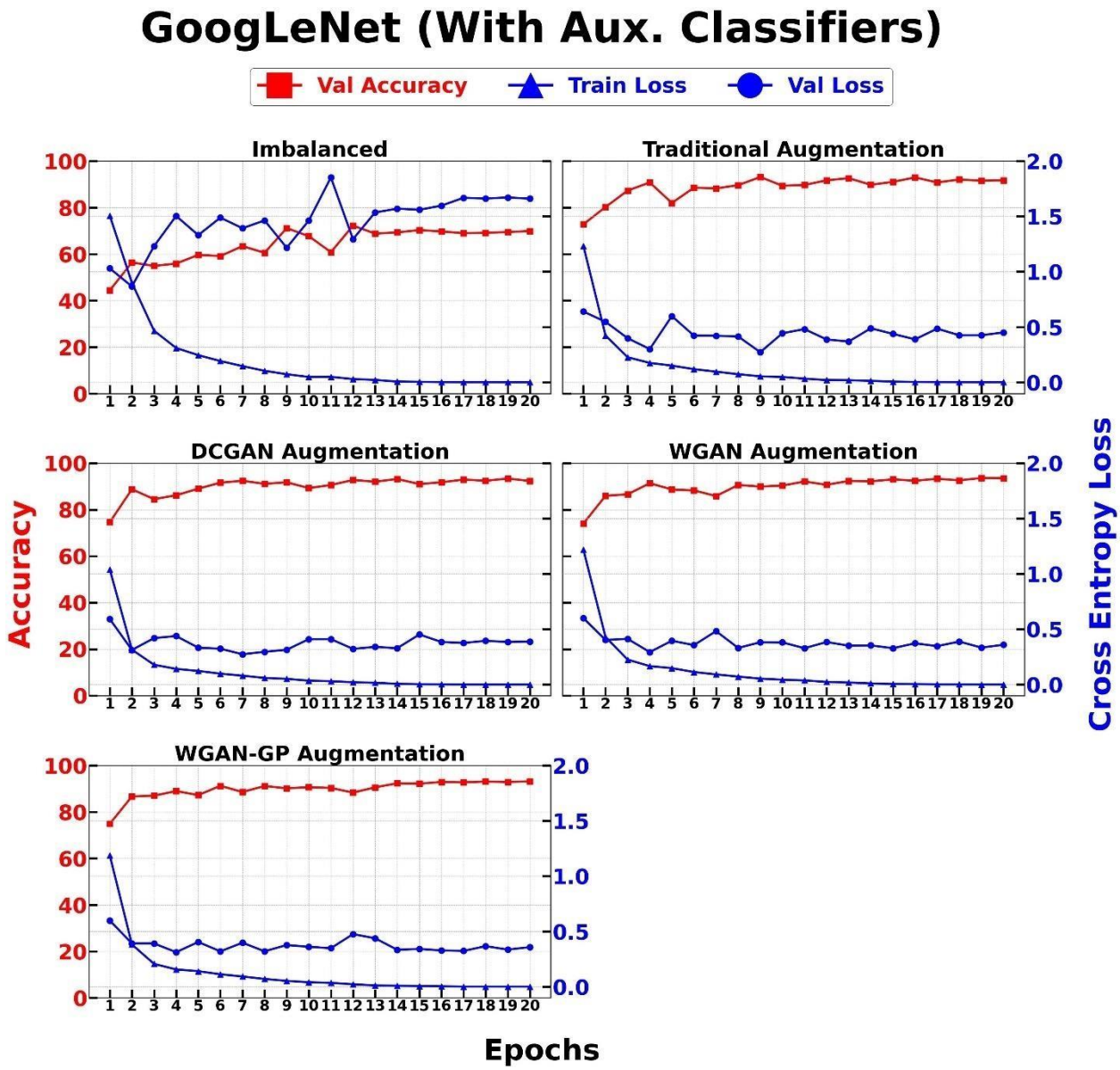


Figure 51 GoogLeNet with Auxiliary classifiers training

We can clearly see that, while training the model on imbalanced data, the validation loss is oscillating very much. The oscillation is still present to a smaller extent on all other cases. Each CNN model trained on imbalanced data has the lowest validation accuracy at the end of training. Similarly, each CNN model trained on traditionally augmented data outperforms the model trained on DCGAN based augmented data at the end of final epoch on validation set. WGAN and WGAN-GP have very similar validation accuracy and their performance will need to be properly accessed using a confusion matrix. It can be

also concluded that augmentation techniques can help enhance the model's performance on the validation data. Traditional augmented data beats the model trained with imbalanced data. Similarly, DCGAN has the lowest performance as compared to WGAN and WGAN-GP. Both GANs, WGAN and WGAN-GP, are very similar in terms of performance as the only difference between them is that the former implements weight clipping, and the latter implements gradient penalty.

5.4.7 DCGAN, WGAN and WGAN-GP development and training

Each of the GAN models was trained for a total of **1500** epochs using the same architecture for the generator and discriminator. The only difference was in terms of the loss function. DCGAN implements the binary cross-entropy loss function and sigmoid activation in the discriminator's last layer. However, as the discriminator improves, the probabilities output by the DCGAN's discriminator approaches 0 or 1, which results in a vanishing gradient problem. Additionally, DCGAN is also much more prone to mode collapse. To solve this problem, WGAN and WGAN-GP implement the W-Loss with weight clipping and gradient penalty. The summary architecture of the generator and discriminator models is given below.

```
In [18]: 1 summary(discriminator, (3, 64, 64))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           3,072
       BatchNorm2d-2           [-1, 64, 32, 32]             128
         LeakyReLU-3           [-1, 64, 32, 32]               0
            Conv2d-4          [-1, 128, 16, 16]         131,072
       BatchNorm2d-5          [-1, 128, 16, 16]             256
         LeakyReLU-6          [-1, 128, 16, 16]               0
            Conv2d-7            [-1, 256, 8, 8]         524,288
       BatchNorm2d-8            [-1, 256, 8, 8]             512
         LeakyReLU-9            [-1, 256, 8, 8]               0
           Conv2d-10            [-1, 512, 4, 4]       2,097,152
      BatchNorm2d-11            [-1, 512, 4, 4]           1,024
        LeakyReLU-12            [-1, 512, 4, 4]               0
           Conv2d-13              [-1, 1, 1, 1]           8,192
          Flatten-14                    [-1, 1]               0
          Sigmoid-15                    [-1, 1]               0
================================================================
Total params: 2,765,696
Trainable params: 2,765,696
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 2.81
Params size (MB): 10.55
Estimated Total Size (MB): 13.41
----------------------------------------------------------------
```

Figure 52 GANs discriminator model summary

```
In [24]: 1 summary(generator, (102, 1, 1))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 512, 4, 4]         835,584
       BatchNorm2d-2            [-1, 512, 4, 4]           1,024
              ReLU-3            [-1, 512, 4, 4]               0
   ConvTranspose2d-4            [-1, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [-1, 256, 8, 8]             512
              ReLU-6            [-1, 256, 8, 8]               0
   ConvTranspose2d-7          [-1, 128, 16, 16]         524,288
       BatchNorm2d-8          [-1, 128, 16, 16]             256
              ReLU-9          [-1, 128, 16, 16]               0
  ConvTranspose2d-10           [-1, 64, 32, 32]         131,072
      BatchNorm2d-11           [-1, 64, 32, 32]             128
             ReLU-12           [-1, 64, 32, 32]               0
  ConvTranspose2d-13            [-1, 1, 64, 64]           1,024
             Tanh-14            [-1, 1, 64, 64]               0
================================================================
Total params: 3,591,040
Trainable params: 3,591,040
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 2.88
Params size (MB): 13.70
Estimated Total Size (MB): 16.57
----------------------------------------------------------------
```

Figure 53 GANs generator model summary

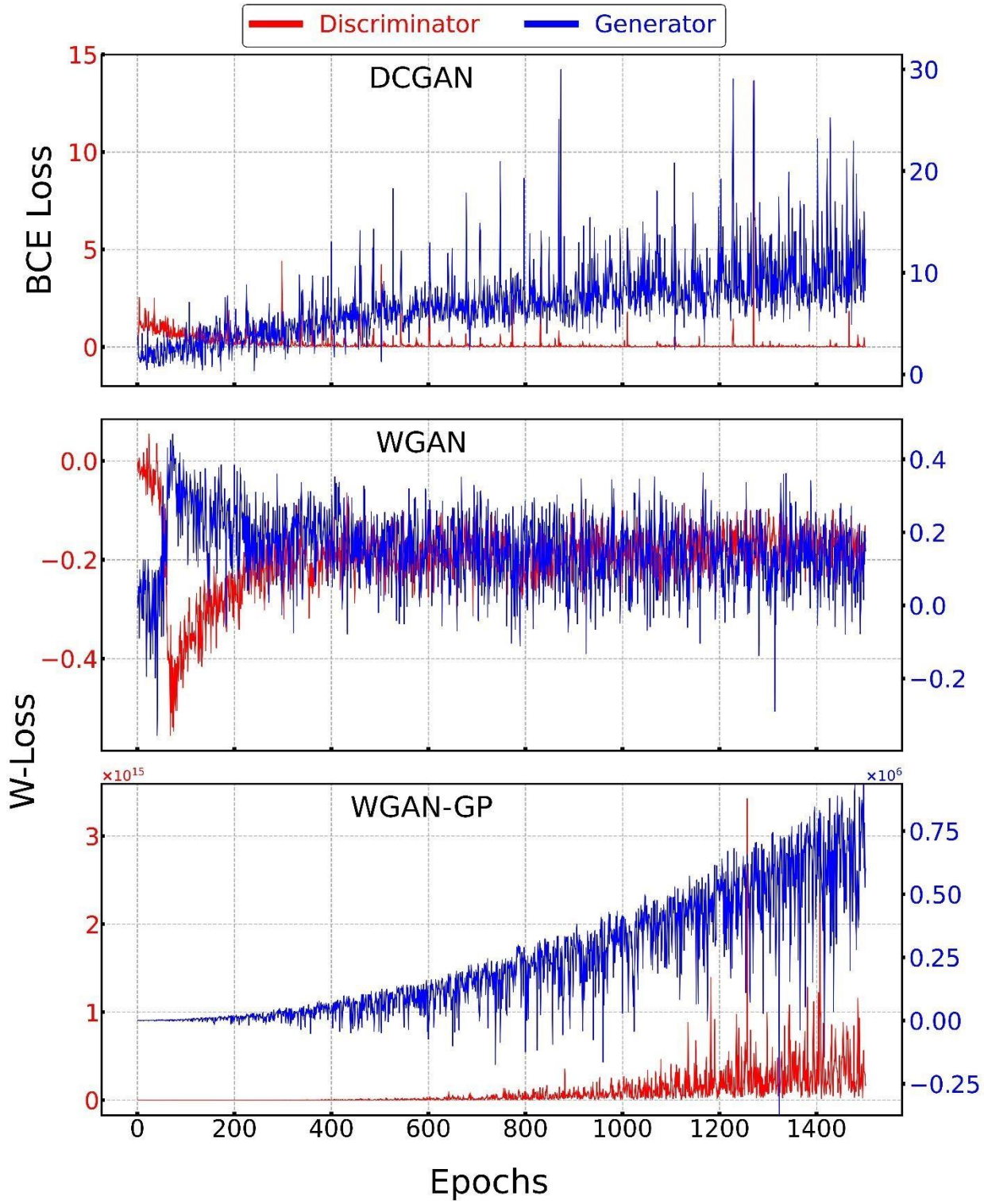


Figure 54 DCGAN, WGAN, and WGAN-GP training

5.4.8 Samples generated using GANs

Each of the three GAN architectures, DCGAN, WGAN and WGAN-GP, were trained for a total of 1500 epochs. While training, in every 10 epochs, a random noise vector is passed into the generator model and a subplot containing images is created and saved. So, a total of approximately 150 images are saved for each GAN model while training. Using **imageio** python library, images for all GAN architectures are converted into gif to visualize the changes in quality of generated images for each architecture while training. The quality of image for final epoch are added below for each of the GAN architecture.

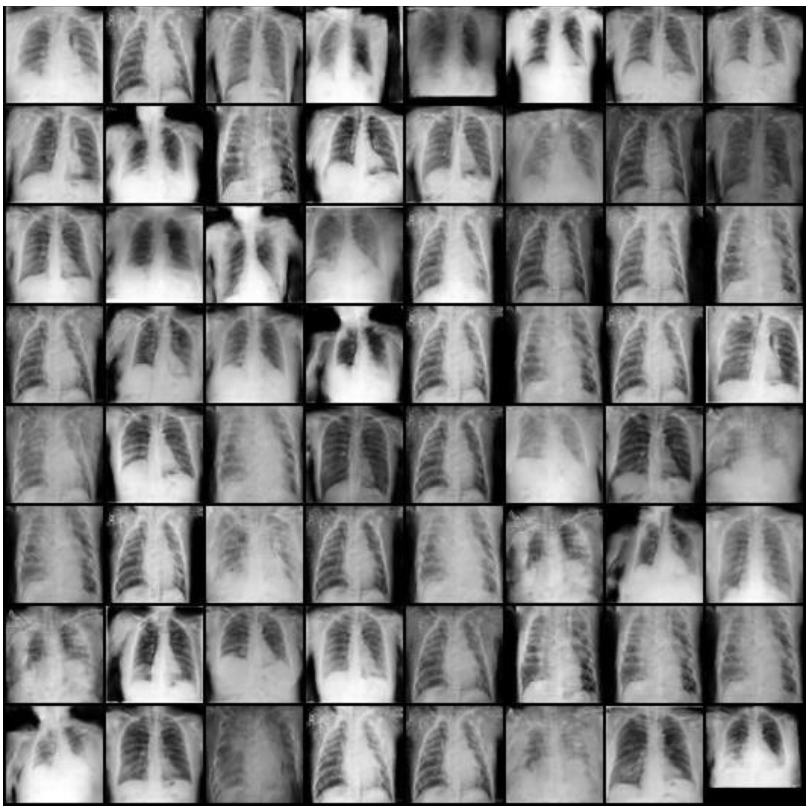


Figure 55 Final epoch generated image for DCGAN

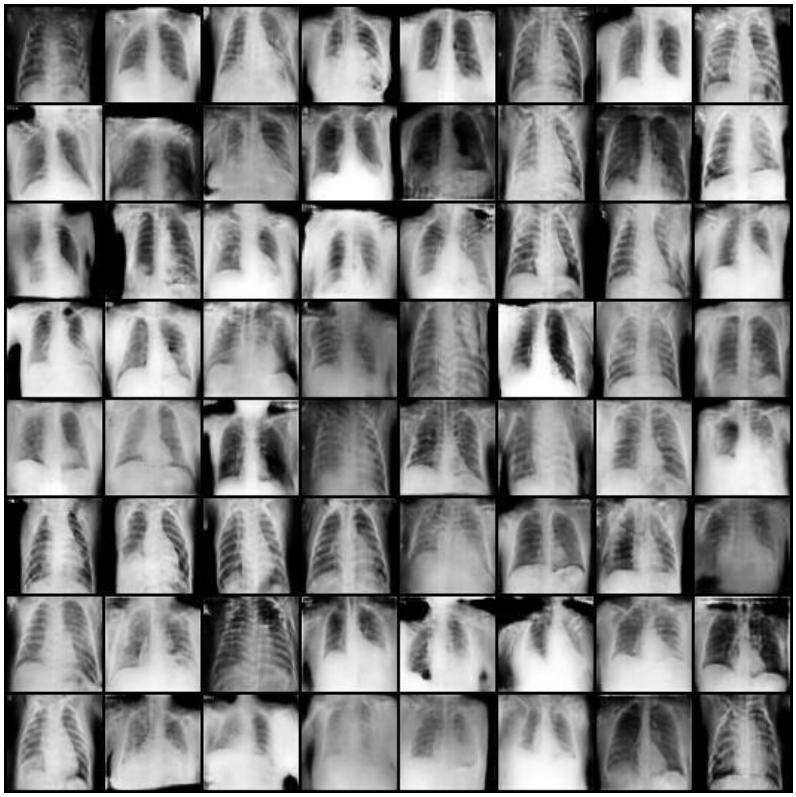


Figure 56 Final epoch generated image for WGAN

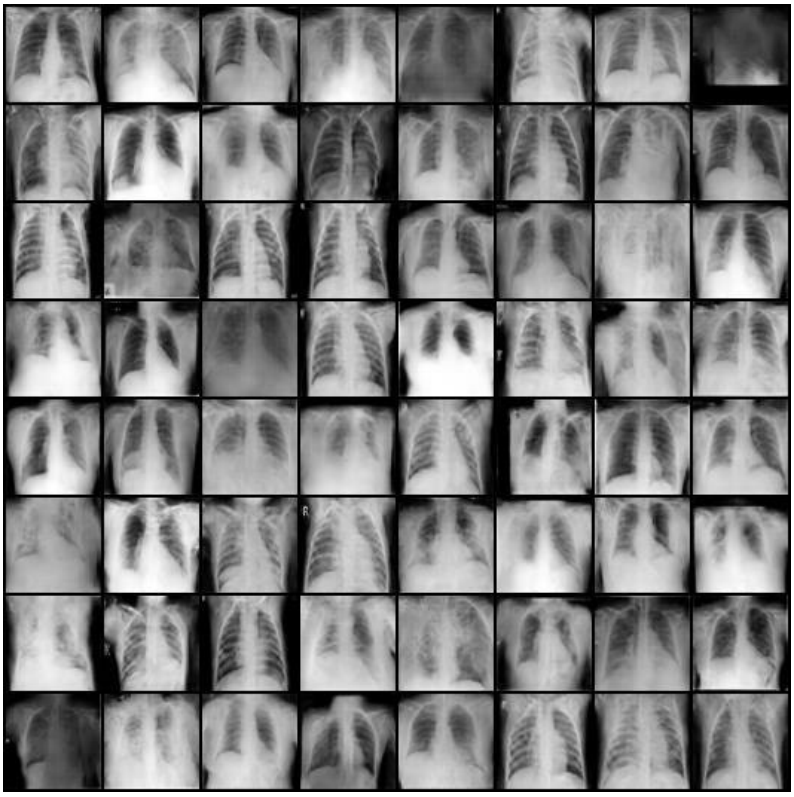


Figure 57 Final epoch generated image for WGAN-GP

5.5 Model performance

The performance of each CNN model on the test dataset is explained in this section. The test dataset has 400 samples for each class: Normal, Covid, and Viral Pneumonia. Additionally, the FID score for each GAN model is also measured.

5.5.1 Confusion Matrix

Confusion matrix is a performance measurement table that is used to analyze the performance of a classifier with respect to individual classes. A confusion matrix for a multiclass classification is given below.

		Predicted		
		Covid	Normal	Viral Pneumonia
Actual	Covid	a	b	c
	Normal	d	e	f
	Pneumonia	g	h	i

Table 6 Confusion matrix

Since this is a multiclass classification, the calculation of false positives and negatives differs from a confusion matrix from a binary classification. For each class in the confusion matrix above, we have 4 outcomes: **True positive, False positive, True negative** and **False negative**. False positive and negative are also often called as **type I** and **II** errors.

For class **Covid**, let TP, TN, FP, and FN be true positive, true negative, false positive and false negative respectively. TP means that the model predicted positive class (covid) and it is a correct prediction (covid). TN means that the model predicted not positive (not covid) and it is a correct prediction (not covid). FP means that the model predicted positive class (covid) but it is an incorrect prediction (not covid). FN means that the model predicted negative class (not covid) and it is an incorrect prediction (covid). Using these techniques, lets calculate the corresponding values for covid class from the confusion matrix.

	True positive	True Negative	False positive	False negative
Covid	a	e + f + h + i	d + g	b + c
Normal	e	a + c + g + i	b + h	d + f
Viral Pneumonia	i	a + b + d + e	c + f	g + h

Table 7 calculating positives and negatives using confusion matrix

Now that we have the positives and negatives, let's look at different performance metrics used to compare models' performance in classification.

5.5.1.1 Precision

Precision tells us among all the positive predictions made by the model, how many of them are correct or positive. For example, the precision of the model for covid class is 92.5 % means that when our model predicts covid class for an input x-ray, it is correct 92.5 % of the time.

Precision = **TP / (TP + FP)**

5.5.1.2 Recall or Sensitivity or True positive rate

Recall tells us among all the positive targets, how many of the targets our model successfully classified as positive. For example, the recall of the model for covid class is 92.5 % means it correctly identifies 92.5 % of covid x-rays.

Recall = **TP / (TP + FN)**

5.5.1.3 F1 score

F1 score is the harmonic mean or weighted average of precision and recall. F1 score helps to interpret the balanced ability of a classification model to not only capture positive cases but also be accurate with the positive cases it captures (precision). For example, the F1 Score of our model is 0.925 means that the classifier's ability to capture positive instances and be accurate with the instances it captures is 0.925.

F1 score = **2 * precision * recall / (recall + precision)**

5.5.1.4 Specificity or True negative rate

Specificity or true negative rate explains that, when the actual value is negative, how often is the prediction made by our model correct. For example, the specificity is almost 96.3 indicates that if our model makes a negative class prediction, it is correct 96.3 % of the times.

Specificity/TPR = **TN / (TN + FP)**

5.5.1.5 False positive rate

When an actual value is negative, how often is the prediction made by our model incorrect is given by the False Positive Rate. Here, the false positive rate of the model for class covid is roughly 0.04 means that if a model makes a covid class prediction, but in fact the target is not class covid, the model is incorrect roughly 4 % of the times.

False positive rate = **FP / (FP + TN)** or **1 – TNR (specificity)**

5.5.1.6 False negative rate

When an actual value is positive, how often the prediction made by our model is incorrect is given by FNR. Here, the false negative rate of the model for class covid is roughly 0.075 which means that if a model makes a not covid class prediction, but in fact the target is class covid, the model is incorrect roughly 7.5 % of the time.

False negative rate = **FN / (FN + TP)**

5.5.1.6 Accuracy

Accuracy simply refers to the number of correct predictions out of all possible predictions. If the accuracy of a model is 80 %, it means that it was able to classify 80 % of the instances correctly. It can be given as:

Accuracy = **TP + TN / (TP + TN + FP + FN)**

5.5.1 EfficientNet performance on test data

The confusion matrix of EfficientNet model created using matplotlib library is given below.

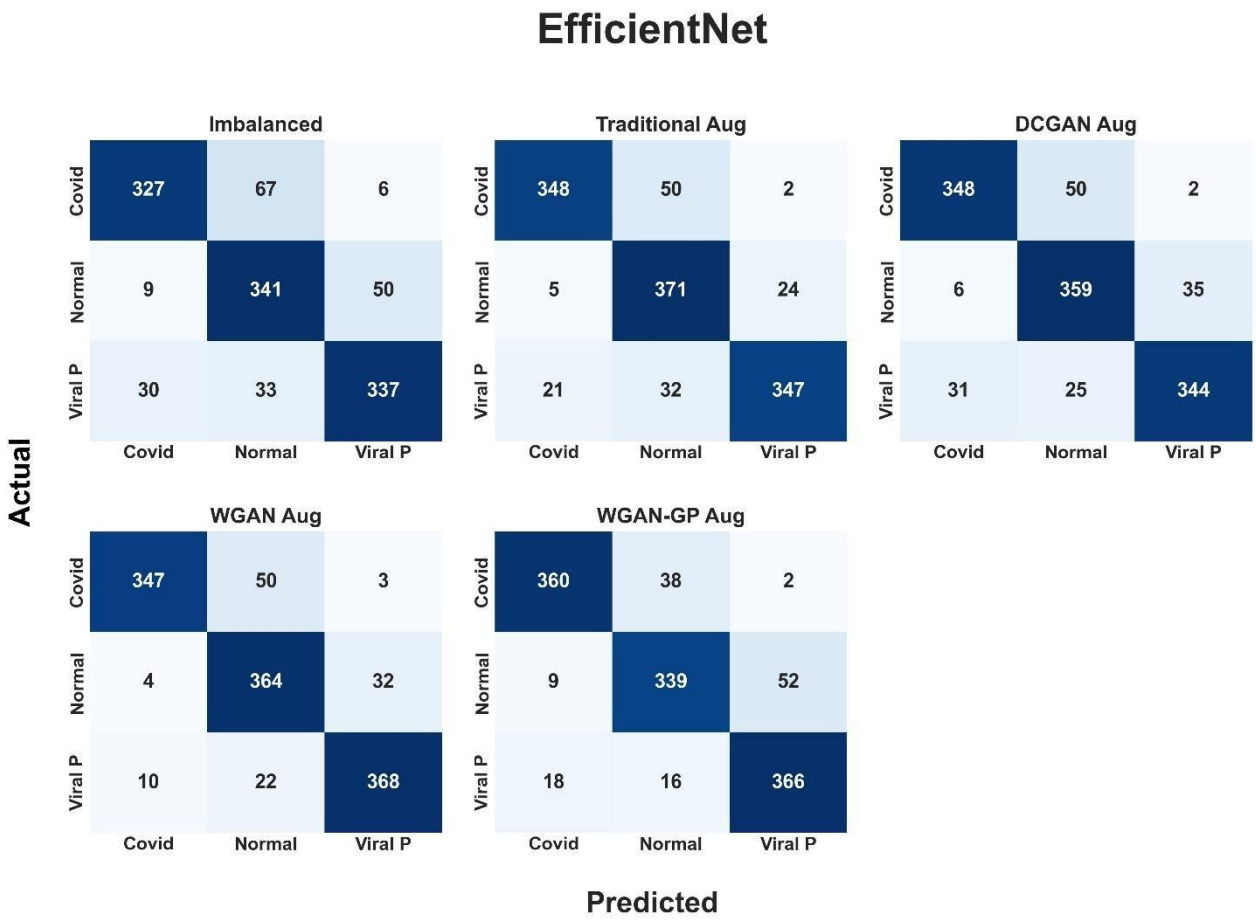


Figure 58 EfficientNet confusion matrix

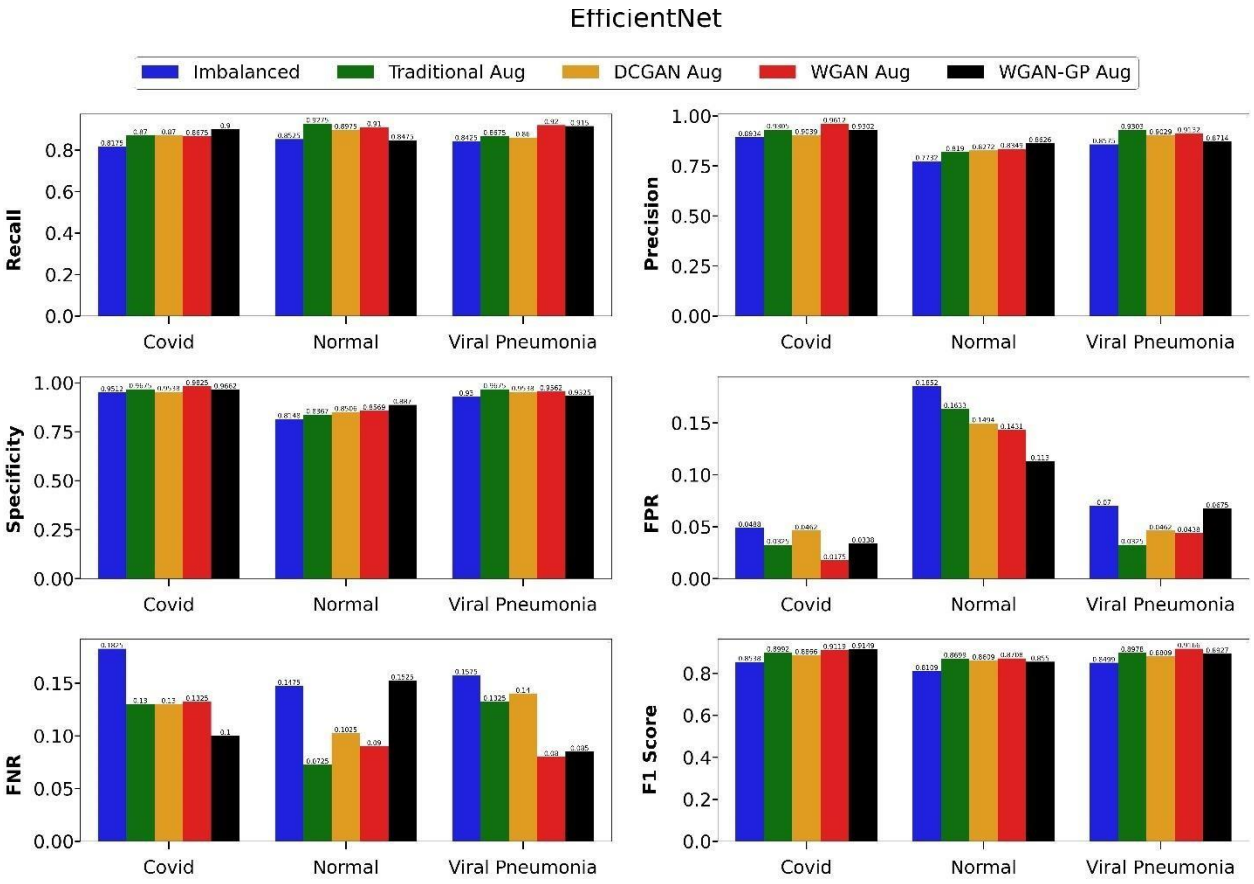


Figure 59 EfficientNet performance metrics bar diagram

From the bar diagram above, we can clearly see that the EfficientNet model trained on imbalanced data performs the worst across all metrics. Similarly, the model trained with traditional data augmentation outperforms DCGAN based data augmentation. WGAN and WGAN-GP have the best performance and their difference is very small. Especially for covid class, WGAN-GP has the best **recall** and **f1 score** of **0.9** and **0.91** respectively. The following table contains the accuracy of the model across each augmentation as compared to imbalanced data.

EfficientNet	**Test accuracy**
Imbalanced	83.75
Traditional augmentation	88.83
DCGAN based augmentation	87.58
WGAN based augmentation	**89.92**

WGAN-GP based augmentation	88.75

Figure 60 Accuracies of EfficientNet model

EfficientNet model trained with WGAN based data augmentation has the highest test accuracy of approximately **89.92** % respectively.

Class	Recall	Precision	F1 score
	Imbalanced data		
Covid	0.8175	0.8934	0.8538
Normal	0.8525	0.7732	0.8109
Viral Pneumonia	0.8425	0.8575	0.8499
	Traditional Augmentation		
Covid	0.87	0.9305	0.8992
Normal	**0.9275**	0.819	0.8699
Viral Pneumonia	0.8675	**0.9303**	0.8978
	DCGAN based Augmentation		
Covid	0.87	0.9039	0.8866
Normal	0.8975	0.8272	0.8609
Viral Pneumonia	0.86	0.9029	0.8809
	WGAN based Augmentation		
Covid	0.8675	**0.9612**	0.9119
Normal	0.91	0.8349	**0.8708**
Viral Pneumonia	**0.92**	0.9132	**0.9166**
	WGAN-GP based Augmentation		
Covid	**0.9**	0.9302	**0.9149**

Normal	0.8475	**0.8626**	0.855
Viral Pneumonia	0.915	0.8714	0.8927

Figure 61 EfficientNet's Precision, recall and f1 for each class

An ROC curve, also called receiver operating characteristics curve, is a graph in which false positive rate is plotted in x-axis and true positive rate is plotted in y-axis. This graph shows the performance of a classifier at all classification thresholds. AUC stands for area under the ROC curve. It means that the AUC score represents the ability of a classifier to predict positive and negative classes correctly. The AUC score of a model is 0.8 signifies that the probability that the model will assign larger probability to a random positive real number than a negative number is 0.8. The value of AUC ranges from 0 to 1 with 1 being the best model. Similarly, 0.5 AUC score signifies that a model is making random predictions. An AUC score of 0 means that the model is predicting 1 as 0, and 0 as 1. The less classification threshold means more positives being predicted by the model.

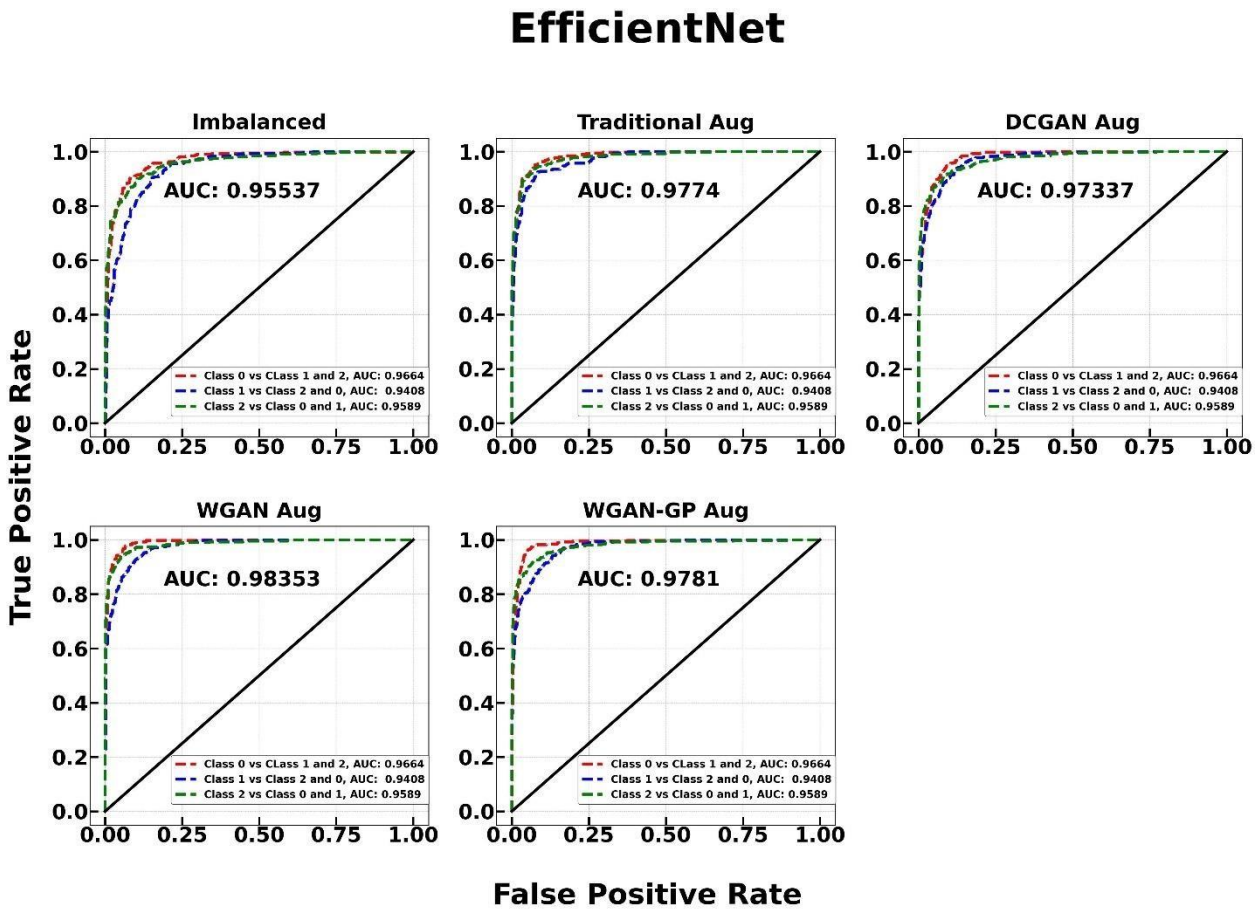


Figure 62 EfficientNet model's AUC ROC Curve

AUC score based on one versus rest of the class policy is used in the above plot. The mean score of AUC scores of all classes is calculated as final AUC score. The EfficientNet model trained with **WGAN** based data augmentation achieved highest AUC score of **0.9835** whereas EfficientNet classifier trained with imbalanced data achieved a minimum AUC score of 0.9553.

5.5.2 ResNet50 performance on test data

The confusion matrix of ResNet50 model created using matplotlib library is given below.

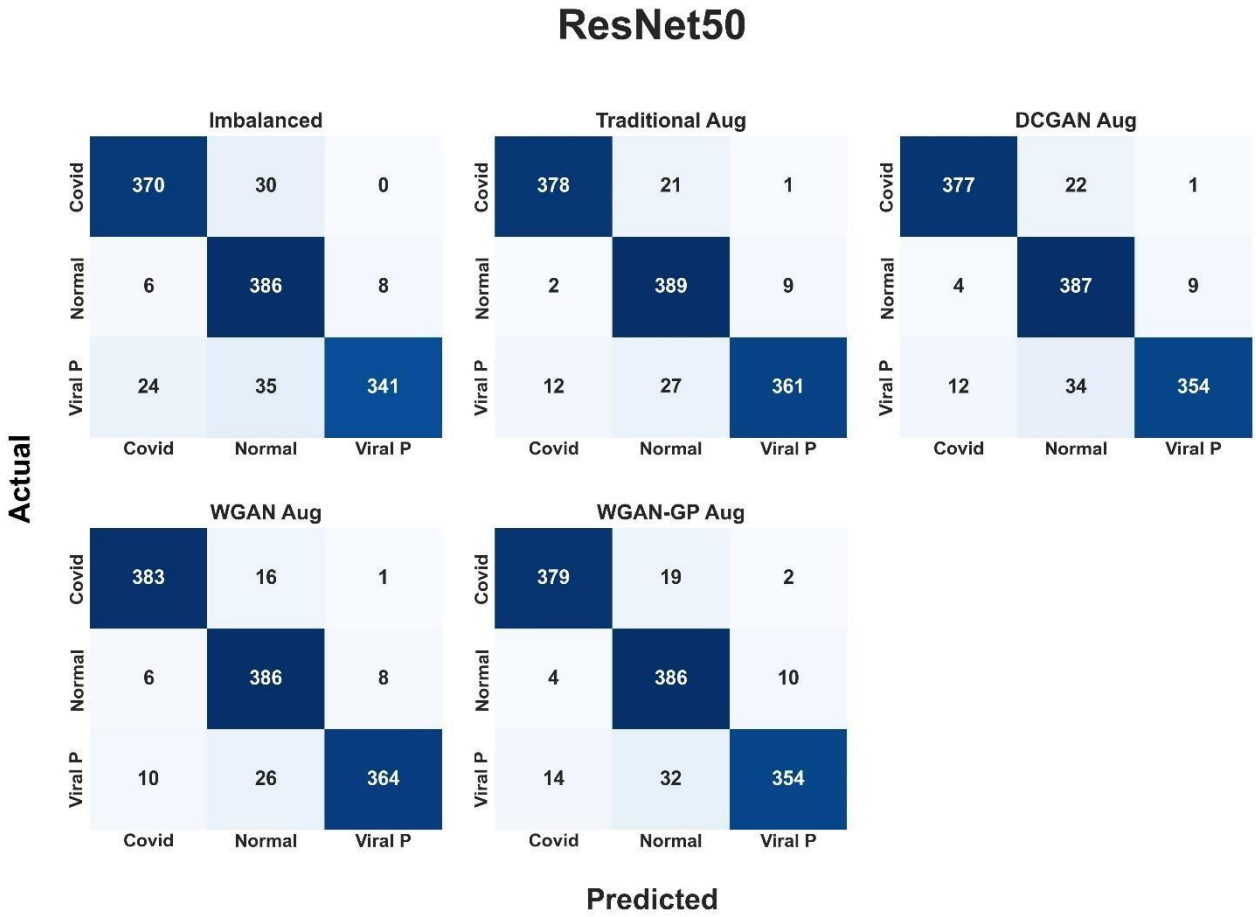


Figure 63 ResNet50 model's confusion matrix

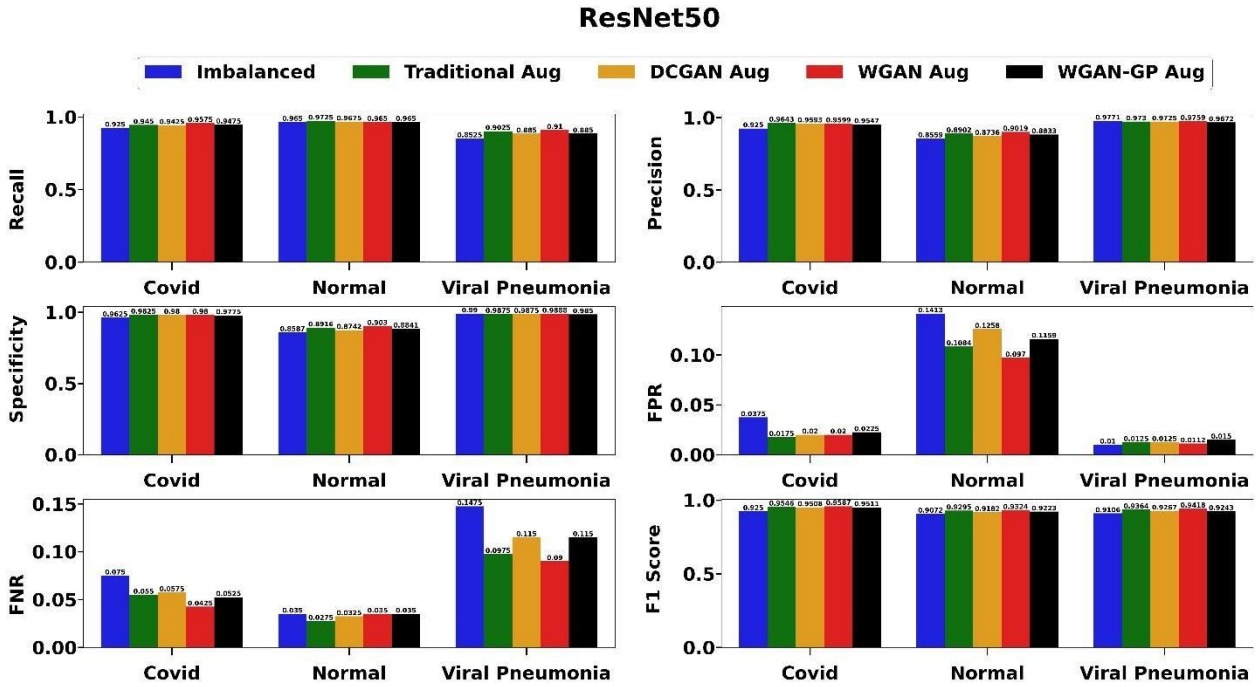


Figure 64 ResNet50 performance metrics bar diagram

From the bar diagram above, we can clearly see that the R+esNet50 model trained on imbalanced data performs the worst across all metrics. Similarly, the model trained with traditional data augmentation outperforms DCGAN based data augmentation. WGAN and WGAN-GP have the best performance and their difference is very small. Especially for covid class, WGAN has the best **recall** and **f1 score** of **0.9575** and **0.9587** respectively. The following table contains the accuracy of the model across each augmentation as compared to imbalanced data.

ResNet50	**Test accuracy**
Imbalanced	0.9142
Traditional augmentation	0.9400
DCGAN based augmentation	0.9317
WGAN based augmentation	**0.9442**
WGAN-GP based augmentation	0.9325

Table 8 Accuracies of ResNet50 model

ResNet50 model trained with WGAN based data augmentation has the highest test accuracy of approximately **94.42** % respectively.

Class	Recall	Precision	F1 score
	Imbalanced data		
Covid	0.9250	0.9250	0.9250
Normal	0.9650	0.8559	0.9072
Viral Pneumonia	0.8525	0.9771	0.9106
	Traditional Augmentation		
Covid	0.9450	0.9643	0.9546
Normal	0.9725	0.8902	0.9295
Viral Pneumonia	0.9025	0.9730	0.9364
	DCGAN based Augmentation		
Covid	0.9425	0.9593	0.9508
Normal	0.9675	0.8736	0.9182
Viral Pneumonia	0.8850	0.9725	0.9267
	WGAN based Augmentation		
Covid	0.9575	0.9599	0.9587
Normal	0.9650	0.9019	0.9324
Viral Pneumonia	0.9100	0.9759	0.9418
	WGAN-GP based Augmentation		
Covid	0.9475	0.9547	0.9511
Normal	0.9650	0.8833	0.9223
Viral Pneumonia	0.8850	0.9672	0.9243

Table 9 EfficientNet's Precision, recall and f1 for each class

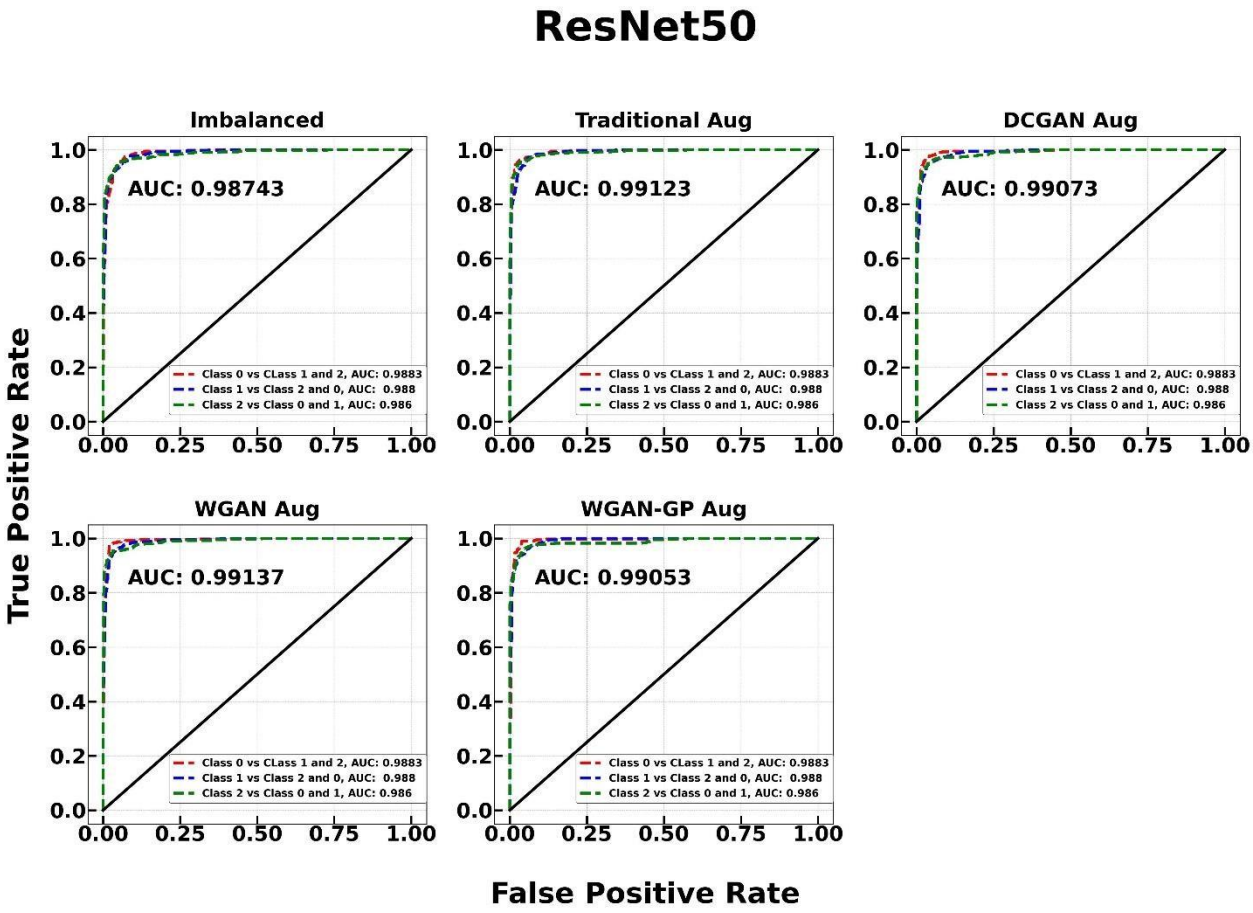


Figure 65 ResNet50 model's AUC ROC Curve

AUC score based on one versus rest of the class policy is used in the above plot. The mean score of AUC scores of all classes is calculated as final AUC score. ResNet50 model trained with **WGAN** based data augmentation achieved highest AUC score of **0.99137** whereas ResNet50 classifier trained with imbalanced data achieved a minimum AUC score of 0.98743.

5.5.3 GoogLeNet performance on test data

GoogLeNet is trained for two cases: with and without auxiliary classifiers. The confusion matrix of GoogLeNet model without auxiliary classifiers is created using matplotlib library is given below.

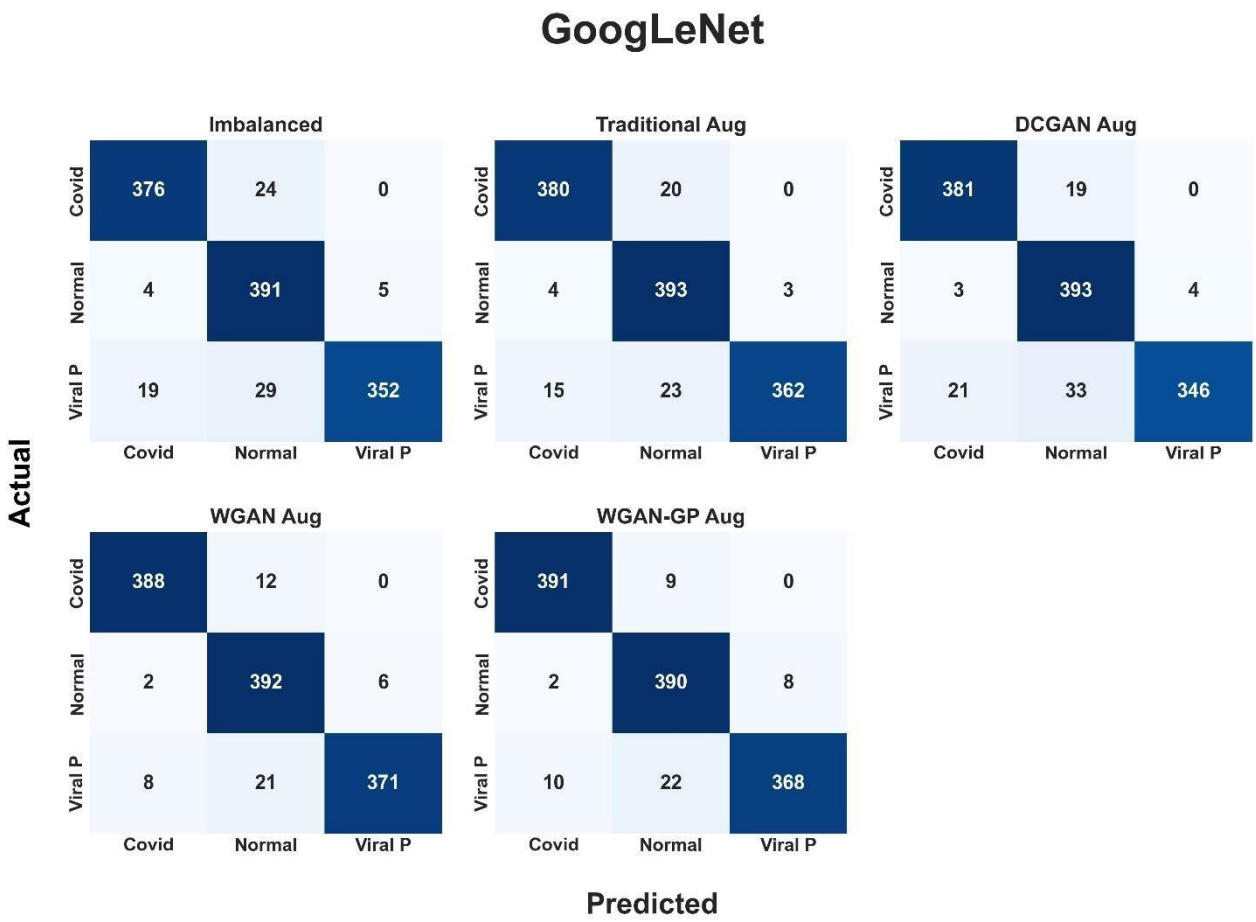


Figure 66 Confusion matrix for GoogLeNet without auxiliary classifier

GoogLeNet without Aux. classifier	Test accuracy
Imbalanced	0.9325
Traditional augmentation	0.9458
DCGAN based augmentation	0.9333
WGAN based augmentation	**0.9592**
WGAN-GP based augmentation	0.9575

Figure 67 Accuracies for GoogLeNet model without aux. classifiers

Class	Recall	Precision	F1 score
	Imbalanced data		
Covid	0.9400	0.9424	0.9412
Normal	0.9775	0.8806	0.9265
Viral Pneumonia	0.8800	0.9860	0.9300
	Traditional Augmentation		
Covid	0.9500	0.9524	0.9512
Normal	0.9825	0.9014	0.9402
Viral Pneumonia	0.9050	0.9918	0.9464
	DCGAN based Augmentation		
Covid	0.9525	0.9407	0.9466
Normal	0.9825	0.8831	0.9302
Viral Pneumonia	0.8650	0.9886	0.9227
	WGAN based Augmentation		
Covid	0.9700	0.9749	0.9724
Normal	0.9800	0.9224	0.9503
Viral Pneumonia	0.9275	0.9841	0.9550
	WGAN-GP based Augmentation		
Covid	0.9775	0.9702	0.9738
Normal	0.9750	0.9264	0.9501
Viral Pneumonia	0.9200	0.9787	0.9484

Table 10 GoogLeNet's without Aux. classifier Precision, recall and f1 for each class

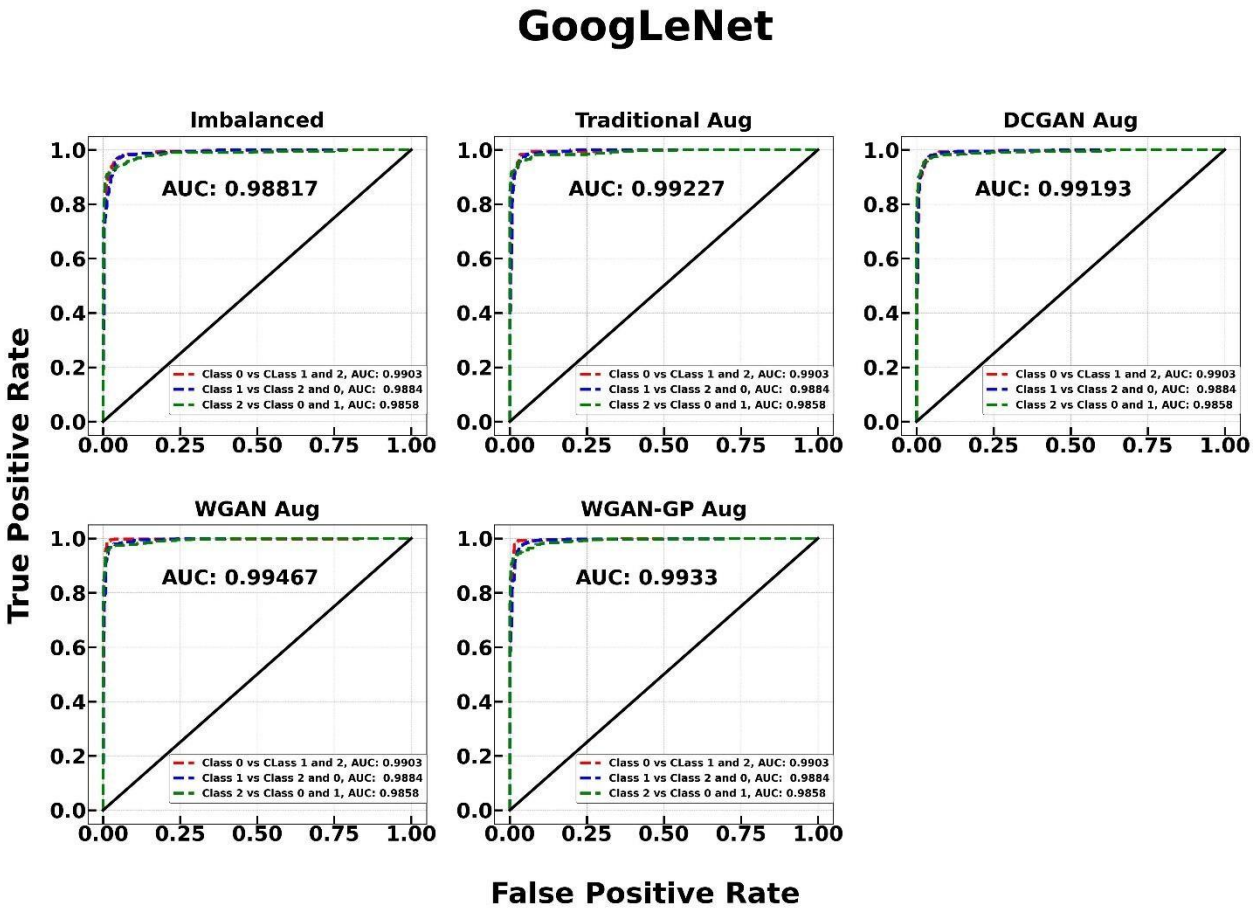


Figure 68 GoogLeNet model without Aux. classifier AUC ROC Curve

AUC score based on one versus rest of the class policy is used in the above plot. The mean score of AUC scores of all classes is calculated as final AUC score. GoogLeNet model trained with **WGAN** based data augmentation achieved highest AUC score of **0.99467** whereas GoogLeNet classifier trained with imbalanced data achieved a minimum AUC score of 0.98817.

The confusion matrix of GoogLeNet model with auxiliary classifiers is created using matplotlib library is given below.

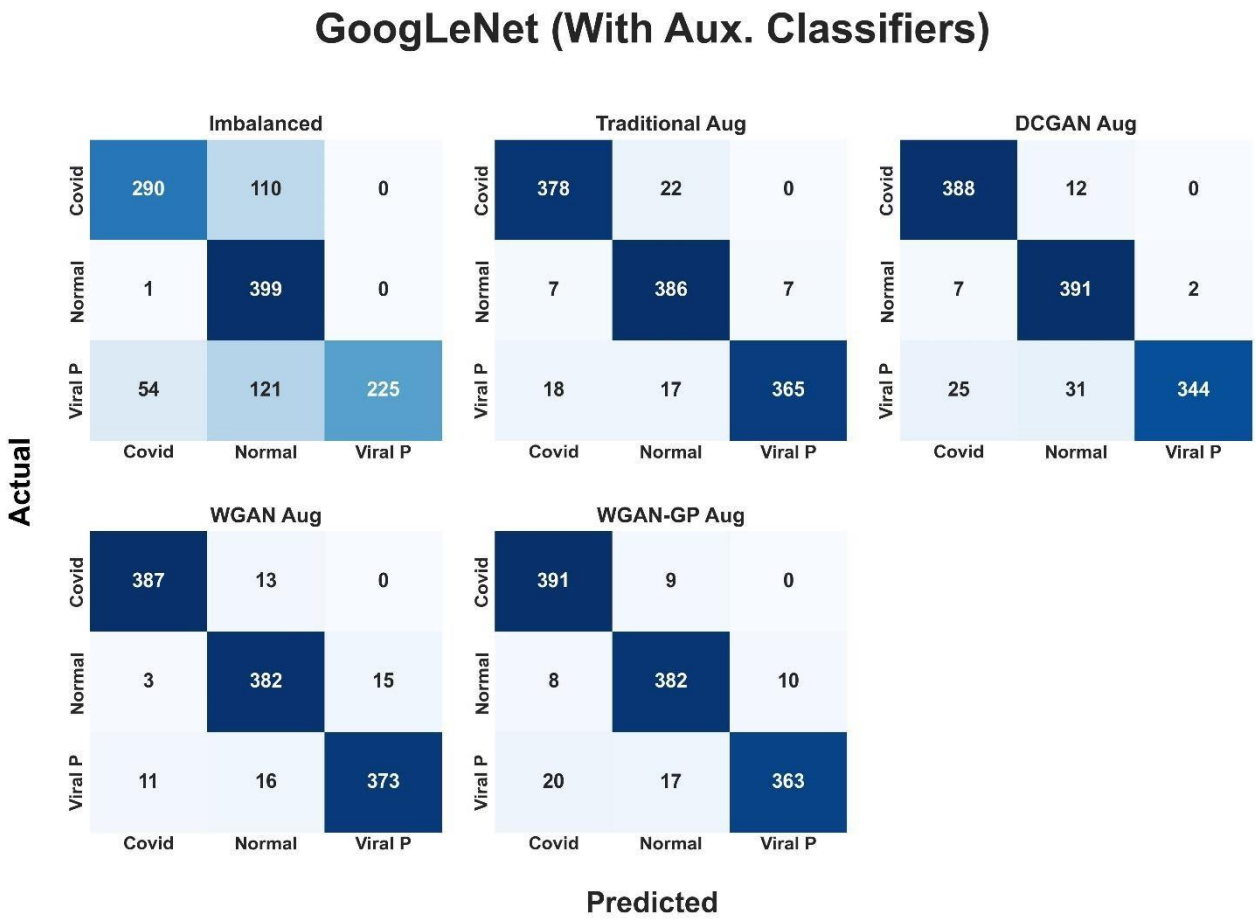


Figure 69 GoogLeNet model with auxiliary classifier confusion matrix

GoogLeNet with Aux. classifier	Test accuracy
Imbalanced	0.7617
Traditional augmentation	0.9408
DCGAN based augmentation	0.9358
WGAN based augmentation	0.9517
WGAN-GP based augmentation	0.9467

Table 11 Accuracies for GoogLeNet model with aux. classifiers

Class	Recall	Precision	F1 score
	Imbalanced data		
Covid	0.7250	0.8406	0.7785
Normal	0.9975	0.6333	0.7747
Viral Pneumonia	0.5625	1.0000	0.7200
	Traditional Augmentation		
Covid	0.9450	0.9380	0.9415
Normal	0.9650	0.9082	0.9357
Viral Pneumonia	0.9125	0.9812	0.9456
	DCGAN based Augmentation		
Covid	0.9700	0.9238	0.9463
Normal	0.9775	0.9009	0.9376
Viral Pneumonia	0.8600	0.9942	0.9222
	WGAN based Augmentation		
Covid	0.9675	0.9651	0.9663
Normal	0.9550	0.9294	0.9420
Viral Pneumonia	0.9325	0.9613	0.9467
	WGAN-GP based Augmentation		
Covid	0.9775	0.9332	0.9548
Normal	0.9550	0.9363	0.9456
Viral Pneumonia	0.9075	0.9732	0.9392

Table 12 GoogLeNet with Aux. classifier Precision, recall and f1 for each class

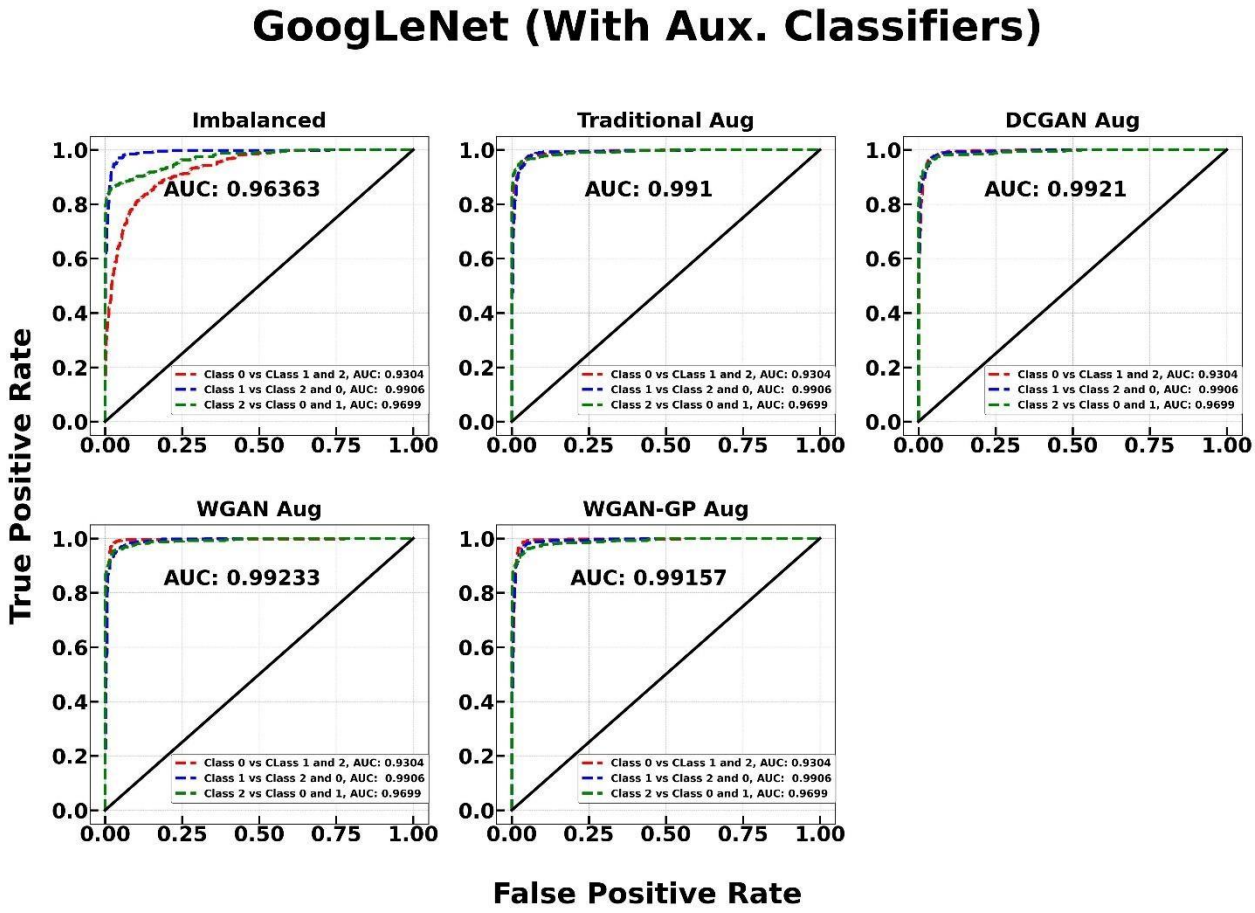


Figure 70 AUC ROC Curve for GoogLeNet model with auxiliary classifiers

AUC score based on one versus rest of the class policy is used in the above plot. The mean score of AUC scores of all classes is calculated as final AUC score. GoogLeNet model trained with **WGAN** based data augmentation achieved highest AUC score of **0.99233** whereas ResNet50 classifier trained with imbalanced data achieved a minimum AUC score of 0.96363.

5.5.4 Fréchet Inception Distance (FID) for GANs

FID is a metric used to evaluate the diversity and quality of images generated by a GAN architecture. It measures the feature distance between real and generated images, and a lower distance indicates better performance of the GAN model. To calculate FID, real and fake images are passed into an InceptionV3 model, and the features extracted from the InceptionV3 model's intermediate activation layer are used to determine the distance

between the probability distributions of real and fake images (Thakur, 2022). The bar plot below showcases the FID score for individual classes in the training dataset.

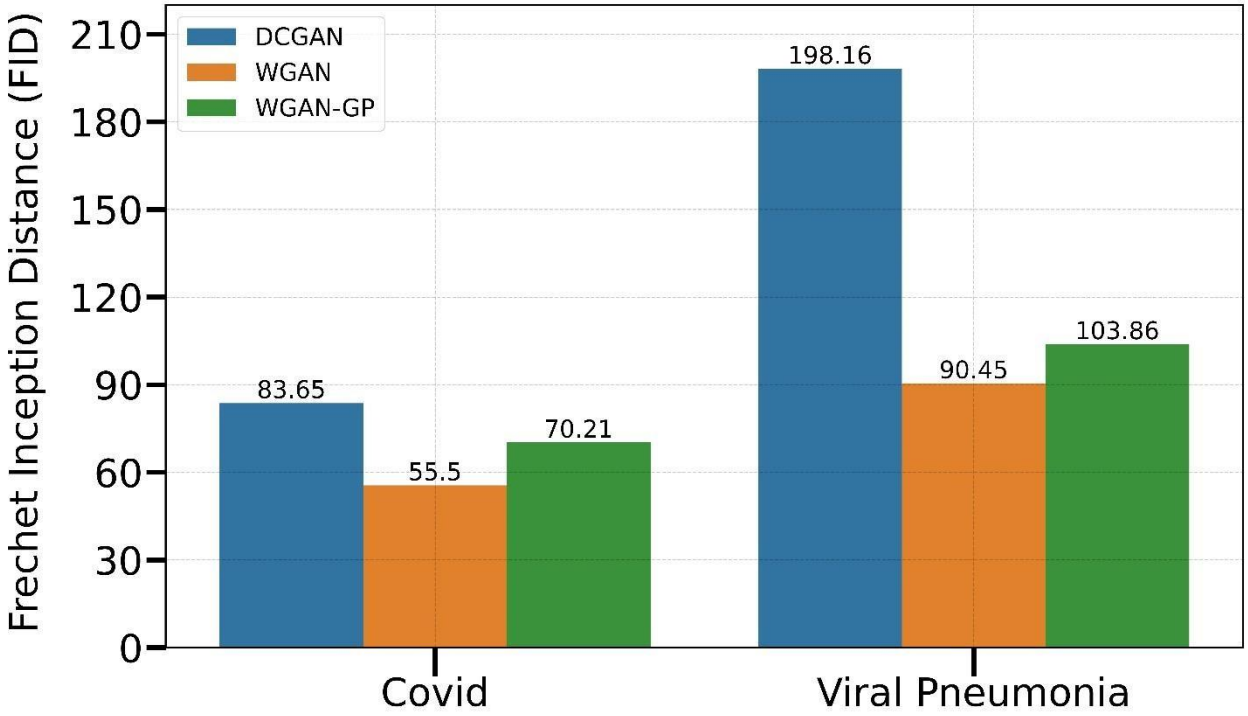


Figure 71 FID Score for GANs

5.6 AI Integration in web application

The best-performing model is selected and deployed in the web application. When a user submits a chest X-ray using the form available on the webpage, the post method in the form collects the image and passes it to the home function in views.py. When the user clicks the submit button, the Model class is instantiated, and the uploaded image is passed to the class. The predict method in the class converts the image into an array and performs necessary transformations before feeding it into the model. The output of the model is predicted probability for each class, and the probabilities are passed to the HTML file using the Jinja web template. The following screenshot contains the code to make predictions in the web application.

```python
class Model:
    def __init__(self):

        with open('GoogLeNet_With_WGANGP.pkl', 'rb') as f:
            self.model = pickle.load(f)

        mean = 0.5199
        std = 0.2488
        image_size = 64
        CHANNELS_IMG = 1
        stats = [mean for _ in range(CHANNELS_IMG)], [std for _ in range(CHANNELS_IMG)]


        self.transformations_to_perform = transform=tt.Compose([
                                    tt.Grayscale(num_output_channels=1),
                                tt.Resize((image_size, image_size)),
                                tt.ToTensor(),
                                tt.Normalize(*stats)])

        self.classes = ['Covid', 'Normal', 'Viral Pneumonia']

    def predict(self, img_path):
        img = self.transformations_to_perform(Image.open(img_path))
        grey_image = np.zeros([1, 224, 224], dtype = np.float64)
        grey_image[:,80:144, 80:144] = img
        grey_image = torch.from_numpy(grey_image)
        grey_image = grey_image.type(torch.FloatTensor)
        grey_image = grey_image.reshape((1, 1, 224, 224))

        a =  self.model(grey_image)
        out = nn.Softmax(dim = 1)(a)
        out = out.detach().numpy()[0]


        outputs = {'Covid': round(out[0] * 100, 5),
            'Normal': round(out[1] * 100, 5),
            'Viral Pneumonia': round(out[2] * 100, 5)}
```

Figure 72 Web app prediction code 1

```
        grey_image = grey_image.reshape((1, 1, 224, 224))

        a =  self.model(grey_image)
        out = nn.Softmax(dim = 1)(a)
        out = out.detach().numpy()[0]


        outputs = {'Covid': round(out[0] * 100, 5),
            'Normal': round(out[1] * 100, 5),
            'Viral_Pneumonia': round(out[2] * 100, 5)}

        selected_class = np.argmax(out)
        print("OUT: ", out)
        print(selected_class)

        description = ""
        symptoms = ""
        solutions = ""

        if selected_class in [0]:
            mycursor.execute("SELECT * FROM data where Class = {}".format(selected_class))
            myresult = list(mycursor.fetchall()[0])

            description = myresult[1]
            symptoms = myresult[2]
            solutions = myresult[3]

            description = description.replace(",,", "<br>")
            # description =  "<p>{}</p>".format(description)
            symptoms = symptoms.replace(",,", "<br>")
            # symptoms =  "<p>{}</p>".format(symptoms)
            solutions = solutions.replace(",,", "<br>")
            # solutions =  "<p>{}</p>".format(solutions)

            f = open('test.txt', 'w')
            f.write("Case: "+self.classes[selected_class]+"\n")
            f.write("Description: "+ description+"\n\n")
            f.write("Symptoms: "+ symptoms+"\n\n")
            f.write("Solutions: "+ solutions+"\n\n")
            f.write("Confidence: "+ str(out[selected_class]))
            f.close()
            # return all_results

        all_results = [outputs, description, symptoms, solutions]
        return all_results
```

Figure 73 Web application prediction code 2

6. Conclusion

MedicareAI has been developed with the main purpose of automating and enhancing covid-19 diagnosis using deep learning and the power of GANs. The recent pandemic crippled Nepal's healthcare system since there were more infections than testing kits could handle. In this case, this project can reduce the timeframe required for testing a symptomatic person by simply uploading a chest x-ray and making predictions in a span of less than **30 seconds**. To ensure highly accurate and precise predictions, three CNN architectures, ResNet50, GoogLeNet, and EfficientNet, were trained and their performance on the test dataset was measured. The GoogLeNet model without auxiliary classifiers, trained with WGAN and WGAN-GP-based augmentation, outperformed every other form of augmentation including imbalanced data. The GoogLeNet model trained on WGAN-based data augmentation was integrated into a web application that can be used by doctors to classify whether a symptomatic person has covid or not.

7. Critical Evaluation of the Project

7.1 Final report

Due to the word limit of 10000 words, I was unable to express the mathematical intuition of the machine learning algorithms effectively. Additionally, there are so many subheadings that confused me time and again while writing the report. I was able to explain GANs architecture and their working mechanism quite effectively in the literature review section. The writing is not perfect due to my limited knowledge of the English language. Overall, the report is very good with slight room for improvement.

7.2 Findings and process

The results of the project were predictable to an extent as GAN-based data augmentation outperformed traditional augmentation. I was very familiar with the PyTorch framework, which made it very easy to research and write code to train GANs and CNN architectures. The web app development was tedious as I was not familiar with web development at all. The web application of this project is very simplistic which indicates my limitations and weakness in web development. But the ML section has very strong results, as CNN architectures trained on techniques as proposed in the project proposal have performed very well. The project implements everything as proposed in the project proposal.

7.3 Self-reflection

This project enhanced my understanding of generative models in the field of deep learning. My skills using NumPy, Matplotlib, and PyTorch were also enhanced. Similarly. I also learned the basics of HTML, CSS, and JavaScript. This project also taught me about agile methodology and scrum framework to be more specific. This project also taught me to use various tools and technologies like Git, GitHub, and Virtual Studio Code.

7.4 System

Developing the front-end and back-end was very tough for me due to my limited understanding of web-based applications. Learning Django for deploying my ML model was very tough for me. I had to explore and research many videos and articles relating to ML model deployment using Django to complete my project. Integrating MySQL database was easy due to my familiarity with MySQL.

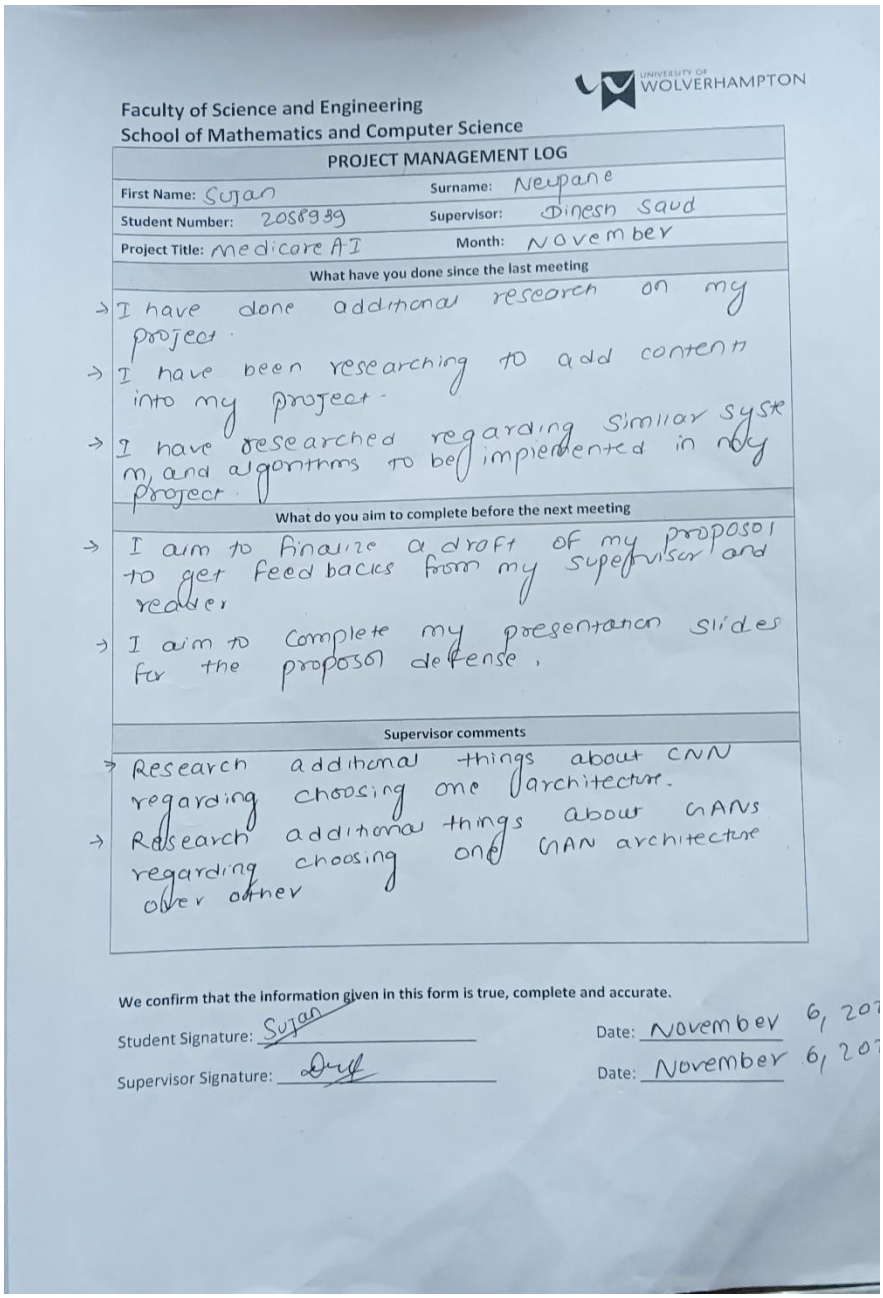
7.5 Future Escalation

As mentioned in the scope and limitations system, the GANs implemented in this project are only conditional but not controllable. In the future work, more powerful controllable GANs like StyleGAN can be implemented to generate high quality larger sized chest x-ray images. Furthermore, the results showcased that GoogLeNet model could had the highest accuracy among all other models. Since the GoogLeNet model used was in 2014, latest inception models can also be used to check if higher accuracy can be achieved. Finally, transfer learning could also be implemented to achieve higher accuracy.

8. Evidence of Project Management

8.1 Log Sheet

8.1.1 Supervisor Meeting held on Nov 6, 2022



UNIVERSITY OF WOLVERHAMPTON

Faculty of Science and Engineering
School of Mathematics and Computer Science

PROJECT MANAGEMENT LOG

First Name: Sujan Surname: Neupane
Student Number: 2058939 Supervisor: Dinesh Saud
Project Title: Medicare AI Month: November

What have you done since the last meeting

→ I have done additional research on my project.
→ I have been researching to add content into my project.
→ I have researched regarding similar system, and algorithms to be implemented in my project.

What do you aim to complete before the next meeting

→ I aim to finalize a draft of my proposal to get feedbacks from my supervisor and reader.
→ I aim to complete my presentation slides for the proposal defense.

Supervisor comments

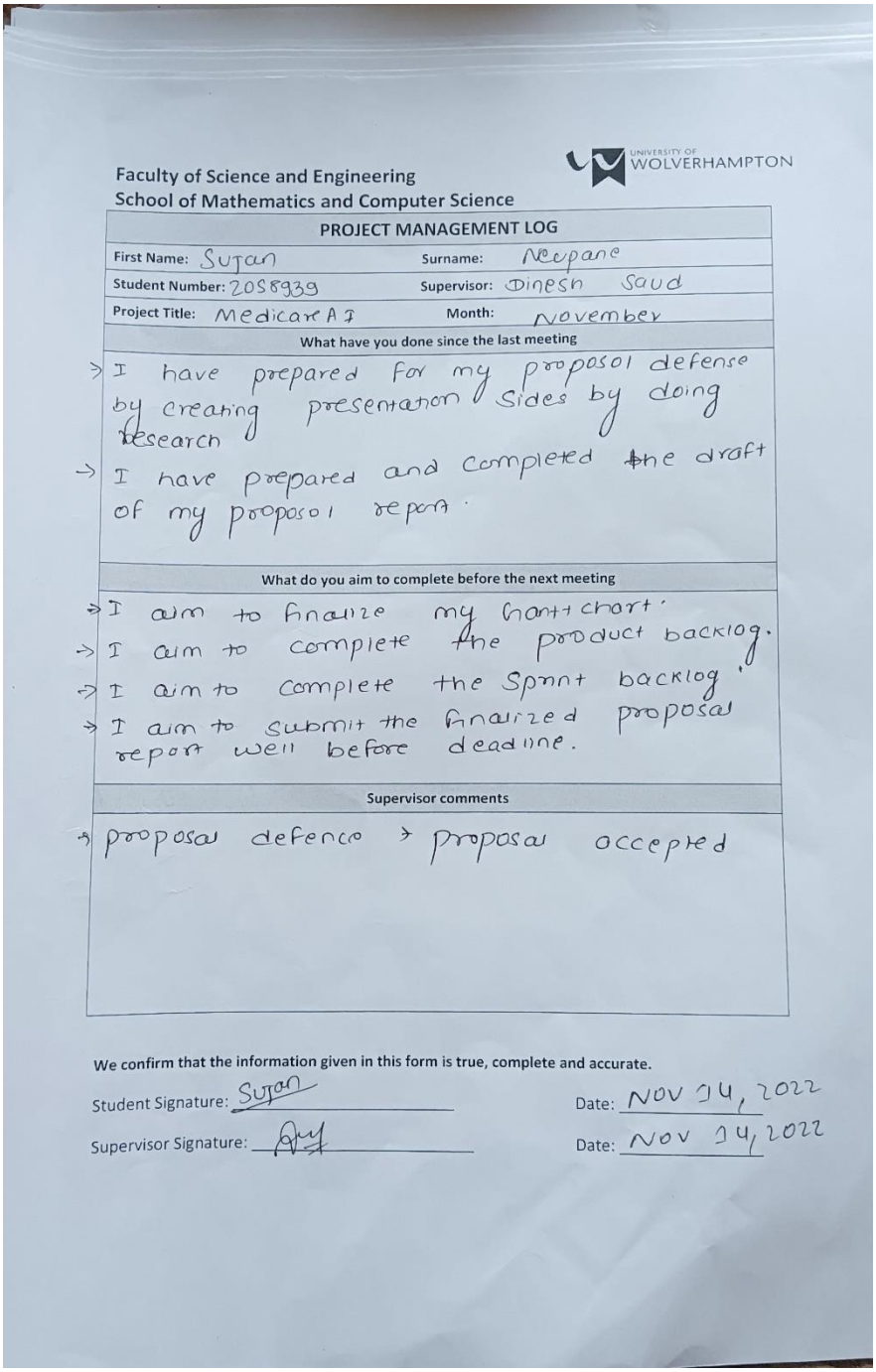
→ Research additional things about CNN regarding choosing one architecture.
→ Research additional things about GANs regarding choosing one GAN architecture over other

We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: November 6, 20
Supervisor Signature: Dinesh Date: November 6, 20

Figure 74 Supervisor Meeting held on Nov 6, 2022

8.1.2 Supervisor Meeting held on Nov 14, 2022



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name: Sujan Surname: Neupane
Student Number: 2058939 Supervisor: Dinesh Saud
Project Title: MedicareAI Month: November

What have you done since the last meeting

→ I have prepared for my proposal defense by creating presentation slides by doing research
→ I have prepared and completed the draft of my proposal report.

What do you aim to complete before the next meeting

→ I aim to finalize my Gantt chart.
→ I aim to complete the product backlog.
→ I aim to complete the Sprint backlog.
→ I aim to submit the finalized proposal report well before deadline.

Supervisor comments

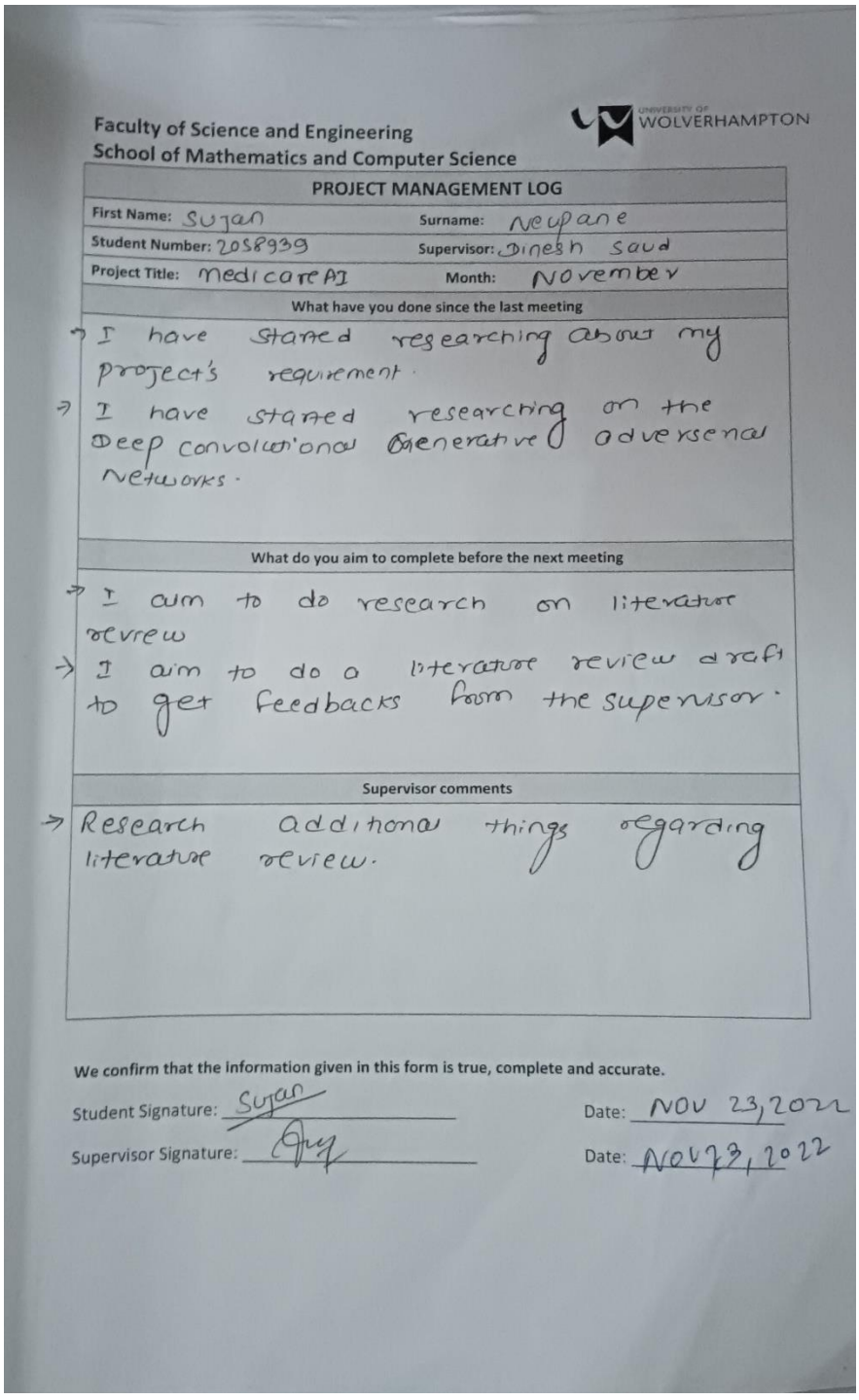
→ proposal defence → proposal accepted

We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: NOV 14, 2022
Supervisor Signature: Dinesh Date: NOV 14, 2022

Figure 75 Supervisor Meeting held on Nov 14, 2022

8.1.3 Supervisor Meeting held on Nov 23, 2022



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name: Sujan	Surname: Neupane
Student Number: 2058939	Supervisor: Dinesh Saud
Project Title: MediCare AI	Month: November

What have you done since the last meeting

- I have stated researching about my project's requirement.
- I have stated researching on the Deep convolutional Generative adversenal Networks.

What do you aim to complete before the next meeting

- I aim to do research on literature review
- I aim to do a literature review draft to get feedbacks from the supervisor.

Supervisor comments

- Research additional things regarding literature review.

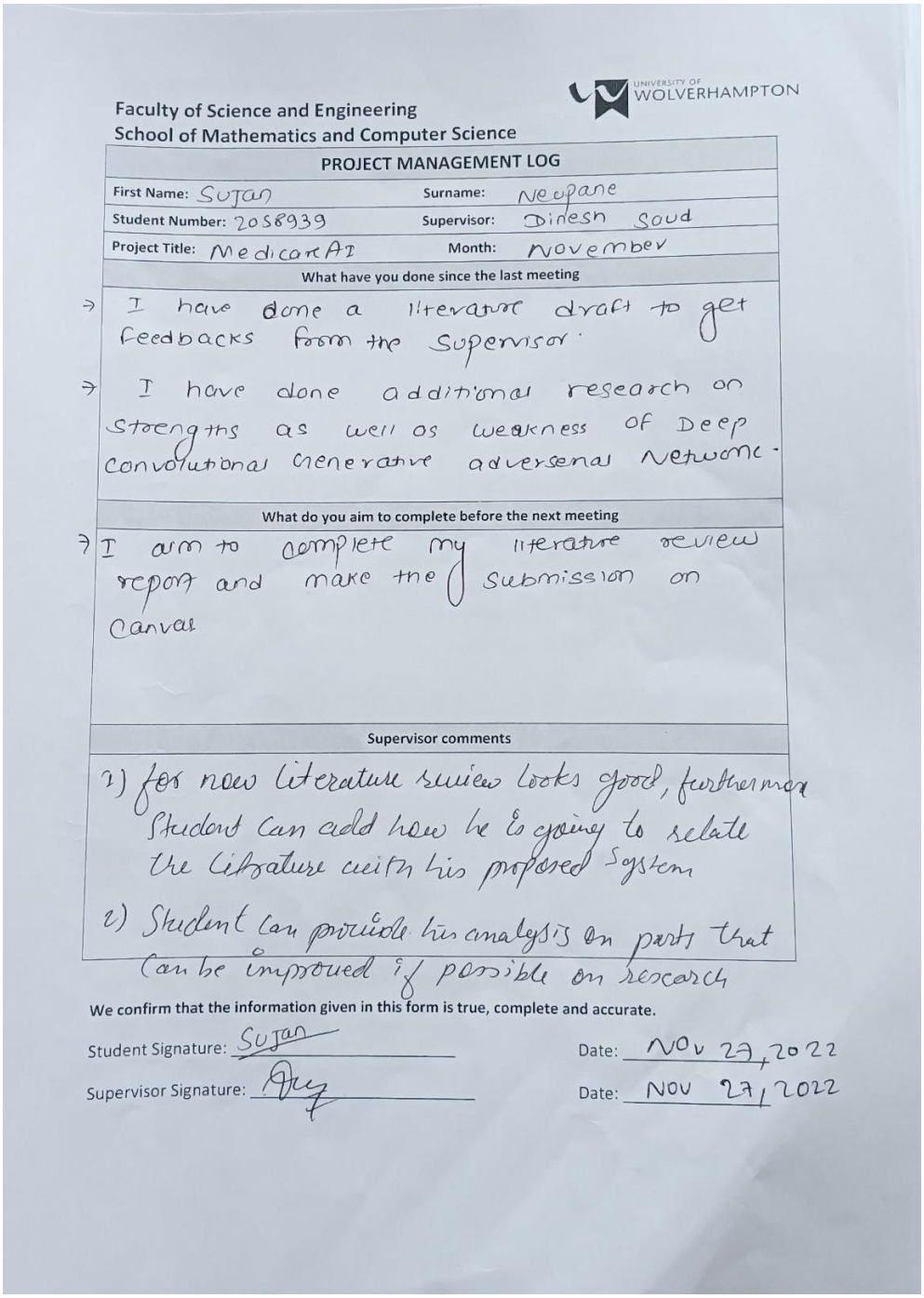
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: NOV 23, 2022

Supervisor Signature: Date: NOV 23, 2022

Figure 76 Supervisor Meeting held on Nov 23, 2022

8.1.4 Supervisor Meeting held on Nov 27, 2022



UNIVERSITY OF WOLVERHAMPTON

Faculty of Science and Engineering
School of Mathematics and Computer Science

PROJECT MANAGEMENT LOG

First Name: Sujan	Surname: Neupane
Student Number: 2058939	Supervisor: Dinesh Soud
Project Title: MedicarAI	Month: November

What have you done since the last meeting

→ I have done a literature draft to get feedbacks from the supervisor.

→ I have done additional research on strengths as well as weakness of Deep convolutional generative adversenal Networc.

What do you aim to complete before the next meeting

→ I aim to complete my literature review report and make the submission on canvas

Supervisor comments

1) for now literature review looks good, furthermore Student can add how he is going to relate the literature with his proposed system

2) Student can provide his analysis on parts that can be improved if possible on research

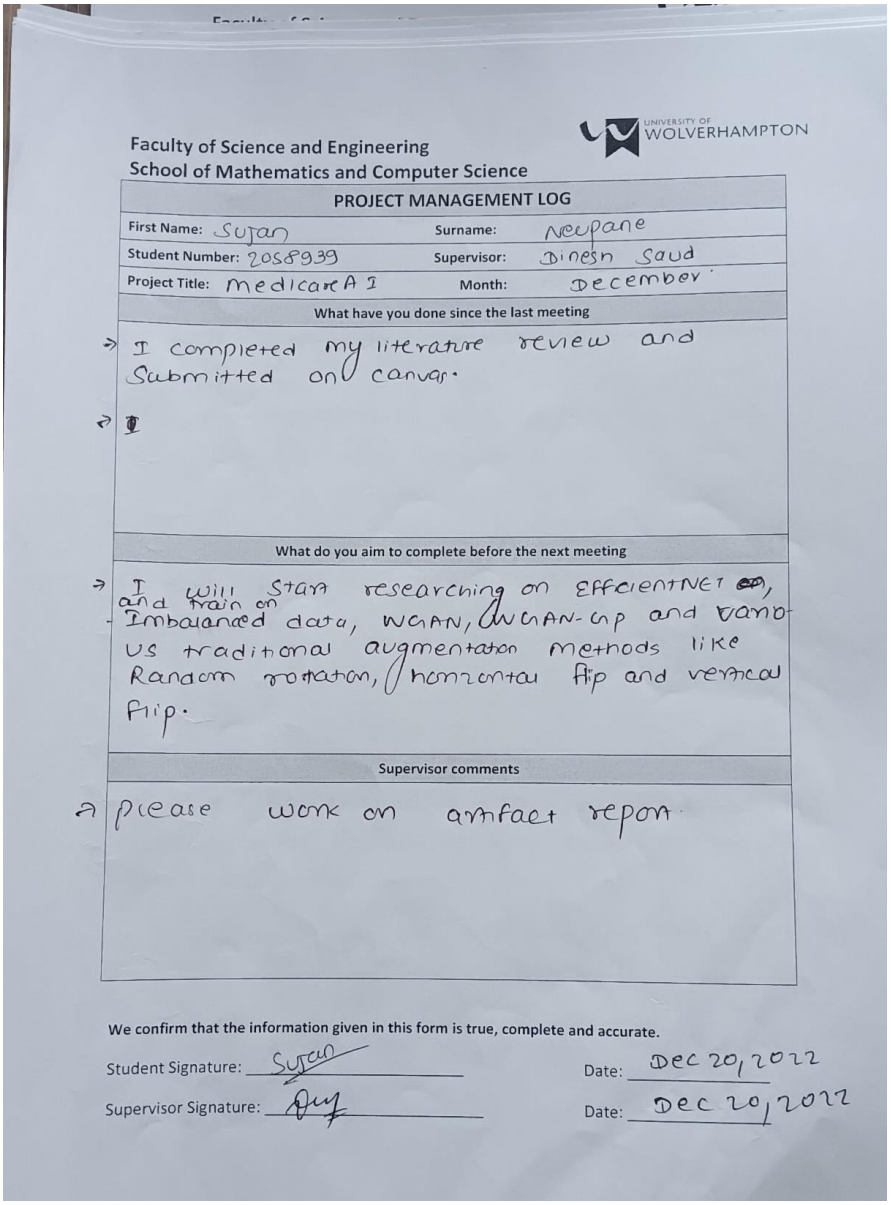
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: Nov 27, 2022

Supervisor Signature: Dinesh Date: Nov 27, 2022

Figure 77 Supervisor Meeting held on Nov 27, 2022

8.1.5 Supervisor Meeting held on Dec 20, 2022



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name:	Sujan	Surname:	Neupane
Student Number:	2058939	Supervisor:	Dinesh Saud
Project Title:	MedicareA I	Month:	December

What have you done since the last meeting

- I completed my literature review and submitted on canvas.

What do you aim to complete before the next meeting

- I will start researching on EfficientNet, and train on Imbalanced data, WGAN, WGAN-GP and various traditional augmentation methods like Random rotation, horizontal flip and vertical flip.

Supervisor comments

- please work on artifact report

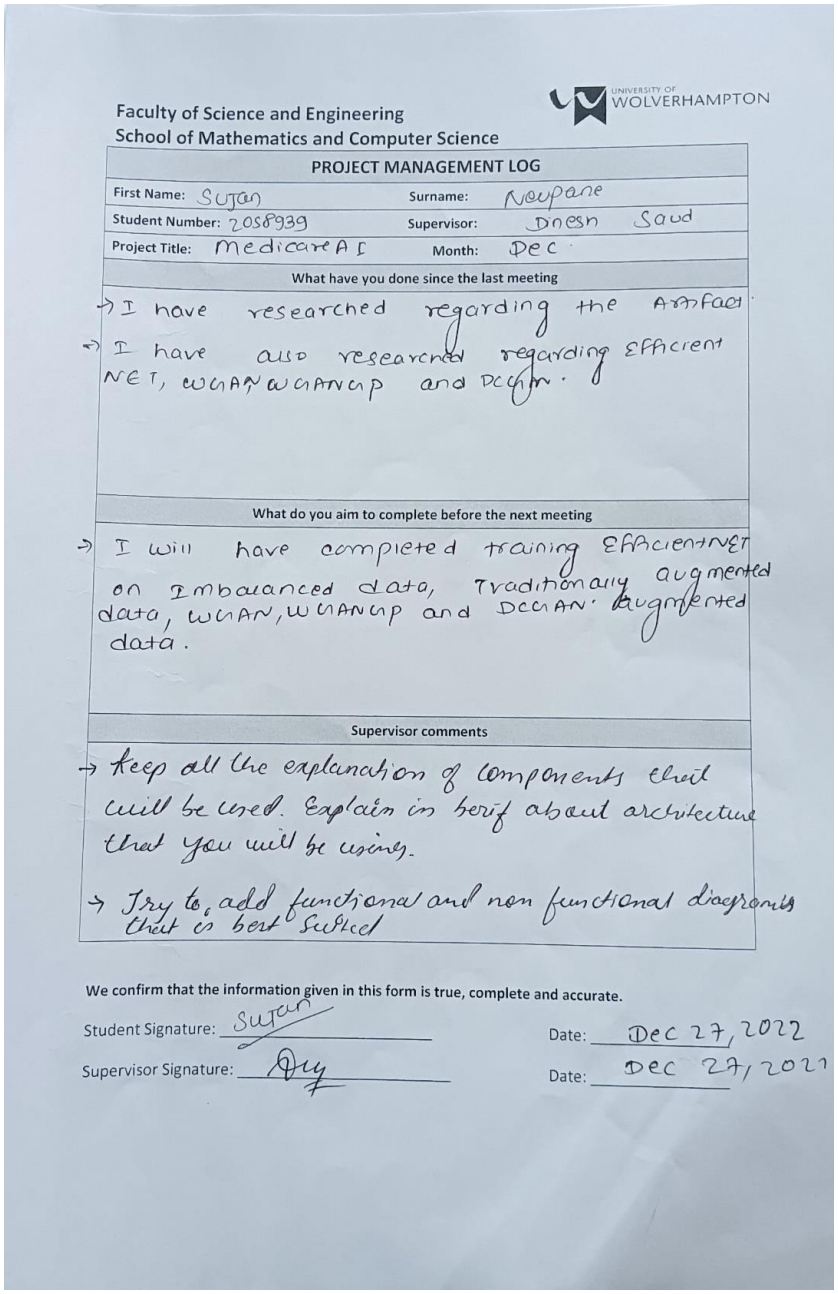
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: Dec 20, 2022

Supervisor Signature: Dinesh Date: Dec 20, 2022

Figure 78 Supervisor Meeting held on Dec 20, 2022

8.1.6 Supervisor Meeting held on Dec 27, 2022



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name:	Sujan	Surname:	Neupane
Student Number:	2058939	Supervisor:	Dinesh Saud
Project Title:	MedicareAI	Month:	Dec

What have you done since the last meeting

→ I have researched regarding the ArmFace.
→ I have also researched regarding Efficient NET, WGAN, WGANGP and DCGAN.

What do you aim to complete before the next meeting

→ I will have completed training EfficientNET on Imbalanced data, Traditionally augmented data, WGAN, WGANGP and DCGAN augmented data.

Supervisor comments

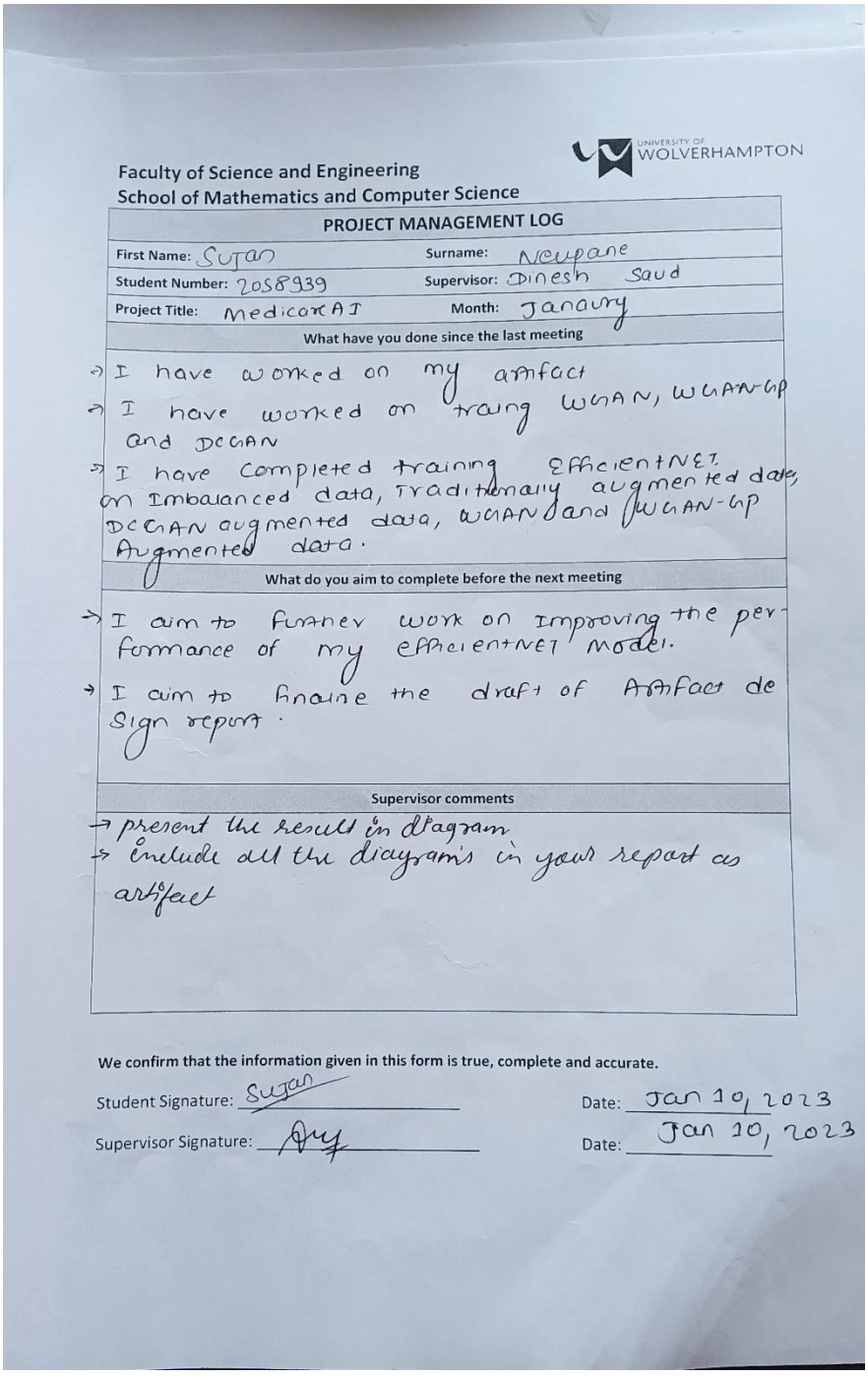
→ Keep all the explanation of components that will be used. Explain in brief about architecture that you will be using.
→ Try to add functional and non functional diagrams that is best suited

We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: Dec 27, 2022
Supervisor Signature: Dinesh Date: Dec 27, 2022

Figure 79 Supervisor Meeting held on Dec 27, 2022

8.1.7 Supervisor Meeting held on Jan 10, 2023



UNIVERSITY OF WOLVERHAMPTON

Faculty of Science and Engineering
School of Mathematics and Computer Science

PROJECT MANAGEMENT LOG

First Name: Sujan	Surname: Neupane
Student Number: 2058939	Supervisor: Dinesh Saud
Project Title: MedicorAI	Month: January

What have you done since the last meeting

- I have worked on my artifact
- I have worked on traing WGAN, WGAN-GP and DCGAN
- I have completed training EfficientNET on Imbalanced data, Traditionally augmented data, DCGAN augmented data, WGAN and WGAN-GP Augmented data.

What do you aim to complete before the next meeting

- I aim to further work on Improving the performance of my efficientNET Model.
- I aim to finalize the draft of Artifact design report.

Supervisor comments

- present the result in diagram
- include all the diagrams in your report as artifact

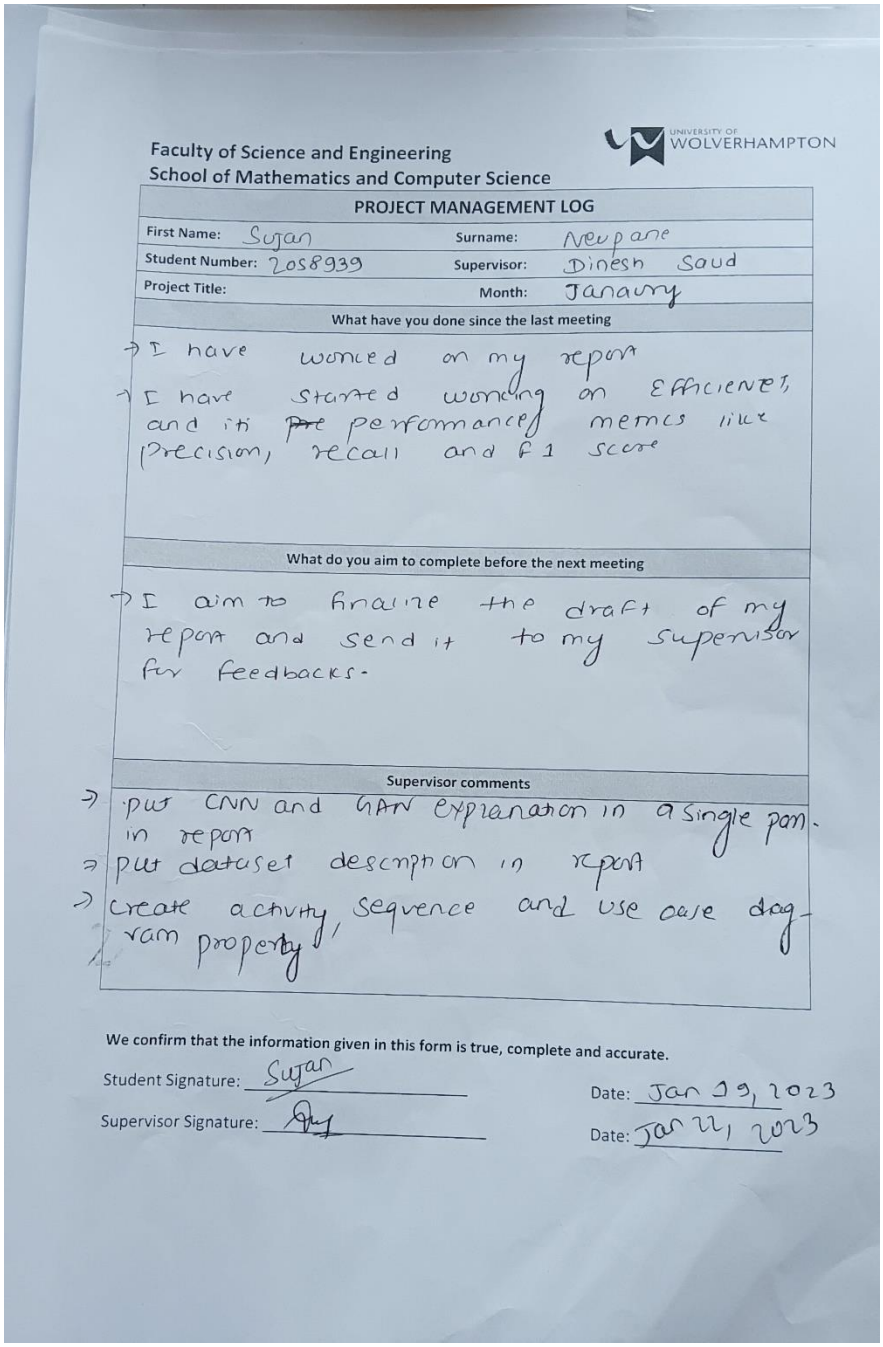
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: Jan 10, 2023

Supervisor Signature: Dinesh Date: Jan 10, 2023

Figure 80 Supervisor Meeting held on Jan 10, 2023

8.1.8 Supervisor Meeting held on Jan 19, 2023



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name:	Sujan	Surname:	Neupane
Student Number:	2058939	Supervisor:	Dinesh Saud
Project Title:		Month:	January

What have you done since the last meeting

→ I have worked on my report
→ I have started working on Efficientnet and its ~~pre~~ performance metrics like Precision, recall and f1 score

What do you aim to complete before the next meeting

→ I aim to finalize the draft of my report and send it to my supervisor for feedbacks.

Supervisor comments

→ put CNN and GAN explanation in a single part in report
→ put dataset description in report
→ create activity sequence and use case diagram properly.

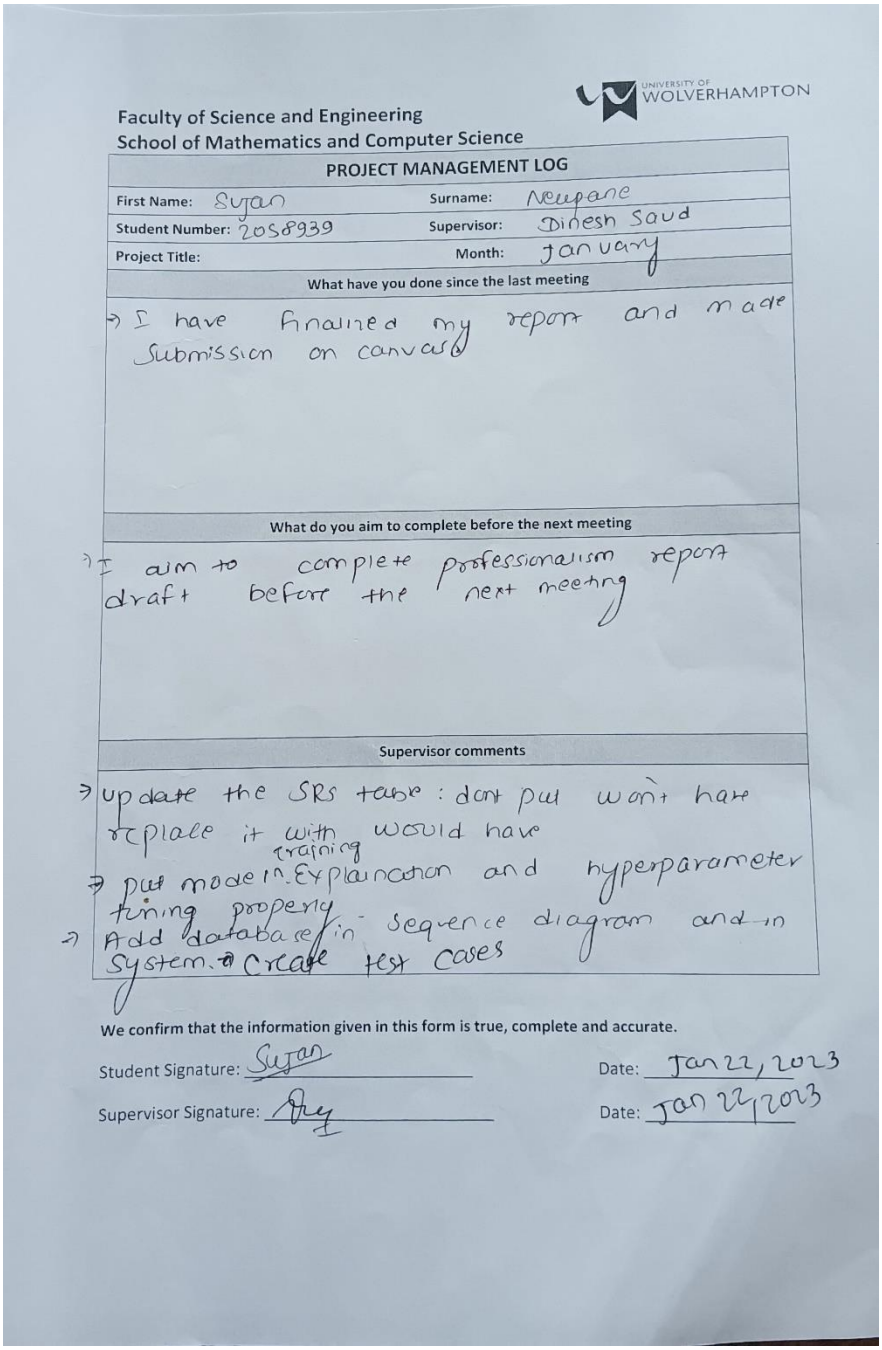
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: Jan 19, 2023

Supervisor Signature: [signature] Date: Jan 22, 2023

Figure 81 Supervisor Meeting held on Jan 19, 2023

8.1.9 Supervisor Meeting held on Jan 22, 2023



UNIVERSITY OF WOLVERHAMPTON

Faculty of Science and Engineering
School of Mathematics and Computer Science

PROJECT MANAGEMENT LOG

First Name:	Sujan	Surname:	Neupane
Student Number:	2058939	Supervisor:	Dinesh Saud
Project Title:		Month:	January

What have you done since the last meeting

→ I have finalized my report and made submission on canvas.

What do you aim to complete before the next meeting

→ I aim to complete professionalism report draft before the next meeting

Supervisor comments

→ update the SRS table : don't put won't have replace it with would have
→ put model training Explanation and hyperparameter tuning property
→ Add database in sequence diagram and in System. → Create test cases

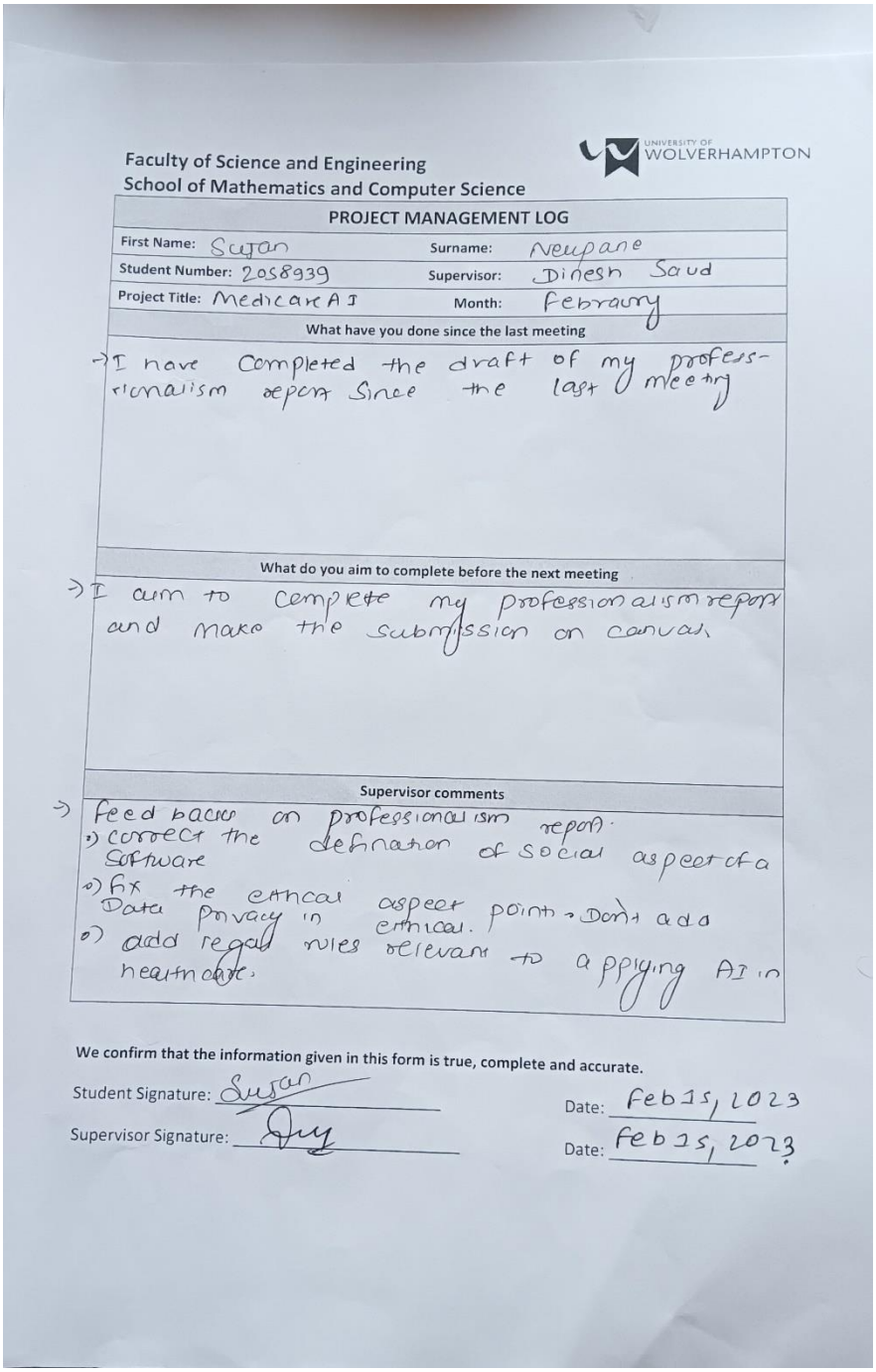
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: Jan 22, 2023

Supervisor Signature: Dinesh Date: Jan 22, 2023

Figure 82 Supervisor Meeting held on Jan 22, 2023

8.1.10 Supervisor Meeting held on Feb 15, 2023



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name:	Sujan	Surname:	Neupane
Student Number:	2058939	Supervisor:	Dinesh Saud
Project Title:	MedicareAI	Month:	Febraury

What have you done since the last meeting

→ I have completed the draft of my professionalism report since the last meetng

What do you aim to complete before the next meeting

→ I aim to complete my professionalism report and make the submission on canvas.

Supervisor comments

→ feed backs on professionalism report.
a) correct the defination of social aspect of a software
b) fix the ethical aspect point - Don't add Data privacy in ethical.
c) add regard wies relevant to applying AI in healthcare.

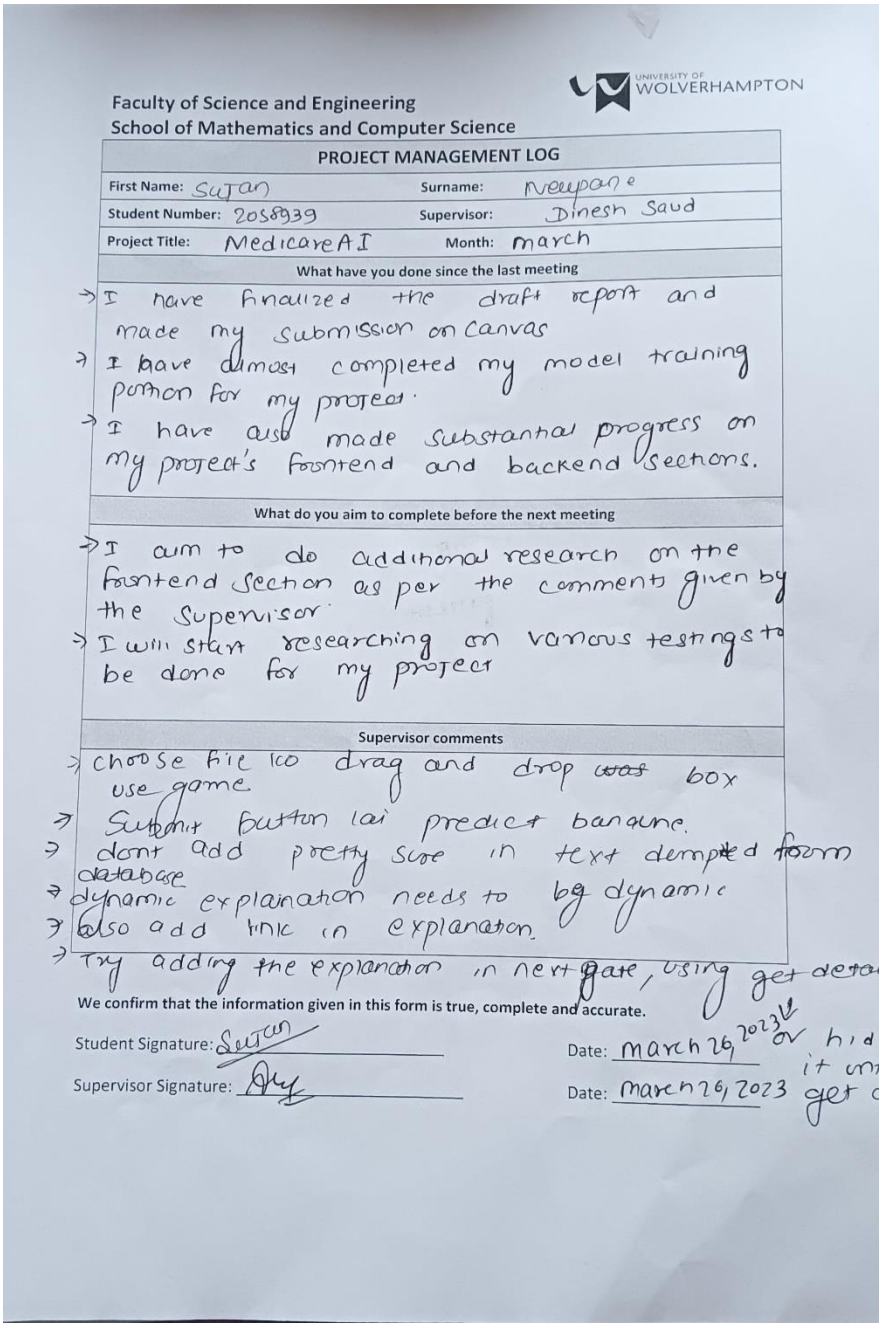
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: Feb 15, 2023

Supervisor Signature: Dinesh Date: Feb 15, 2023

Figure 83 Supervisor Meeting held on Feb 15, 2023

8.1.11 Supervisor Meeting held on Mar 26, 2023



UNIVERSITY OF WOLVERHAMPTON

Faculty of Science and Engineering
School of Mathematics and Computer Science

PROJECT MANAGEMENT LOG

First Name:	Sujan	Surname:	Neupane
Student Number:	2058939	Supervisor:	Dinesh Saud
Project Title:	MedicareAI	Month:	march

What have you done since the last meeting

- I have finalized the draft report and made my submission on canvas
- I have almost completed my model training portion for my project.
- I have also made substantial progress on my project's frontend and backend sections.

What do you aim to complete before the next meeting

- I aim to do additional research on the frontend section as per the comments given by the supervisor.
- I will start researching on various testings to be done for my project

Supervisor comments

- choose file ico drag and drop ~~was~~ box use game.
- Submit button lai predict banaune.
- dont add pretty score in text dempted from database
- dynamic explanation needs to bg dynamic
- also add link in explanation.
- Try adding the explanation in next page, using get deta

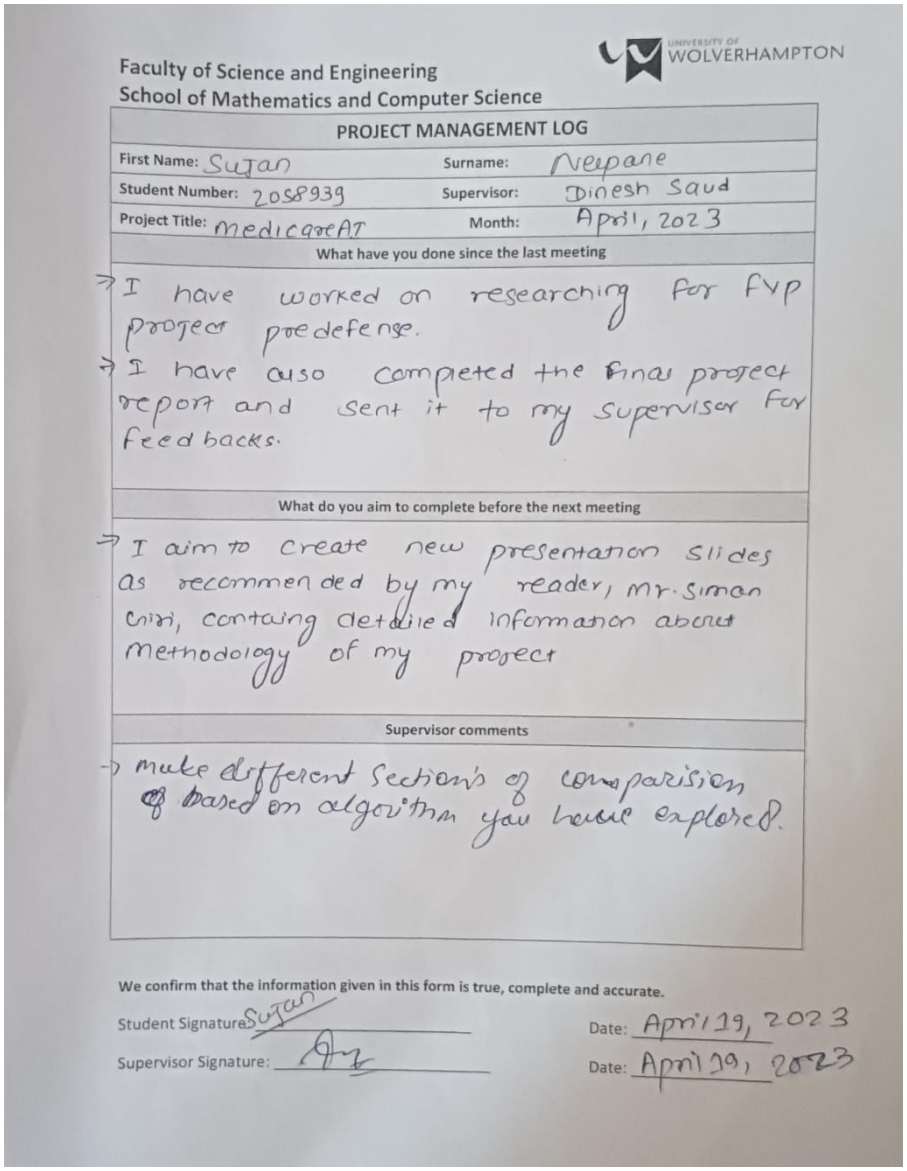
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: March 26, 2023

Supervisor Signature: Dinesh Date: March 26, 2023

Figure 84 Supervisor Meeting held on Mar 26, 2023

8.1.12 Supervisor Meeting held on April 19, 2023



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name: Sujan	Surname: Neupane		
Student Number: 2058939	Supervisor: Dinesh Saud		
Project Title: MedicareAI	Month: April, 2023		

What have you done since the last meeting

→ I have worked on researching for FYP project predefense.

→ I have also completed the final project report and sent it to my supervisor for feedbacks.

What do you aim to complete before the next meeting

→ I aim to create new presentation slides as recommended by my reader, Mr. Simon Giri, containg detailed information about methodology of my project

Supervisor comments

→ make different sections of comparision of based on algorithm you have explored.

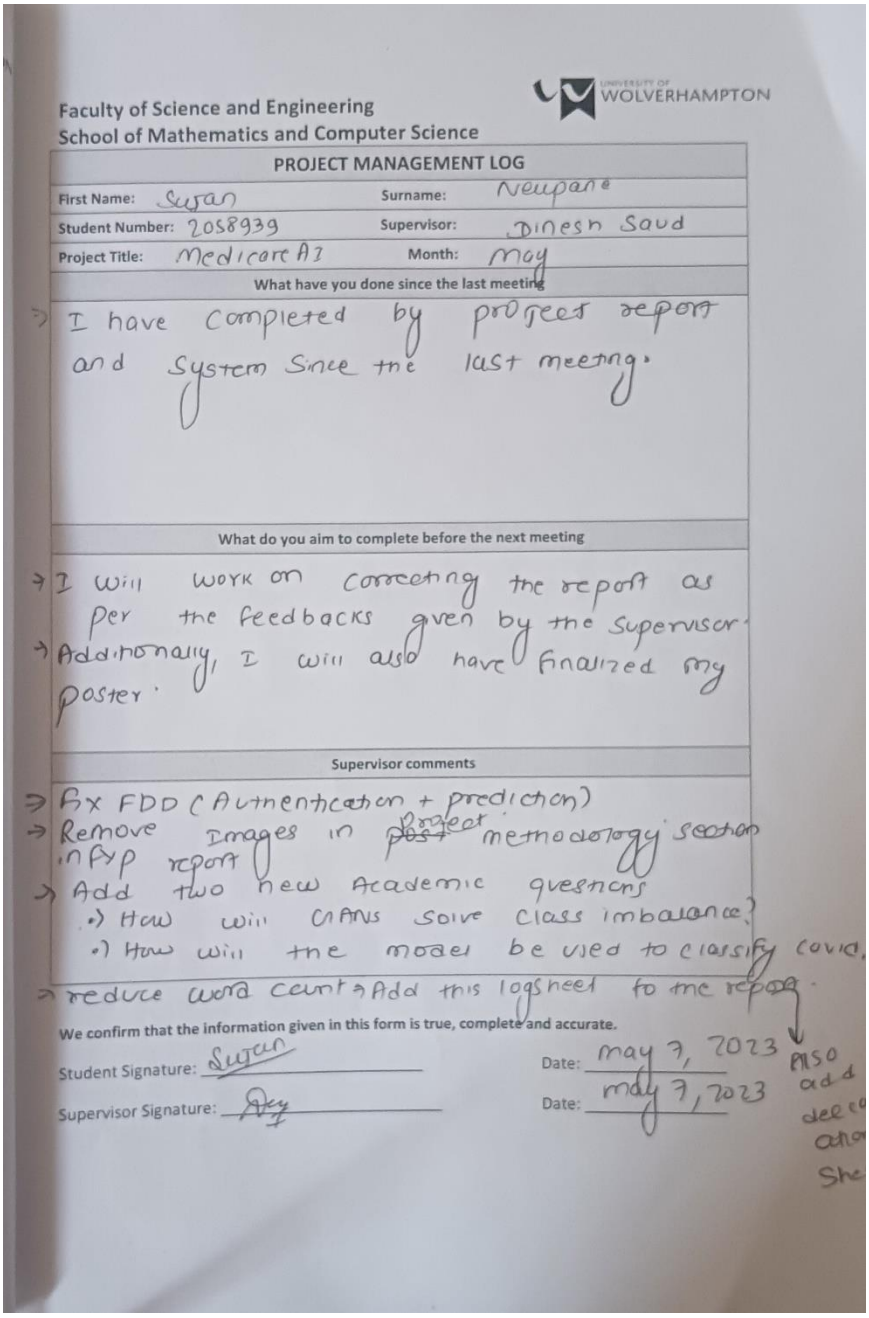
We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: April 19, 2023

Supervisor Signature: Date: April 19, 2023

Figure 85 Supervisor Meeting held on April 19, 2023

8.1.13 Supervisor Meeting held on May 7, 2023



Faculty of Science and Engineering
School of Mathematics and Computer Science

UNIVERSITY OF WOLVERHAMPTON

PROJECT MANAGEMENT LOG

First Name:	Sujan	Surname:	Neupane
Student Number:	2058939	Supervisor:	Dinesh Saud
Project Title:	Medicare AI	Month:	May

What have you done since the last meeting

→ I have completed by project report and System Since the last meeting.

What do you aim to complete before the next meeting

→ I will work on correcting the report as per the feedbacks given by the supervisor.
→ Additionally, I will also have finalized my poster.

Supervisor comments

→ Fix FDD (Authentication + prediction)
→ Remove Images in ~~post~~ Project methodology section in FYP report
→ Add two new Academic questions
  • How will GANs solve class imbalance?
  • How will the model be used to classify covid,
→ reduce word count → Add this logsheet to the report.

We confirm that the information given in this form is true, complete and accurate.

Student Signature: Sujan Date: May 7, 2023

Supervisor Signature: [signature] Date: May 7, 2023

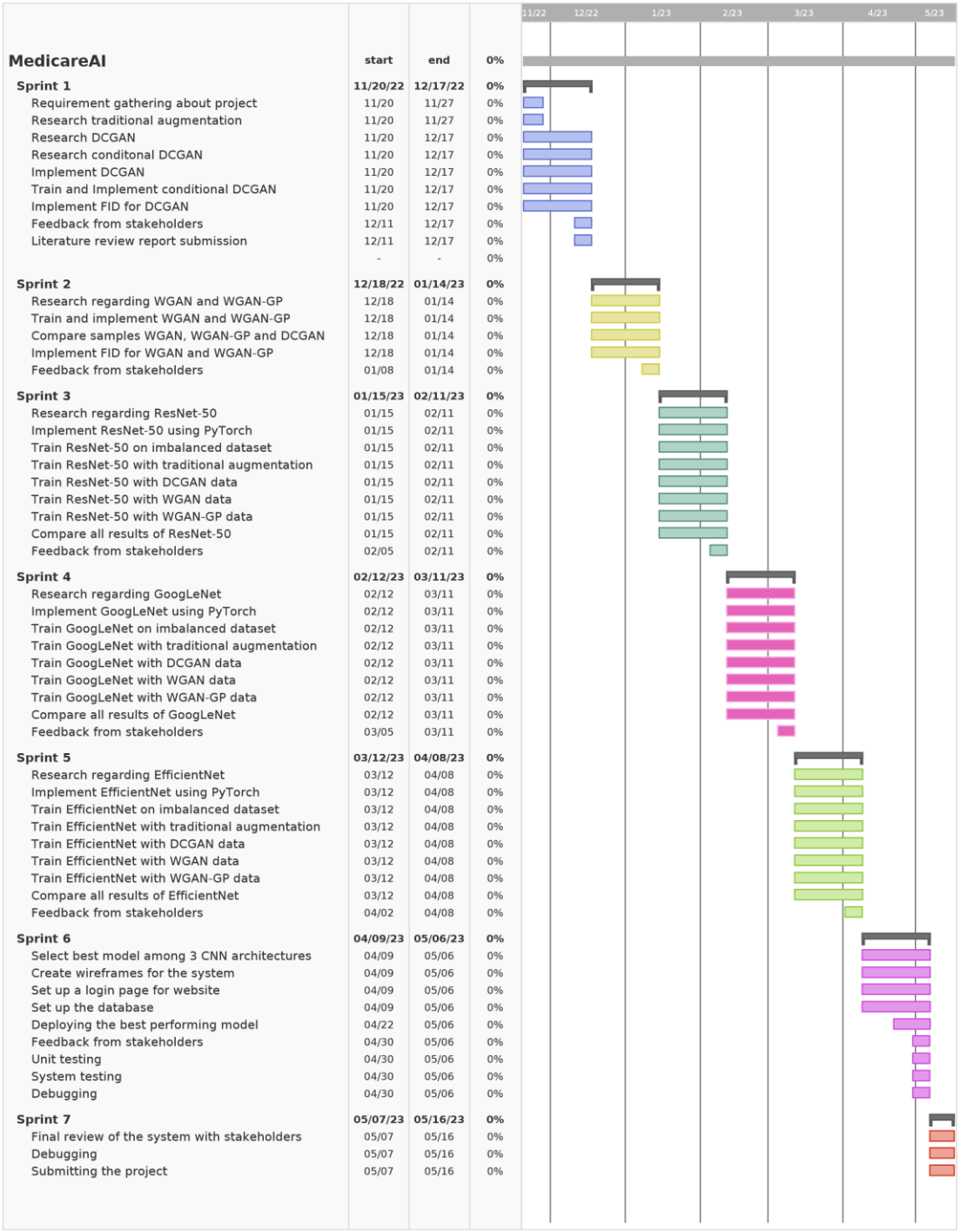
Also add dee ca ano She

Figure 86 Supervisor Meeting held on May 7, 2023

8.2 Gantt Chart

teamgantt
Created with Free Edition

Printed: 11/16/2022



MedicareAI	start	end	0%
Sprint 1	**11/20/22**	**12/17/22**	**0%**
Requirement gathering about project	11/20	11/27	0%
Research traditional augmentation	11/20	11/27	0%
Research DCGAN	11/20	12/17	0%
Research conditonal DCGAN	11/20	12/17	0%
Implement DCGAN	11/20	12/17	0%
Train and Implement conditional DCGAN	11/20	12/17	0%
Implement FID for DCGAN	11/20	12/17	0%
Feedback from stakeholders	12/11	12/17	0%
Literature review report submission	12/11	12/17	0%
	-	-	0%
Sprint 2	**12/18/22**	**01/14/23**	**0%**
Research regarding WGAN and WGAN-GP	12/18	01/14	0%
Train and implement WGAN and WGAN-GP	12/18	01/14	0%
Compare samples WGAN, WGAN-GP and DCGAN	12/18	01/14	0%
Implement FID for WGAN and WGAN-GP	12/18	01/14	0%
Feedback from stakeholders	01/08	01/14	0%
Sprint 3	**01/15/23**	**02/11/23**	**0%**
Research regarding ResNet-50	01/15	02/11	0%
Implement ResNet-50 using PyTorch	01/15	02/11	0%
Train ResNet-50 on imbalanced dataset	01/15	02/11	0%
Train ResNet-50 with traditional augmentation	01/15	02/11	0%
Train ResNet-50 with DCGAN data	01/15	02/11	0%
Train ResNet-50 with WGAN data	01/15	02/11	0%
Train ResNet-50 with WGAN-GP data	01/15	02/11	0%
Compare all results of ResNet-50	01/15	02/11	0%
Feedback from stakeholders	02/05	02/11	0%
Sprint 4	**02/12/23**	**03/11/23**	**0%**
Research regarding GoogLeNet	02/12	03/11	0%
Implement GoogLeNet using PyTorch	02/12	03/11	0%
Train GoogLeNet on imbalanced dataset	02/12	03/11	0%
Train GoogLeNet with traditional augmentation	02/12	03/11	0%
Train GoogLeNet with DCGAN data	02/12	03/11	0%
Train GoogLeNet with WGAN data	02/12	03/11	0%
Train GoogLeNet with WGAN-GP data	02/12	03/11	0%
Compare all results of GoogLeNet	02/12	03/11	0%
Feedback from stakeholders	03/05	03/11	0%
Sprint 5	**03/12/23**	**04/08/23**	**0%**
Research regarding EfficientNet	03/12	04/08	0%
Implement EfficientNet using PyTorch	03/12	04/08	0%
Train EfficientNet on imbalanced dataset	03/12	04/08	0%
Train EfficientNet with traditional augmentation	03/12	04/08	0%
Train EfficientNet with DCGAN data	03/12	04/08	0%
Train EfficientNet with WGAN data	03/12	04/08	0%
Train EfficientNet with WGAN-GP data	03/12	04/08	0%
Compare all results of EfficientNet	03/12	04/08	0%
Feedback from stakeholders	04/02	04/08	0%
Sprint 6	**04/09/23**	**05/06/23**	**0%**
Select best model among 3 CNN architectures	04/09	05/06	0%
Create wireframes for the system	04/09	05/06	0%
Set up a login page for website	04/09	05/06	0%
Set up the database	04/09	05/06	0%
Deploying the best performing model	04/22	05/06	0%
Feedback from stakeholders	04/30	05/06	0%
Unit testing	04/30	05/06	0%
System testing	04/30	05/06	0%
Debugging	04/30	05/06	0%
Sprint 7	**05/07/23**	**05/16/23**	**0%**
Final review of the system with stakeholders	05/07	05/16	0%
Debugging	05/07	05/16	0%
Submitting the project	05/07	05/16	0%

Figure 128 Gantt Chart

9. References

Anon., 2020. Complex Network Classification with Convolutional Neural Network. *Tsinghua Science and Technology ,* 25(4), pp. 1-11.

Bajaj, A., 2022. *Understanding Gradient Clipping (and How It Can Fix Exploding Gradients Problem).* [Online]
Available at: https://neptune.ai/blog/understanding-gradient-clipping-and-how-it-can-fixexploding-gradients-problem
[Accessed 16 January 2023].

Campos, V. et al., 2018. *Scaling a Convolutional Neural Network for classification of Adjective Noun Pairs with TensorFlow on GPU Clusters.* Madrid, IEEE.

Deep Dive Into Deep Learning, 2023. *Transposed Convolution.* [Online] Available at: https://d2l.ai/chapter_computer-vision/transposed-conv.html [Accessed 15 January 2023].

Dive into Deep Learning, 2023. *Padding and Stride.* [Online]
Available at: https://d2l.ai/chapter_convolutional-neural-networks/padding-andstrides.html
[Accessed 16 January 2023].

Donghwan , K., Joo, J. & Suk , K. C., 2022. *Fake Data Generation for Medical Image Augmentation using GANs.* Jeju Island, IEEE.

Fierce Healthcare, 2021. *Data privacy solution TripleBlind secures $24M backed by General Catalyst, Mayo Clinic.* [Online]
Available at: https://www.fiercehealthcare.com/tech/data-privacy-solution-tripleblindsecures-24m-backed-by-general-catalyst-mayo-clinic [Accessed 9 November 2022].

Google, 2023. *Overview of GAN Structure.* [Online]
Available at: https://developers.google.com/machine-learning/gan/gan_structure
[Accessed 15 January 2023].

Gugger, S., 2018. *Convolution in depth.* [Online]
Available at: https://sgugger.github.io/convolution-in-depth.html [Accessed
14 January 2022].

Hasan, F., 2023. *What are some deep details about pooling layers in CNN?.* [Online]
Available at: https://www.educative.io/answers/what-are-some-deep-details-aboutpooling-layers-in-cnn [Accessed 16 January 2023].
Hussain, B. Z. et al., 2022. *Wasserstein GAN based Chest X-Ray Dataset.* Glasgow, IEEE.

IBM, 2020. *Neural networks from scratch.* [Online]
Available at: https://developer.ibm.com/learningpaths/get-started-with-deeplearning/neural-networks-from-scratch/ [Accessed 16 January 2023].

IBM, 2022. *Generative adversarial networks explained.* [Online]
Available at: https://developer.ibm.com/articles/generative-adversarial-networksexplained/
[Accessed 16 January 2023].

Liang, Z., Huang, X. J., Li, J. & Chan, S., 2020. *Enhancing Automated COVID-19 Chest X-ray Diagnosis by Image-to-Image GAN Translation.* Seoul, IEEE.

Ma, E., 2019. *How does ResNet improve performance.* [Online]
Available at: https://medium.com/dataseries/how-does-resnet-improve-performancecaaa436f885b
[Accessed 12 November 2022].

Makhanov, N., Tu, N. A. & Wong, K.-S., 2022. *A Survey on Deep Learning Advances and Emerging Issues in Pneumonia and COVID19 Prediction.* Daegu, IEEE.

MathWorks, 2023. *What Is a Convolutional Neural Network?.* [Online]
Available at: https://www.mathworks.com/discovery/convolutional-neural-networkmatlab.html
[Accessed 14 January 2023].

Mirri.AI, 2022. *Next-Gen Data For AI.* [Online]
Available at: https://www.mirry.ai/main [Accessed
9 Nov 2022].

Papers with code, 2023. *EfficientNet.* [Online]
Available at: https://paperswithcode.com/method/efficientnet [Accessed
16 January 23023].

pawangfg, 2018. *Understanding GoogLeNet Model – CNN Architecture.* [Online]
Available at: https://www.geeksforgeeks.org/understanding-googlenet-model-cnnarchitecture/
[Accessed 12 November 2022].

Riva, M., 2022. *Batch Normalization in Convolutional Neural Networks.* [Online]
Available at: https://www.baeldung.com/cs/batch-normalizationcnn#:~:text=Normalization%20is%20a%20pre%2Dprocessing,and%20decrease%20its %20learning%20speed.
[Accessed 14 January 2022].

Suganya, D. & Kalpana, R., 2022. *Automated COVID-19 diagnosis using Deep Multiple Instance Learning with CycleGAN.* Villupuram, IEEE.

Thakur, A., 2022. *How to Evaluate GANs using Frechet Inception Distance (FID).* [Online]
Available at: https://wandb.ai/ayush-thakur/gan-evaluation/reports/How-to-EvaluateGANs-using-Frechet-Inception-Distance-FID---Vmlldzo0MTAxOTI
[Accessed 24 March 2023].

Tonic, 2022. *Using Neural Networks to Synthesize Complex Data Relationships, with AI Synthesizer.* [Online]
Available at: https://www.tonic.ai/blog/using-neural-networks-to-synthesize-complexdata-relationships-with-ai-synthesizer [Accessed 9 November 2022].

TrailTwin, 2022. *What is Trial Twin?.* [Online]
Available at: https://trialtwin.com/ [Accessed
9 November 2022].

TripleBlind, 2022. *The HIPAA-Ready Data Solution.* [Online]
Available at: https://tripleblind.ai/ [Accessed
9 November 2022].

Turing, 2023. *How to Choose an Activation Function For Deep Learning.* [Online]
Available at: https://www.turing.com/kb/how-to-choose-an-activation-function-for-deeplearning
[Accessed 16 January 2023].

Tutorials Point, 2023. *Software Requirement Specification.* [Online] Available at:
https://www.tutorialspoint.com/software_testing_dictionary/software_requirement_specification.htm
[Accessed 20 January 2023].

Venu, S. K. & Ravula, S., 2021. Evaluation of Deep Convolutional Generative Adversarial. *MDIP,* pp. 1-13.

Viel, 2022. *SOLUTIONS ENABLING BETTER USE OF HEALTH DATA FOR RESEARCH, DEVELOPMENT AND INNOVATION.* [Online]
Available at: https://veil.ai/
[Accessed 9 November 2022].

Waheed, A. et al., 2020. CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection. *IEEE,* Volume 8, pp. 1-8. WisdomPlexus, 2022. *Pros and Cons of Kanban Explained.* [Online]
Available at: https://wisdomplexus.com/blogs/pros-cons-kanban/ [Accessed 15 November 2022].

Yaren , Ö. & Deniz, T., 2021. *Comparison of Deep Learning Models AlexNet and.* Istanbul, IEEE.

Appendix

1. Convolutional Neural Networks

Convolutional neural networks or CNN are a class of deep learning algorithms that are used in the field of computer vision to perform various kinds of tasks like image classification, recognition, and object detection. CNNs are similar to an ordinary neural network like a multilayer perceptron; they are made up of neurons that have learnable parameters. Like an ordinary neural network, they have an input layer, one or more hidden layers, and an output layer. CNNs also comprise activation functions like ReLu, and LeakyReLU (MathWorks, 2023). There are multiple components in a CNN, and they are explained below.

1.1 Convolutional layer

The convolutional layer is the main layer of a CNN. The majority of operations, called convolutions, occur in this layer. In the node or initial layer of a CNN, an input image and a filter are required. If an image is RGB, it has 3 dimensions: number of channels, height and width of an image. The feature detector or filter is moved across the receptive fields of the image to check if there is an underlying feature or pattern in the image. A filter is a collection of kernels. A kernel moves across the image with a factor of a variable called stride. I

In a 2d convolution operation or convolution, we start with a kernel which is simply a matrix of weights that are randomly initialized. The kernel slides over the image array or matrix and performs element wise multiplication. Finally, the resultant matrix is summed up to return a single pixel. Now, the kernel slides over the image by a factor of stride and performs elementwise multiplication and sums of the results to return another pixel. This way, a kernel repeats the process for every location present in the image resulting in the conversion of an original image to a 2d matrix of features.

Here, the size of a kernel is also important because the larger the kernel, the more the features/pixels are multiplied and combined to produce new features (Anon., 2020).

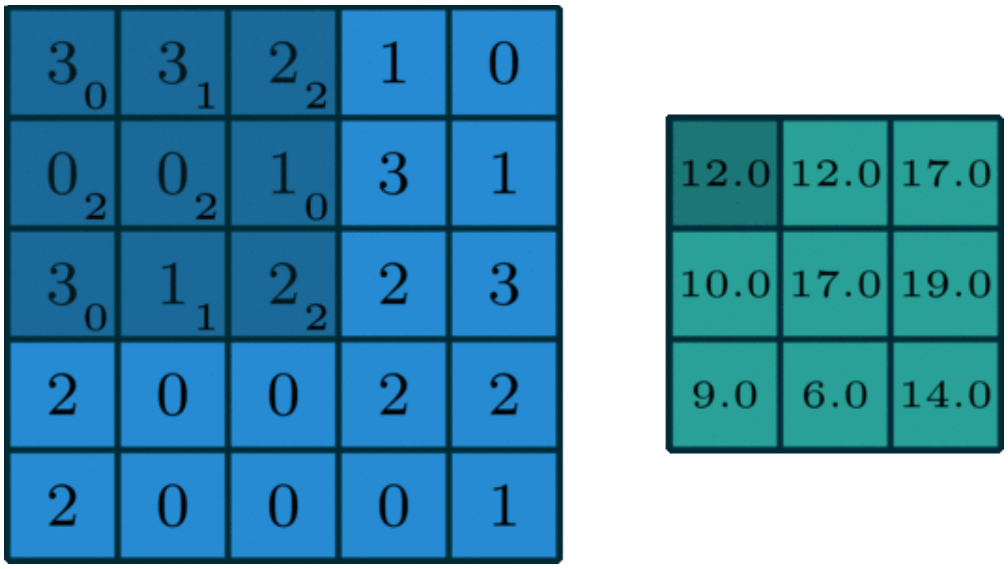


Figure 87 Convolution operation (Irhum Shafkat)

A filter is a collection of kernels used in the convolution operation. In case of an RGB image, we use 3 individual kernels for each channel, which are collectively called as a filter. Similarly, in case of a greyscale image, a filer contains only 1 kernel due to the input image having only one channel. Therefore, in this case, a kernel is same as a filter.

Let us consider the following RGB image.

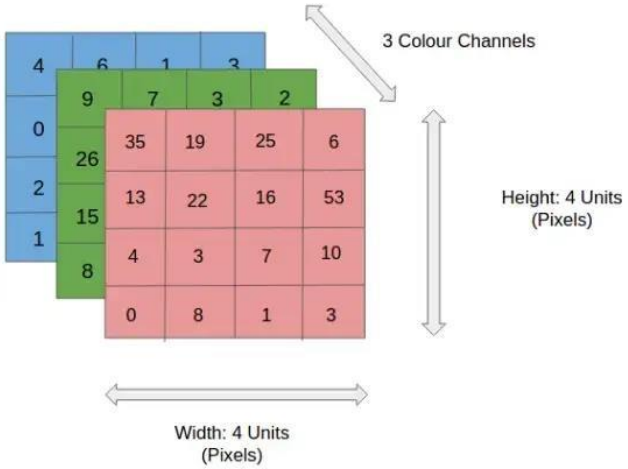


*Figure 88 A 3*4*4 RGB image (Sumit Saha)*

The above image has 3 dimensions: number of channels (3), height (4) and width (4). Let us suppose that we are applying 10 filters to this image; this means that the resultant feature map will have 10 channels. Since this is an RGB image, each one of the 10 filters will have 3 kernels that will be multiplied to each channel of the image. Each kernel of each filter will convolution operation and return 3 resultant 2d matrix of features which will be combined to from a single channel. Additionally, each filter also has a bias term which will be added to the resultant single channel formed after summing each result of the kernels. This way, we have one output channel for each filter leading to a total of 10 channels in the resultant output of the convolution operation. Since the output has 10 channels, it can't be called a image. It is rather called as a **feature map**.

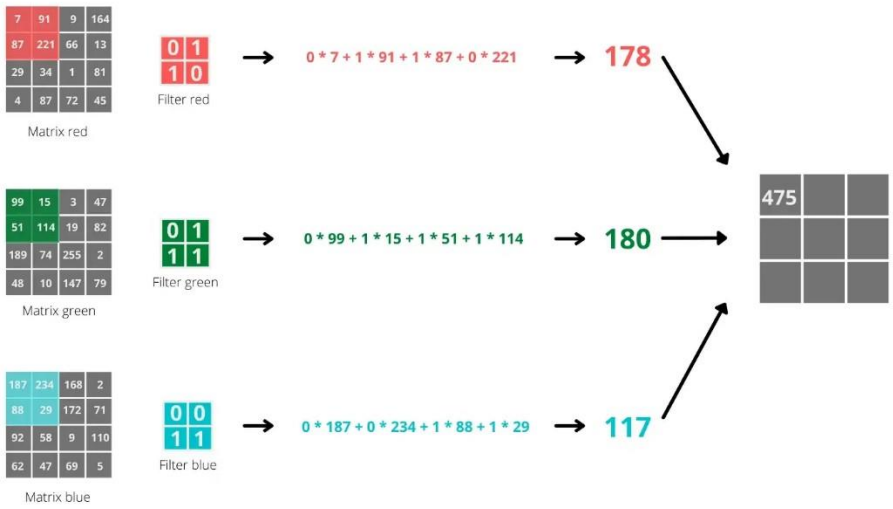


Figure 89 Convolution operation in RGB image (Niklas Lang)

A complete convolution operation is explained below using PyTorch.

By Sujan Neupane

In [80]:
```
import torch.nn as nn
import torch.nn.functional as F
import torch
```

In [81]:
```
# a 3*5*5 array
a_3_channel_5by5_image = torch.randn((3,5,5))
```

*Figure 90 Creating a 3*5*5 array*

Here, I have created a tensor with 3 channels and height and width of 5 containing random numbers from a normal distribution.

Convolution layers

```
In [82]:
1 conv_layer1 = nn.Conv2d(3, 8, kernel_size=3, stride=1, padding=1)
2 print("WEIGHTs SHAPE: ", conv_layer1.weight.shape)
3 print("BIAS SHAPE: ", conv_layer1.bias.shape)

WEIGHTs SHAPE:  torch.Size([8, 3, 3, 3])
BIAS SHAPE:  torch.Size([8])

In [83]:
1 conv_layer2 = nn.Conv2d(8, 1, kernel_size=5, stride=1, padding=0)
2 print("WEIGHTs SHAPE: ", conv_layer2.weight.shape)
3 print("BIAS SHAPE: ", conv_layer2.bias.shape)

WEIGHTs SHAPE:  torch.Size([1, 8, 5, 5])
BIAS SHAPE:  torch.Size([1])
```

Figure 91 Two convolution layers

In the first cell, I initialized the first convolution layer that takes in a 3-channel tensor. A total of 8 filters will be applied to the input image with stride and padding set to 1. In the second convolutional layer, it will take a feature map with 8 channels and apply 1 filter containing 8 kernels with padding set to 0 and stride set to 1.

Convolution operations for RGB image

```
In [84]:
1 a_3_channel_5by5_image = a_3_channel_5by5_image.reshape((1,3,5,5))
2 output_from_first_convlayer = conv_layer1(a_3_channel_5by5_image)
3 print("OUTPUT FROM FIRST CONV LAYER: ", output_from_first_convlayer.shape)

OUTPUT FROM FIRST CONV LAYER:  torch.Size([1, 8, 5, 5])

In [85]:
1 output_from_second_convlayer = conv_layer2(output_from_first_convlayer)
2 print("OUTPUT FROM SECOND CONV LAYER: ", output_from_second_convlayer.shape)

OUTPUT FROM SECOND CONV LAYER:  torch.Size([1, 1, 1, 1])

In [86]:
1 output_from_second_convlayer

Out[86]: tensor([[[[-0.0954]]]], grad_fn=<ConvolutionBackward0>)
```

Figure 92 Output from convolutional layers

Here, the input to the first conv layer is 3 * 5 * 5. A total of 8 filters each with 3 kernels are applied. The padding and strides are set to 1 due to which the resultant feature map will have preserved the dimension of original tensor. Therefore, the shape of the resultant feature map will be 8 * 5 * 5. Now, this feature map is passed onto the second conv layer.

The second conv layer will take in input with 8 channels and apply only 1 filter containing 8 kernels which will perform element wise multiplication and sum the result. The 8 kernels will result in 8 2d matrices of features which will be summed to form a single matrix and finally bias term for this layer is added. Since padding is set to 0 and the kernel size is equal to the width and height of the feature map, the kernel element wise matrix multiplication and sum will result in a single pixel for each of 8 kernels which are added to form a single pixel. Therefore, this layer will result in a single pixel.

1.2 Stride and Padding

Padding refers to adding empty pixels around the image which will be useful to preserve the original dimension of the image in a feature map.

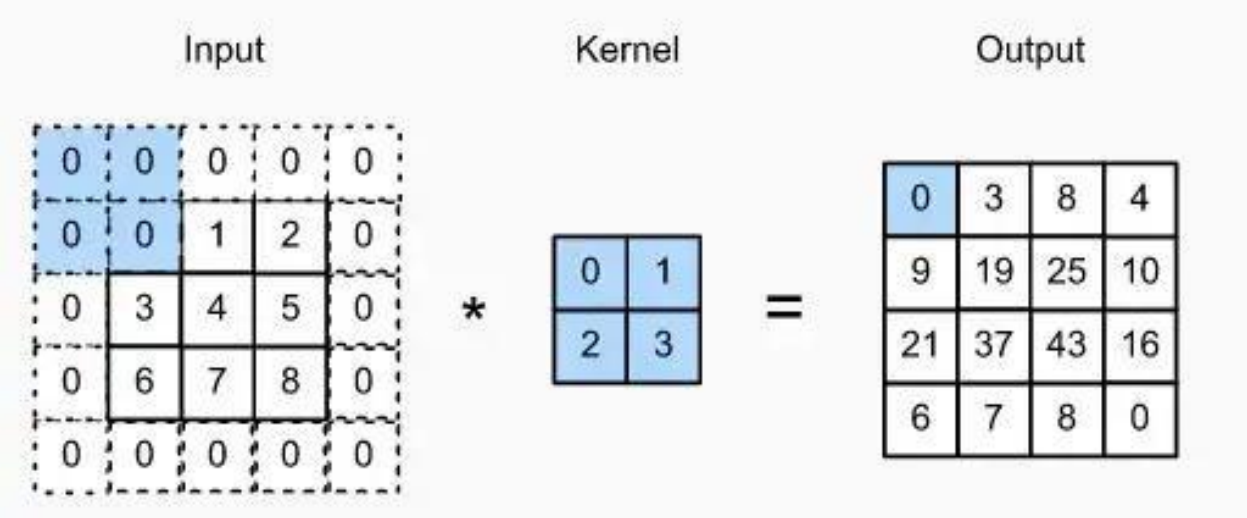


Figure 93 Padding in CNN (Abhisek Kumar Pandey)

Stride refers to the number of pixels to shift by a kernel while performing a convolutional operation (Dive into Deep Learning, 2023).

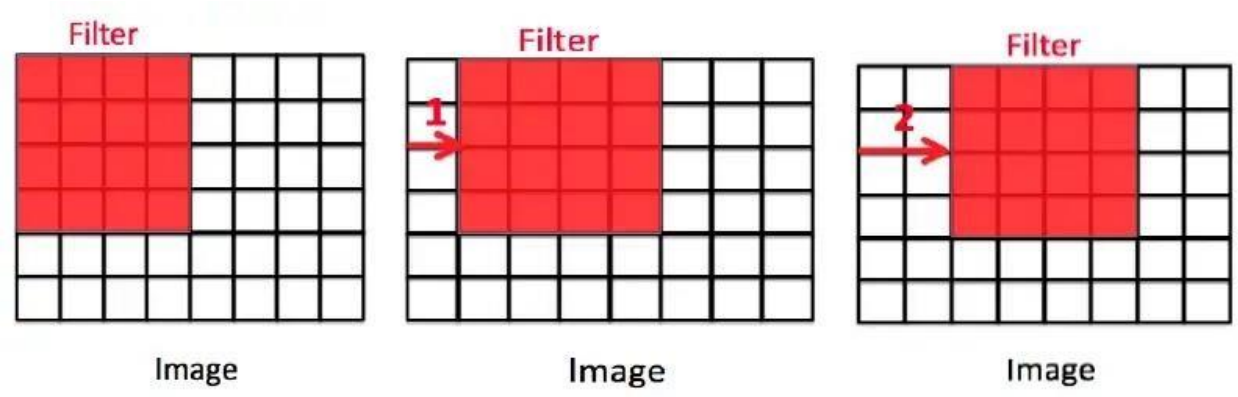


Figure 94 Stride in CNN (Abhisek Kumar Pandey)

1.3 Pooling Layer

The polling layer, like the convolutional layer, is used to reduce the dimensions or spatial size of the convolved feature. This is primarily done to perform dimensionality reduction and reduce the amount of computational power required to perform data processing. Additionally, it is also used to extract dominant features from a feature map (Hasan, 2023). There are two types of pooling:

- Max pooling
- Average pooling

In max pooling, the maximum value from the receptive area of a 2d kernel is used.

Whereas in average pooling, the average of all pixels is used (Gugger, 2018).

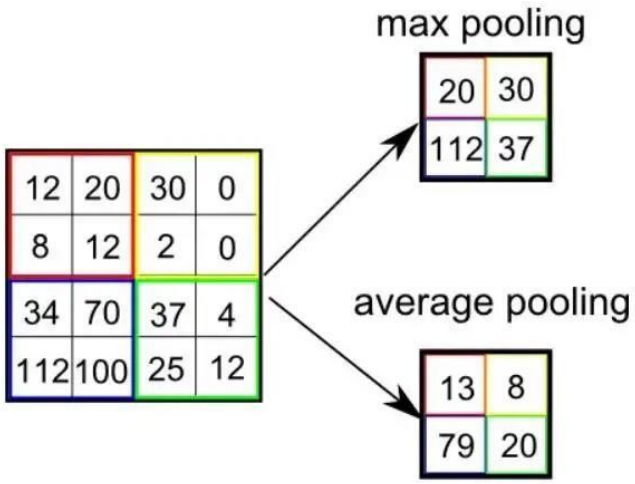


Figure 95 Types of pooling (Sumit Saha)

The following screenshots explain how max and average pooling can be implemented using PyTorch.

Max pooling

```
In [94]: 1 tensor = torch.tensor(np.array([[1.,2,3,4],[1,2,3,4]]));tensor
Out[94]: tensor([[1., 2., 3., 4.],
                 [1., 2., 3., 4.]], dtype=torch.float64)

In [96]: 1 nn.MaxPool2d(2,2)(tensor.reshape(1,1,2,4))
Out[96]: tensor([[[[2., 4.]]]], dtype=torch.float64)
```

Average pooling

```
In [98]: 1 nn.AvgPool2d(2,2)(tensor.reshape(1,1,2,4))
Out[98]: tensor([[[[1.5000, 3.5000]]]], dtype=torch.float64)
```

Done by Sujan Neupane

Figure 96 Average and Max pooling in PyTorch

nn.Maxpool2d and nn.AvgPool2d are used to perform max and average pooling in PyTorch. A kernel size of 2*2 and stride of 1 is used to convert a 2 * 4 tensor to 1 * 2 tensor.

1.4 Normalization Layer

Normalization is a preprocessing technique that is used to standardize or scale data. When different features have values in different ranges, it will result in difficult training of our model and our model being stuck on local minimum of its loss function. Not normalizing our data can prevent our model from reaching its optimal performance. There are two main methods to perform feature scaling:

- Minmax Scaling
- Standard Scaling

In Minmax scaling, we can subtract each feature with the mean of the dataset and divide by the difference of maximum and minimum value of the dataset. This will effectively result in the dataset having a value range of [0,1] respectively.

$$x_{normalized} = \frac{x - m}{x_{max} - x_{min}}$$

Figure 97 Minmax scaling (Martin Riva)

Here, x refers to the sample in a feature, m refers to the mean of the feature, x_{max} and x_{min} refer to the maximum and minimum values in a feature.

In Standard scaling, we will normalize all data points such that the resultant dataset will have mean of 0 and standard deviation of 1. It is also called **Z-Score Normalization** (Campos, et al., 2018)**.**

$$x_{normalized} = \frac{x - m}{s}$$

Figure 98 Standard Scaling (Martin Riva)

Here, x refers to a sample in a dataset whereas m and s refer to the mean and standard deviation of the dataset.

Batch Normalization is primarily used in CNNs to normalize the pixels in a feature map in intermediate layers. While training in each mini batch of data, the tensors are normalized using following formula.

$$z^N = \left(\frac{z - m_z}{s_z} \right)$$

Figure 99 Batch Normalization formula (Martin Riva)

Here, m_z and s_z refer to batch mean and standard deviation.

Batch Normalization helps to speed up the training and help models converge faster. Features that are not normalized will result in the model's loss oscillating. Meaning, the model is not able to converge smoothly. When we input normalized tensors to a conv layer, the resultant feature map will have N channels with W width and H height. However, the resultant pixels in the feature maps are not necessarily normalized. Directly passing the un-normalized pixels in the feature map to another conv layer will inherently be bad for the entire model. Applying batch normalization to the feature map of each conv layer will result in reduction in the **internal covariate shift** of the network. This is so because, even if our input data is normalized, the distribution of pixels in the internal layers' feature maps are constantly changing (Riva, 2022).

The screenshots below implement batch normalization in PyTorch.

Batch Normalization

```
In [99]:
1 layer = nn.Sequential(nn.Conv2d(3, 8, kernel_size=3, stride=1, padding=1),
2                       nn.BatchNorm2d(8), #  BATCH NORMALIZTION
3                       nn.Conv2d(8, 1, kernel_size=5, stride=1, padding=0),
4                        )

In [100]:
1 layer(a_3_channel_5by5_image.reshape((1,3,5,5)))

Out[100]: tensor([[[[-0.1042]]]], grad_fn=<ConvolutionBackward0>)
```

Figure 100 Batch Normalization in PyTorch

1.5 Fully Connected Layer

After multiple conv layers, the dimension of the original image tensors is significantly reduced and important features are extracted, we can flatten the tensor in a vector and pass it into a fully connected layer or a linear layer to get the final output. This layer can be used to perform classification task by using an activation like SoftMax or Sigmoid to get predicted probabilities in terms of output.

Fully connected layer

In [115]:
```
# a 3*16*16 array
a_3_channel_5by5_image = torch.randn((1, 3,16,16))
```

In [118]:
```
layer = nn.Sequential(
                      # 3 * 16 * 16 -> FIRST CONV LAYER INPUT
                      nn.Conv2d(3, 8, kernel_size=3, stride=1, padding=1),
                      nn.BatchNorm2d(8), #  BATCH NORMALIZTION
                      nn.MaxPool2d(2, 2),
                      # 8 * 8 * 8 -> FIRST CONV LAYER OUTPUT

                      # 8 * 8 * 8 -> SECOND CONV LAYER INPUT
                      nn.Conv2d(8, 16, kernel_size=3, stride=1, padding=1),
                      nn.BatchNorm2d(16), #  BATCH NORMALIZATION
                      nn.MaxPool2d(2, 2),
                       # 16 * 4 * 4 -> SECOND CONV LAYER INPUT

                      # 16 * 4 * 4 -> THIRD CONV LAYER INPUT
                      nn.Conv2d(16, 1, kernel_size=3, stride=1, padding=1),
                      # 1 * 4 * 4 -> THIRD CONV LAYER OUTPUT


                      # FLATTENING THE TENSOR to a vector
                      nn.Flatten(),
                      nn.Linear(1*4*4, 3)
                      )
```

In [120]:
```
layer(a_3_channel_5by5_image)
```

Out[120]:
```
tensor([[-0.7627, -0.1518, -0.7800]], grad_fn=<AddmmBackward0>)
```

Figure 101 Fully connected layer in PyTorch example

This way, a fully connected layer can be added to the CNN architecture to perform binary or multiclass classification. The output from the above neural network can be passed through a SoftMax activation function to get the output in terms of predicted probabilities.

1.6 Forward pass, backward pass and Adam optimizer in CNN

An input tensor is passed through various layers of a CNN architecture. This flow of information from the input layers to the output layers is called **forward pass or forward propagation.** In forward propagation, our image tensor first gets through the initial conv layer which performs the convolutional operation and returns a feature map. This feature map undergoes batch normalization to reduce internal covariate shift, and a max-pooling layer is applied. Then, an activation like ReLu could be used.

The feature map passes through series of various convolutional layers for capturing important features from a larger image tensor and reduce its dimension. Finally, the feature map from the final conv layer is flattened into an array and passed onto the linear layer to make predictions and the corresponding loss is calculated. This whole process falls under forward pass. The image below explains the forward pass implemented in PyTorch for a simple CNN architecture.

Forward Pass

In [146]:
```
# a 3*16*16 array
a_3_channel_16by16_image = torch.randn((1, 3,16,16))

# the target for our image tensor
targets = torch.tensor(np.array([1.,0,0]))
```

In [147]:
```
layer = nn.Sequential(
                      # 3 * 16 * 16 -> FIRST CONV LAYER INPUT
                      nn.Conv2d(3, 8, kernel_size=3, stride=1, padding=1),
                      nn.BatchNorm2d(8), #  BATCH NORMALIZTION
                      nn.MaxPool2d(2, 2),
                      # 8 * 8 * 8 -> FIRST CONV LAYER OUTPUT

                      # 8 * 8 * 8 -> SECOND CONV LAYER INPUT
                      nn.Conv2d(8, 16, kernel_size=3, stride=1, padding=1),
                      nn.BatchNorm2d(16), #  BATCH NORMALIZATION
                      nn.MaxPool2d(2, 2),
                       # 16 * 4 * 4 -> SECOND CONV LAYER INPUT

                      # 16 * 4 * 4 -> THIRD CONV LAYER INPUT
                      nn.Conv2d(16, 1, kernel_size=3, stride=1, padding=1),
                      # 1 * 4 * 4 -> THIRD CONV LAYER OUTPUT


                      # FLATTENING THE TENSOR to a vector
                      nn.Flatten(),
                      nn.Linear(1*4*4, 3)
                      )
```

Figure 102 Forward pass in CNN

Here, a 3 * 16 * 16 is created using random numbers from a normal distribution. A target vector containing probabilities for a class is created. The tensor is passed through the CNN architecture to get an output.

```
In [148]:
1 output = layer(a_3_channel_16by16_image)
2 print("OUTPUT: ", output)

OUTPUT:  tensor([[ 0.5302,  0.5894, -0.3978]], grad_fn=<AddmmBackward0>)
```

Figure 103 Output from a CNN

After the model returns output in terms of predicated probabilities for each class, we calculate **Cross Entropy Loss**.

```
In [168]:
1 # Cross entropy loss
2 CEL = nn.CrossEntropyLoss()
3
4
5 ## calculating cross entropy loss
6 loss = CEL(targets, output)
7 print("CROSS ENTROPY LOSS: ", loss)

CROSS ENTROPY LOSS:  tensor(-0.3520, dtype=torch.float64, grad_fn=<DivBackward1>)
```

Figure 104 Cross entropy loss calculation

Now, we have both targets and predicted outputs. We use those values to calculate cross entropy loss, as this is a multiclass classification. Otherwise, we could have used Binary cross entropy loss (BCE Loss). These steps are part of forward propagation.

Here, our model has made very incorrect prediction because the parameters, weights, and biases, of our model are randomly initialized. Since there are three classes, the probability of each class is roughly 33 %. Now that we have our cross-entropy loss, we can use this loss to optimize our model by propagating the loss from final layer to initial layers and calculating gradients: derivative of loss with respect to weights and biases in each layer using chain rule. After gradients are calculated for final layer, its weights and biases are updated using an optimizer like

Stochastic gradient descent or Adam optimizer. Using chain rule, the gradients for layer other than final layer are calculated and parameters through the layers are updated.

The loss calculated is a quadratic function of our model's parameters. Our main objective is to find the values of weights and biases of our model where this loss is at it's minimum: global minimum. The PyTorch implementation of backward propagation using Adam optimizer is given below.

Backpropagation, gradient descent and softmax activation

In [206]:
```
# here, adam optimizer is used with a learning rate of 0.01

optimizer = torch.optim.Adam # Adam optimizer is used
optimizer = optimizer(layer.parameters(), lr = 0.01)
```

In [207]:
```
# Train for 10 epochs
for i in range(10):
    output = layer(a_3_channel_16by16_image)

    # calculating loss
    loss = CEL(targets, output)

# this statement below computes derivative of loss with respect to every parameter in our model whose requires_grad = True
    loss.backward(retain_graph=True)

# After gradients are calculated, the model's weights and biases are updated using gradient descent
# This operation is performed by the statement below
    optimizer.step()

# This statement removes all calculated gradients as they take up space. In each iteration, new gradients are to be calcula-
# -ted and after performing gradient descent, the space occupied by them needs to be released.
    optimizer.zero_grad()
```

In [208]:
```
output = layer(a_3_channel_16by16_image);output
```

Out[208]:
```
tensor([[ -7.8693, -20.8986, -15.8122]], grad_fn=<AddmmBackward0>)
```

In [209]:
```
# applying softmax activation to the outputs of our trained model

F.softmax(output, dim = 1) * 100
```

Out[209]:
```
tensor([[9.9964e+01, 2.1943e-04, 3.5506e-02]], grad_fn=<MulBackward0>)
```

In [210]:
```
9.9964e+01
```

Out[210]:
```
99.964
```

Recall that our target vector had [1., 0, 0] and predicted probability for first class is 99.9 %

Figure 105 Backpropagation in PyTorch

In the screenshot above, we calculated the cross-entropy loss for our input tensor. After calculating the loss in each epoch, we calculated the derivative of our loss with respect to the weights and biases of our model and performed gradient descent to update the model's parameters using Adam optimizer. By the end of 10^{th} epoch, our model's parameters are updated, and it makes the prediction for class one to be 0.99964 in terms of probability which is correct. This way, forward propagation, backward propagation, and optimization algorithm like Adam work hand in hand to train our model.

1.7 Some activation functions in CNN

The activation function, also known as the transfer function, maps the output of a model to a range of values like 0 to 1 or -1 to 1 either linearly or non-linearly. There are several activation functions that are used along with convolutional neural networks. Some of them are explained briefly below.

1.7.1 Sigmoid or Logistic activation function

This activation function's curve in a graph looks like an S-shape. This activation is primarily used to map the outputs of a binary classifier into a range of 0 to 1. It means that this activation is used to interpret the output of a model in terms of predicted probabilities of a positive class. It is a non-linear activation function. Its maximum value is 1 and minimum is 0. The derivative of a sigmoid function ranges from 0 to 0.25 (IBM, 2020).

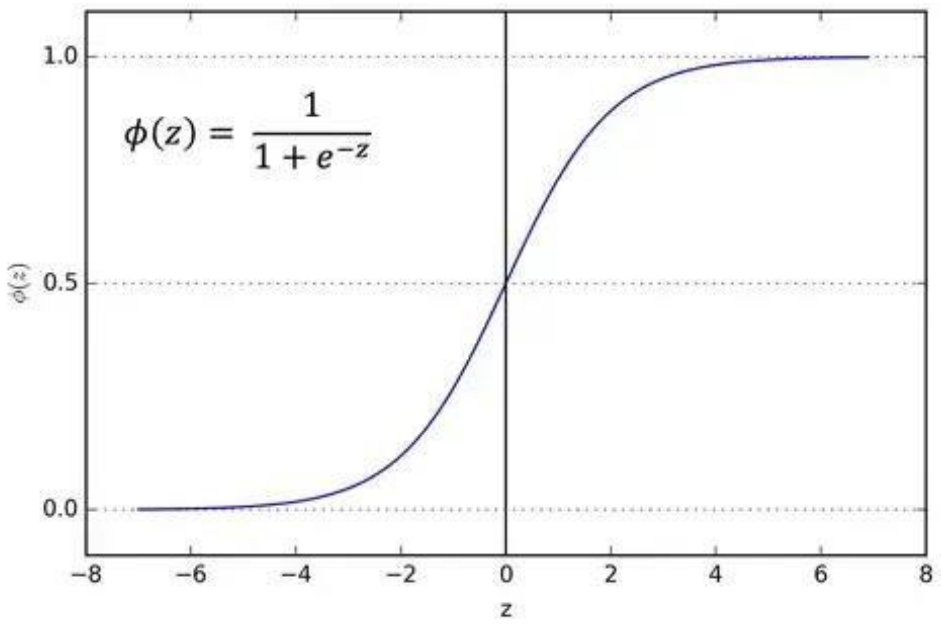


Figure 106 Sigmoid activation (Sagar Sharma)

1.7.2 Softmax activation function

The Softmax activation function is also a non-linear activation function that is used to map the output of a classifier in terms of predicted probabilities. This activation can used for both binary and multiclass classification, unlike the sigmoid activation that can only be used for binary classification. Its value also ranges from 0 to 1, as the sum of predicted probabilities of all classes is 1.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Figure 107 Softmax activation (Thomas Wood)

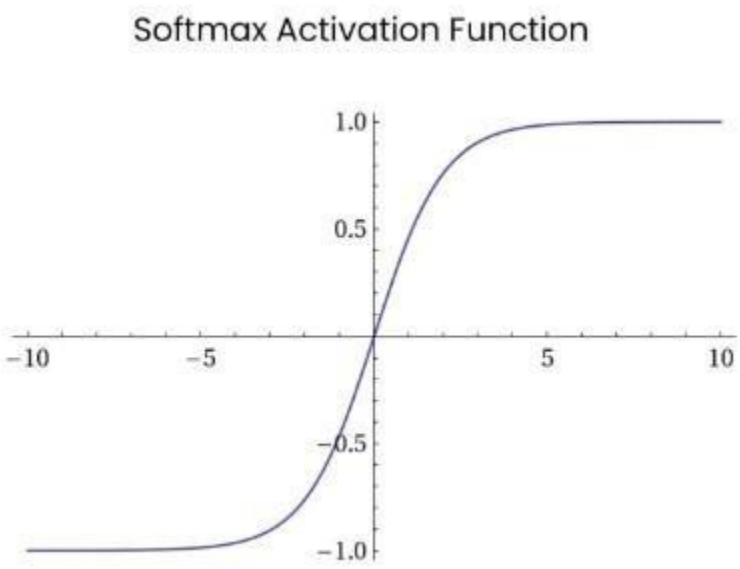


Figure 108 Softmax activation graph (Kajal Pawar)

1.7.3 Hyperbolic tangent activation function (Tanh)

It is a non-linear activation function that is used to map the output of a model to a range of -1 to 1. The Tanh activation has a similar curve (S-shaped) as compared to the logistic activation function. Both sigmoid and Tanh activation functions are used in feed forward process. The image below showcases the Tanh activation.

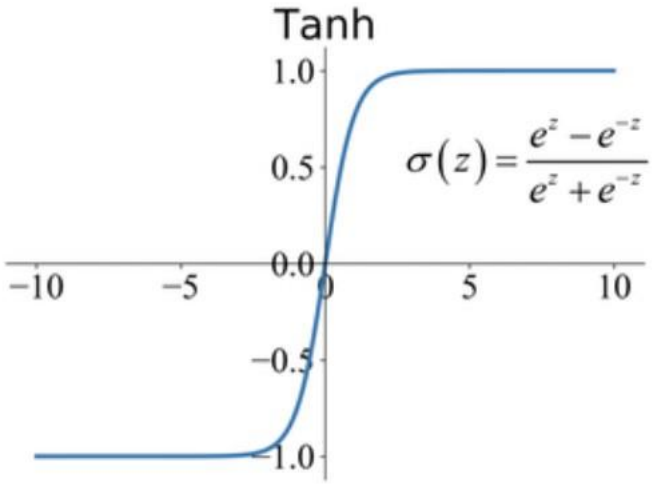


Figure 109 Tanh activation (Junxi Feng)

1.7.4 Rectified Linear Unit (ReLu) activation function

ReLu is also another type of non-linear activation function that maps negative value to 0. Its output ranges from 0 to infinity. There is inherently a problem with ReLu. It could hamper a model's training because all negative values are mapped as 0.

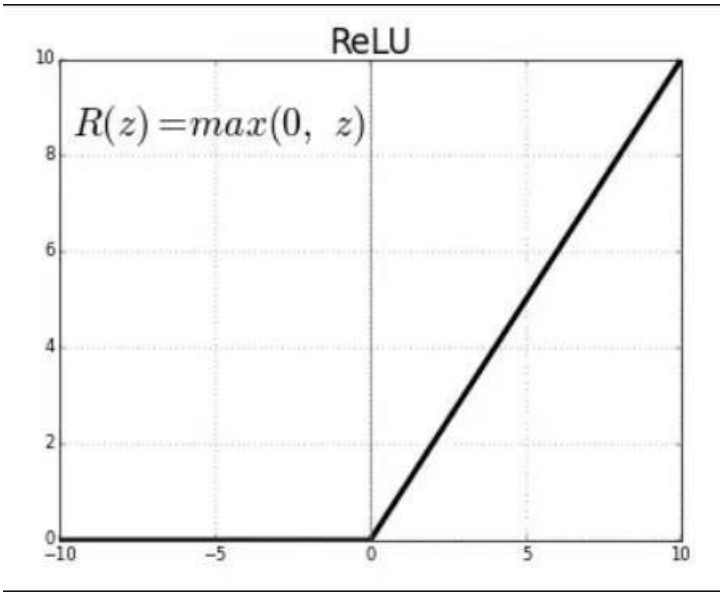


Figure 110 ReLu activation (Sagar Sharma)

1.7.5 Leaky ReLu

Leaky ReLu is another type of non-linear activation function that was mainly developed to address the issue of ReLu. Unlike ReLu that maps all negative values to 0, a very small slope parameter is used in Leaky ReLu that is used to incorporate some information from negative values (Turing, 2023).

$$R(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{otherwise} \end{cases}$$

Figure 111 Working of Leaky ReLu (Benjamin McCloskey)

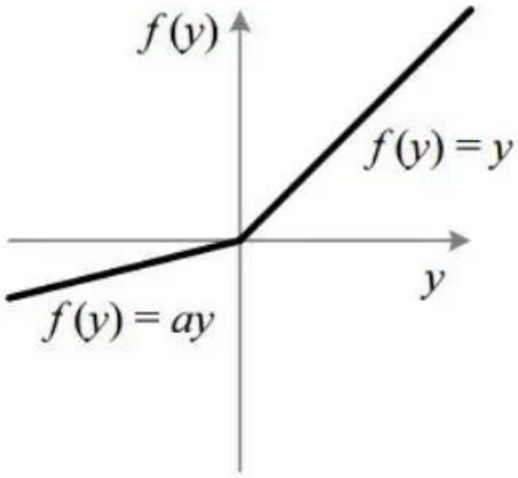


Figure 112 Leaky ReLu activation function (Sagar Sharma)

2. Generative Adversarial Networks

Generative Adversarial Networks, GANs, are a class of deep learning generative models that can mimic the distribution of original training data. GANs was originally introduced by Ian Goodfellow and colleagues in 2014. In Generative Adversarial Networks, two neural networks namely discriminator and generator contest each other in a zero-sum game, where the loss of one network is the gain of the other. GANs can reproduce the distribution of the initial training data so that they can be applied to several tasks, including the creation of images, image-to-image translation, and videos (Google, 2023).

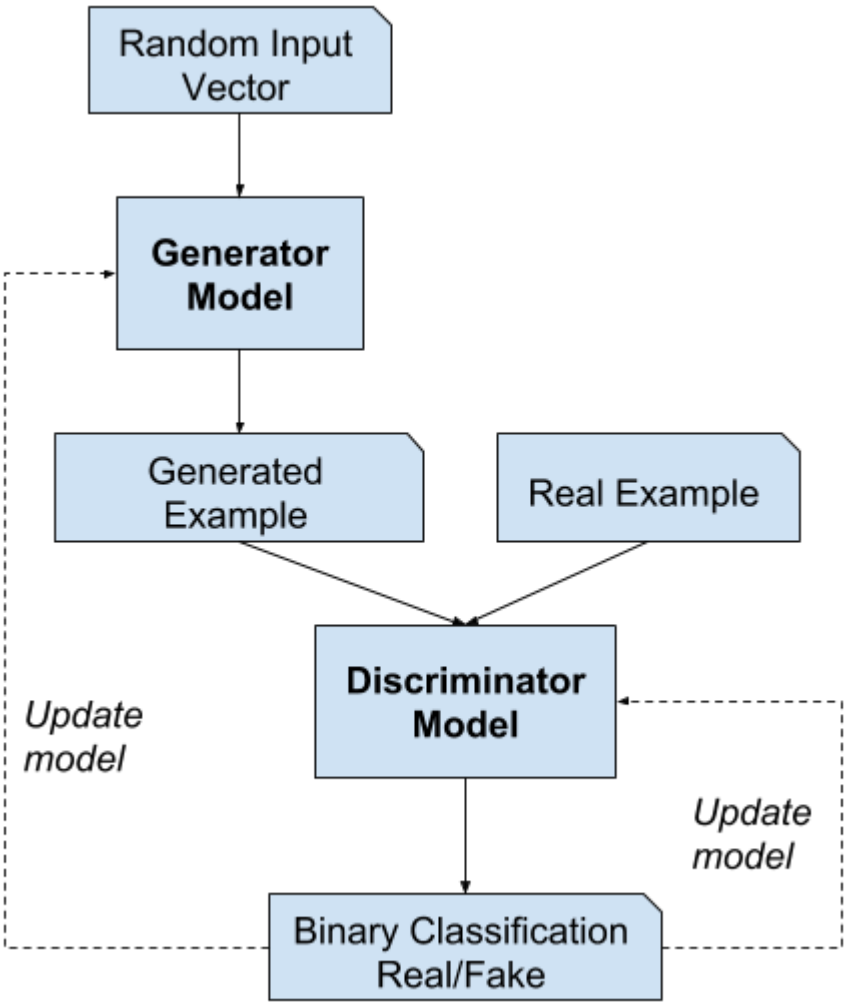


Figure 113 GANs working mechanism (Jason Brownlee)

2.1 Deconvolution or Transposed convolution

Three GAN architectures are implemented in this project: **DCGAN**, **WGAN** and **WGANGP**. All of these architectures implement transposed convolution for upsampling a random noise vector. Several transposed convolution layer, also known as deconvolution, are used in these architectures The main aim of a deconvolution layer is to generate a feature map whose spatial dimensions exceed than that of an input feature map and to carry out trainable upsampling (Deep Dive Into Deep Learning, 2023).The gif below showcases the working mechanism of a transposed convolution.

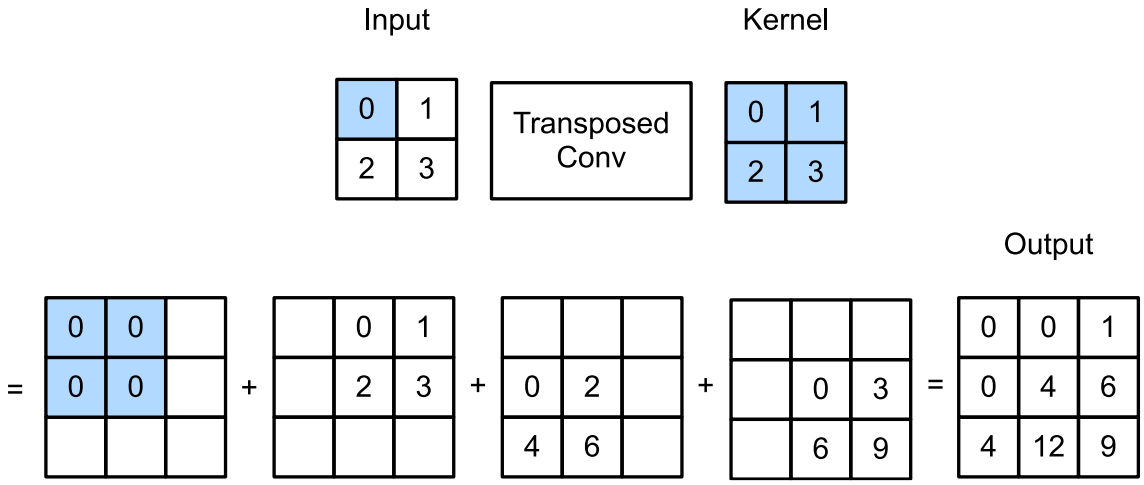


Figure 114 Transposed convolution intuition (Dive into deep learning)

Additionally, I have also performed transposed convolution in PyTorch which I will add below.

Deconvolution or Transposed convolution

```
In [81]:
# creating a custom tensor

tensor = torch.tensor(((0., 1), (2,3))).reshape(1,1,2,2);tensor

Out[81]: tensor([[[[0., 1.],
                   [2., 3.]]]])

In [82]:
# creating a custom kernel matrix

custom_kernel = torch.tensor([[0., 1],
                              [2, 3]], requires_grad  =True).reshape(1,1,2,2);custom_kernel

Out[82]: tensor([[[[0., 1.],
                   [2., 3.]]]], grad_fn=<ReshapeAliasBackward0>)
```

Figure 115 creating a custom tensor and kernel matrix

Here, I created a tensor with dimension 1 * 1 * 2 * 2. The first two dimensions account for the batch and number of channels. The last two dimensions are for rows and columns of the matrix. Similarly, to know that our transposed convolution operation is correct, I custom created a kernel or weights matrix which will be used to perform deconvolution. I have experimented with different values of padding and stride while performing deconvolution.

2.1.1 Stride and padding set to 1 and 0

With stride = 1 and padding = 0

```
In [83]:
# creating a transposed convolutional layer with a custom weight matrix/kernel

deconvolution_layer = nn.ConvTranspose2d(1, 1, kernel_size=2, stride=1, padding=0, bias=False)  # Stride = 1, padding = 0

with torch.no_grad():
    deconvolution_layer.weight = nn.Parameter(custom_kernel)

In [84]:
# performing deconvolution
deconvolution_layer(tensor)

Out[84]: tensor([[[[ 0.,  0.,  1.],
                   [ 0.,  4.,  6.],
                   [ 4., 12.,  9.]]]], grad_fn=<ConvolutionBackward0>)
```

Figure 116 Stride and padding set to 1 and 0 in deconvolution

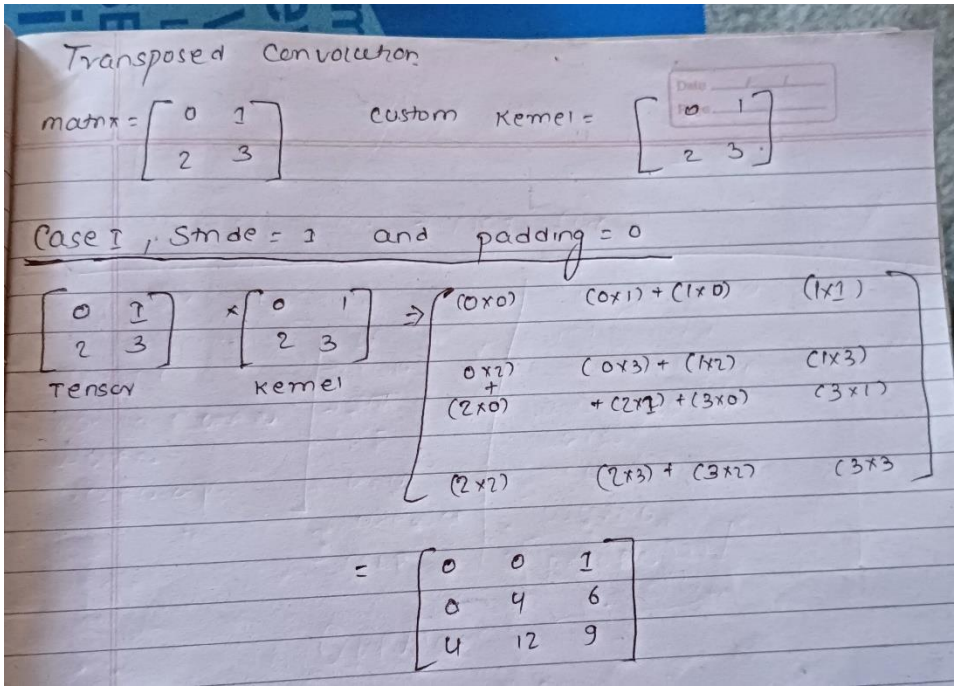


Figure 117 Stride and padding set to 1 and 0 in deconvolution in copy

2.1.2 Stride and padding set to 2 and 0

With stride = 2 and padding = 0

In [91]:
```
# creating a transposed convolutional layer with a custom weight matrix/kernel

deconvolution_layer = nn.ConvTranspose2d(1, 1, kernel_size=2, stride=2, padding=0, bias=False) # Stride = 2, padding = 0

with torch.no_grad():
    deconvolution_layer.weight = nn.Parameter(custom_kernel)
```

In [92]:
```
# performing deconvolution
deconvolution_layer(tensor)
```

Out[92]:
```
tensor([[[[0., 0., 1.],
          [0., 0., 2., 3.],
          [0., 2., 0., 3.],
          [4., 6., 6., 9.]]]], grad_fn=<ConvolutionBackward0>)
```

Increasing stride any more will result in the same matrix, as there is no more space to slide the kernel

Figure 118 Stride and padding set to 2 and 0 in deconvolution

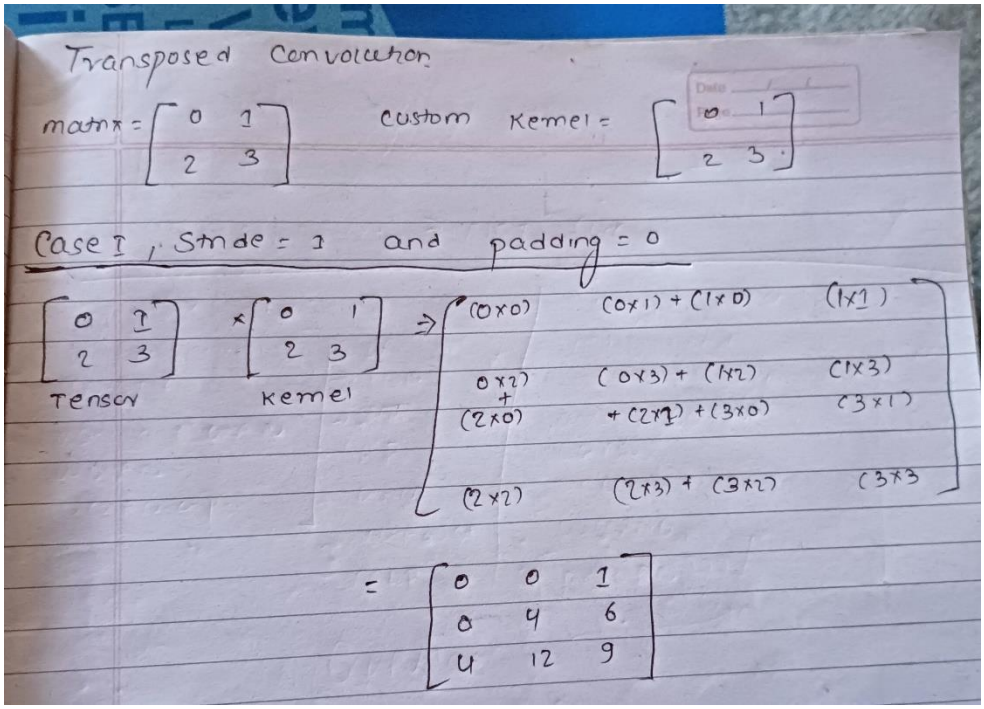


Figure 119 Stride and padding set to 2 and 0 in deconvolution in copy

2.1.3 Stride and padding set to 3 and 0

Increasing the stride any more will result in a larger matrix but the values inside the matrix will be the same. The only difference is that the odd rows and columns added will have 0 values.

Increasing stride any more will result in the same numbers but in additinal odd spaces, values will be appended.

```
In [121]:
# performing deconvolution
deconvolution_layer(tensor)  # STRIDE = 3

Out[121]: tensor([[[[0., 0., 0., 0., 1.],
                    [0., 0., 0., 2., 3.],
                    [0., 0., 0., 0., 0.],
                    [0., 2., 0., 0., 3.],
                    [4., 6., 0., 6., 9.]]]], grad_fn=<ConvolutionBackward0>)
```

Figure 120 Stride and padding set to 3 and 0 in deconvolution

2.1.4 Stride and padding set to 1

In convolution, padding is applied to the input whereas, in deconvolution, padding is applied to the output

For example, when specifying the padding number on either side of the height and width as 1, the first and last rows and columns will be removed from the transposed convolution output.

```
In [107]:
def deconvolution(tensor, stride, padding):
    print("STRIDE: ", stride)
    print("PADDING: ", padding)
    # creating a transposed convolutional layer with a custom weight matrix/kernel
    deconvolution_layer = nn.ConvTranspose2d(1, 1, kernel_size=2, stride=stride, \
                                             padding= padding, bias=False)

    with torch.no_grad():
        deconvolution_layer.weight = nn.Parameter(custom_kernel)

    return deconvolution_layer(tensor)

In [110]:
# performing deconvolution
deconvolution(tensor, 1, 0)

STRIDE:  1
PADDING:  0

Out[110]: tensor([[[[ 0.,  0.,  1.],
                    [ 0.,  4.,  6.],
                    [ 4., 12.,  9.]]]], grad_fn=<ConvolutionBackward0>)

In [109]:
# performing deconvolution
deconvolution(tensor, 1, 1)

STRIDE:  1
PADDING:  1

Out[109]: tensor([[[[4.]]]], grad_fn=<ConvolutionBackward0>)
```

Since padding = 1, the first and last rows and columns of the output is removed

Figure 121 Stride and padding set to 1 in deconvolution

In padding, the n first and last rows and columns of an output are removed.

2.1.5 Stride and padding set to 2

With stride = 2 and padding = 2

```
In [112]:
1 # performing deconvolution with stride 2 and padding 0
2 deconvolution(tensor, 2, 0)

STRIDE:  2
PADDING:  0

Out[112]: tensor([[[[0., 0., 0., 1.],
                    [0., 0., 2., 3.],
                    [0., 2., 0., 3.],
                    [4., 6., 6., 9.]]]], grad_fn=<ConvolutionBackward0>)
```

```
In [113]:
1 # performing deconvolution with stride 2 and padding 1
2 deconvolution(tensor, 2, 1)

STRIDE:  2
PADDING:  1

Out[113]: tensor([[[[0., 2.],
                    [2., 0.]]]], grad_fn=<ConvolutionBackward0>)
```

Figure 122 Stride and padding set to 2 in deconvolution

2.2 DCGAN

DCGAN stands for deep convolutional generative adversarial network. DCGAN uses convolution and transposed convolutions-based layers in the discriminator and generator, as opposed to the multilayer perceptron used by the vanilla GAN. In the discriminator's architecture, Leaky ReLu activation is used to ensure that gradients actively pass through the architecture during generator training. The following images below highlight the architecture of the generator and discriminator in DCGAN.

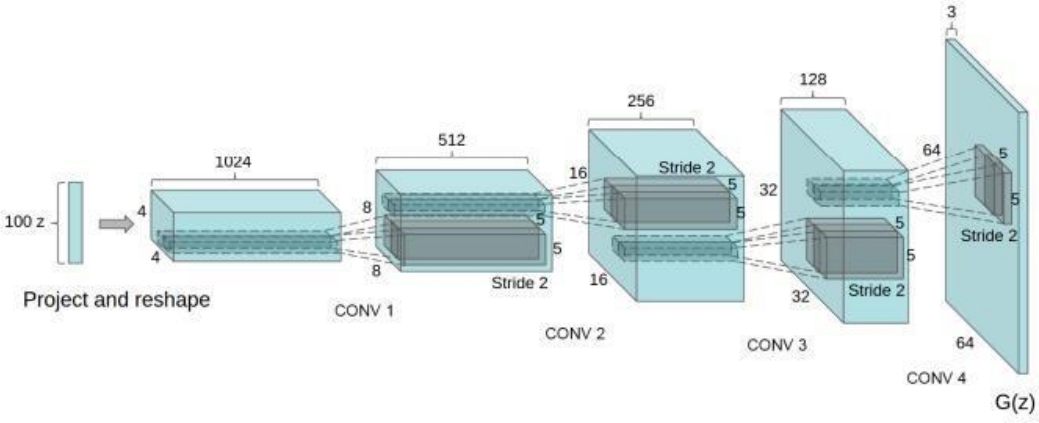


Figure 123 DCGAN Generator architecture (Original Paper)

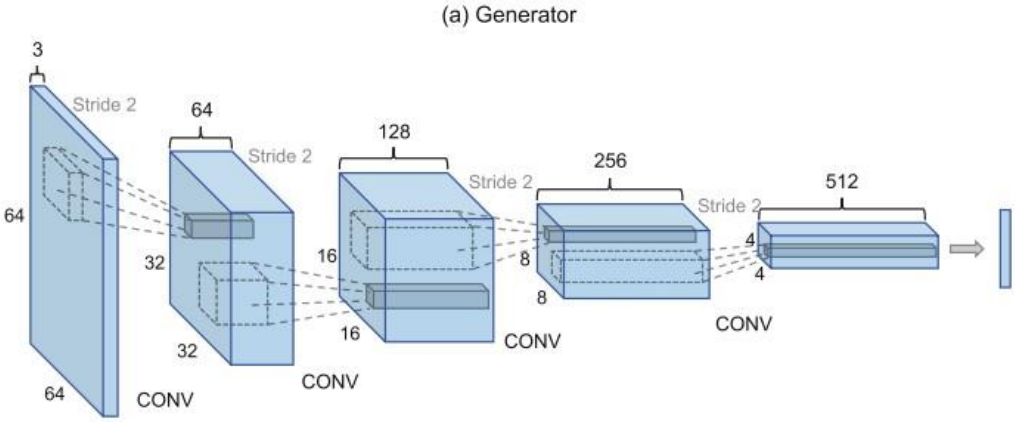


Figure 124 Generator architecture (Original Paper)

Additionally, I have also added the summary of the DCGAN's generator and discriminator models that I implemented in PyTorch.

```
In [18]:  1  summary(discriminator, (3, 64, 64))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           3,072
       BatchNorm2d-2           [-1, 64, 32, 32]             128
         LeakyReLU-3           [-1, 64, 32, 32]               0
            Conv2d-4          [-1, 128, 16, 16]         131,072
       BatchNorm2d-5          [-1, 128, 16, 16]             256
         LeakyReLU-6          [-1, 128, 16, 16]               0
            Conv2d-7            [-1, 256, 8, 8]         524,288
       BatchNorm2d-8            [-1, 256, 8, 8]             512
         LeakyReLU-9            [-1, 256, 8, 8]               0
           Conv2d-10            [-1, 512, 4, 4]       2,097,152
      BatchNorm2d-11            [-1, 512, 4, 4]           1,024
        LeakyReLU-12            [-1, 512, 4, 4]               0
           Conv2d-13              [-1, 1, 1, 1]           8,192
          Flatten-14                    [-1, 1]               0
          Sigmoid-15                    [-1, 1]               0
================================================================
Total params: 2,765,696
Trainable params: 2,765,696
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.05
Forward/backward pass size (MB): 2.81
Params size (MB): 10.55
Estimated Total Size (MB): 13.41
----------------------------------------------------------------
```

Figure 125 Summary of discriminator model

```
In [24]:  1  summary(generator, (102, 1, 1))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 512, 4, 4]         835,584
       BatchNorm2d-2            [-1, 512, 4, 4]           1,024
              ReLU-3            [-1, 512, 4, 4]               0
   ConvTranspose2d-4            [-1, 256, 8, 8]       2,097,152
       BatchNorm2d-5            [-1, 256, 8, 8]             512
              ReLU-6            [-1, 256, 8, 8]               0
   ConvTranspose2d-7          [-1, 128, 16, 16]         524,288
       BatchNorm2d-8          [-1, 128, 16, 16]             256
              ReLU-9          [-1, 128, 16, 16]               0
  ConvTranspose2d-10           [-1, 64, 32, 32]         131,072
      BatchNorm2d-11           [-1, 64, 32, 32]             128
             ReLU-12           [-1, 64, 32, 32]               0
  ConvTranspose2d-13            [-1, 1, 64, 64]           1,024
             Tanh-14            [-1, 1, 64, 64]               0
================================================================
Total params: 3,591,040
Trainable params: 3,591,040
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 2.88
Params size (MB): 13.70
Estimated Total Size (MB): 16.57
----------------------------------------------------------------
```

Figure 126 DCGAN Generator summary

2.3 WGAN and WGAN-GP

The DCGAN model mentioned above has issues such as mode collapse and vanishing gradient. These issues can make DCGAN training difficult and unsteady. Model collapse is caused mostly by the generator producing the same output all of the time. As the discriminator improves but is unable to properly identify images generated by the generator, this feedback is passed on to the generator, resulting in the generator producing the same images that the discriminator was unable to distinguish as real or false to trick the discriminator. After some training, the discriminator model improves and can classify those repeated fake images, breaking free from the local minimum of its loss function. This could lead to one of two outcomes. The generator will now be unable to diversify its generation because the discriminator will catch its bluff every time it generates a similar image, stopping the generator from improving. Alternatively, the generator can migrate to a different mode of distribution and collapse to a different mode. This is known as mode collapse, and it effectively results in no training. In other words, the training is terminated due to the mode collapse problem (IBM, 2022). Furthermore, DCGAN employs BCE loss, or binary cross-entropy, as its loss function. To categorize real and fake images, the discriminator employs sigmoid activation in its last layer to compress the result into a range of 0 to 1. As the discriminator improves throughout training, the output probabilities will approach 0 and 1. As the discriminator's projected probability approaches 0 and 1, the gradients will continue to shrink until they are very near 0. Such small gradients are quite detrimental to the generator. As a result, the discriminator model in DCGAN quickly outperforms the generator, resulting in no generator training. This is known as the vanishing gradient problem. It is caused primarily by sigmoid activation function and BCE loss.

To prevent mode collapse and vanishing gradient problem, WGAN introduces two new concepts while training GANs: Wasserstein loss and weight clipping. Instead of squashing the output value into 0 and 1 through the sigmoid function, the

discriminator simply can return any real value. Hence, the discriminator is called a critic in this type of GAN as it is not classifying between real and fake but simply returning a real value for a given image. Here, W-Loss implements Earth Mover's distance which measures the distance between the probability distributions of real and fake images. It means that the discriminator is trying to maximize this loss as it wants to successfully differentiate real and fake images. Meanwhile, the generator is trying to minimize this loss as it wants the distributions to be as close as possible to fool the discriminator.

For training GANs with this new loss function, the critic needs to meet a special condition called 1-Lipschitz (1-L) continuity. For the critic to be 1-L continuous, its slope or gradient must be at most one at any point. This condition will make sure that the W-Loss calculated is validly approximating the Earth Mover's distance. There are two different methods to enforce 1-L continuity in critic: Weight clipping and gradient penalty. Weight clipping will force weights to a fixed interval of the critic. By limiting the values of weights of the critic, the critic can't take on many different values for weights. Its ability to perform well can also be limited. For implementing gradient penalty, we can simply add a regularization term to our W-Loss. It will penalize the critic if its gradient is higher than one effectively ensuring 1-L continuity. WGAN implements weight clipping whereas WGAN-GP implements gradient penalty. Weight clipping is a technique to keep the Lipschitz constant of the critic in WGAN under control. It means that all parameters of the critic are limited to a specific range. In the WGAN implemented in this project, the parameters of the critic are clipped to a range of [-0.01, 0.01]. Gradient penalty is another technique to enforce Lipschitz constant of the critic in WGAN-GP by enforcing the condition that the gradients of the critic's output with respect to the inputs will always have unit norm. Gradient penalty is a better technique to enforce 1 Lipchitz constant on critic than weight clipping because weight clipping limits the learning ability of the critic (Donghwan , et al., 2022).

2.4 Training GANs with conditional generation

The WGAN and WGAN-GP use the same generator and discriminator architecture as DGCAN. The only changes will be made to the loss function to solve mode collapse and vanishing gradient problem. Additionally, all three GAN architectures are Conditional GANs. This means the generator can generate images of specific classes. While training GANs, the discriminator is fed an image with its labels concatenated to original image as different channels such that all values of those channels will be 0 if the image does not belong to that class else 1. Similarly, class information as one hot encoded vector is concatenated to the noise vector which is passed into the generator while training.

In [41]:

```python
def train_discriminator(real_images, opt_d, cur_batch_size, real_labels):
    # Clear discriminator gradients
    opt_d.zero_grad()

    ''' Getting fake images from generator for corresponding labels: real_labels'''
    one_hot_labels = F.one_hot(real_labels.to(device), n_classes)
    image_one_hot_labels = one_hot_labels[:, :, None, None]
    image_one_hot_labels = image_one_hot_labels.repeat(1, 1, image_size, image_size)

    # generate fake images
    noise = torch.randn(cur_batch_size, latent_size, device=device)
    noise_and_labels = torch.cat([noise, one_hot_labels.float()],dim=1)
    noise_and_labels = noise_and_labels.view(len(noise_and_labels), generator_input_dim, 1, 1)
    fake_images = generator(noise_and_labels)
    # Pass real images through discriminator



    ''' getting putputs from generator by passing real and fake images correspondingly '''
    fake_image_and_labels = torch.cat([fake_images,image_one_hot_labels],dim=1)
    real_image_and_labels = torch.cat([real_images,image_one_hot_labels],dim=1)

    # passing real images through discriminator
    real_preds = discriminator(real_image_and_labels).reshape(-1)

    # pass fake images through discriminator
    fake_preds = discriminator(fake_image_and_labels).reshape(-1)

    loss = -(torch.mean(real_preds) - torch.mean(fake_preds))

    loss.backward()
    opt_d.step()


    # WEIGHT CLIPPING
    # clip critic weights between -0.01, 0.01
    for p in discriminator.parameters():
        p.data.clamp_(-WEIGHT_CLIP, WEIGHT_CLIP)
    return loss.item()
```

Figure 127 WGAN discriminator conditional training

In [42]:

```python
def train_generator(opt_g, cur_batch_size, real_labels):
    # Clear generator gradients
    opt_g.zero_grad()

    ''' Getting fake images from generator for corresponding labels: real_labels'''
    one_hot_labels = F.one_hot(real_labels.to(device), n_classes)
    image_one_hot_labels = one_hot_labels[:, :, None, None]
    image_one_hot_labels = image_one_hot_labels.repeat(1, 1, image_size, image_size)

    # generate fake images
    noise = torch.randn(cur_batch_size, latent_size, device=device)
    noise_and_labels = torch.cat([noise, one_hot_labels.float()],dim=1)
    noise_and_labels = noise_and_labels.view(len(noise_and_labels), generator_input_dim, 1, 1)
    fake_images = generator(noise_and_labels)


    ''' getting putputs from generator by passing real and fake images correspondingly '''
     # Try to fool the discriminator
    fake_image_and_labels = torch.cat([fake_images,image_one_hot_labels],dim=1)
    preds = discriminator(fake_image_and_labels).reshape(-1)


    loss = -torch.mean(preds)

    # Update generator weights
    loss.backward()
    opt_g.step()

    return loss.item()
```

Figure 128 WGAN generator conditional training