Text

Description automatically generated

Project and Professionalism

(6CS007)

**MedicareAI**

Student Id : 2058939

College Id : np03cs4s210019

Student Name : Sujan Neupane

Supervisor : Mr. Dinesh Saud

Reader : Mr. Siman Giri  
Module Leader : Mr. Biraj Dulal

Submitted on :

**Title and Declaration sheet**

# 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**

[1. Introduction 1](#_Toc130238323)

[1.1 Project briefing 1](#_Toc130238324)

[1.1. 1 Project Introduction 1](#_Toc130238325)

[1.1.2 Problem Statement 3](#_Toc130238326)

[1.1.3 The Project as a Solution 4](#_Toc130238327)

[1.1.4 Aspects of AI 5](#_Toc130238328)

[1.2 Aims 9](#_Toc130238329)

[1.3 Objectives 9](#_Toc130238330)

[1.4 Artefact 10](#_Toc130238331)

[1.5 Academic Questions 12](#_Toc130238332)

[1.7 Scope and Limitation of the project 13](#_Toc130238333)

[1.8 Report Structure 14](#_Toc130238334)

[2. Literature Review 15](#_Toc130238335)

[2.1 Research Papers 15](#_Toc130238336)

[2.1.1 Evaluation of Deep Convolutional Generative Adversarial Networks for Data Augmentation of Chest X-ray Images 15](#_Toc130238337)

[2.1.2 Comparison of Deep Learning Models AlexNet and GoogLeNet in Detection of Pneumonia and Covid19 17](#_Toc130238338)

[2.1.3 CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection 18](#_Toc130238339)

[2.1.4 Enhancing Automated COVID-19 Chest X-ray Diagnosis by Image-to-Image GAN Translation 20](#_Toc130238340)

[2.1.5 Wasserstein GAN based Chest X-Ray Dataset Augmentation for Deep Learning Models: COVID-19 Detection Use-Case 22](#_Toc130238341)

[2.1.6 Automated COVID-19 diagnosis using Deep Multiple Instance Learning with CycleGAN 25](#_Toc130238342)

[2.1.7 A Survey on Deep Learning Advances and Emerging Issues in Pneumonia and COVID19 Prediction 27](#_Toc130238343)

[2.2 Convolutional Neural Networks and Generative Adversarial Networks 29](#_Toc130238344)

[2.2.1 Convolutional layer 29](#_Toc130238345)

[2.2.2 Stride and Padding 34](#_Toc130238346)

[2.2.3 Pooling Layer 35](#_Toc130238347)

[2.2.4 Normalization Layer 37](#_Toc130238348)

[2.2.5 Fully Connected Layer 39](#_Toc130238349)

[2.2.6 Forward pass, backward pass and Adam optimizer in CNN 40](#_Toc130238350)

[2.2.7 Some activation functions in CNN 43](#_Toc130238351)

[2.3 Generative Adversarial Networks 48](#_Toc130238352)

[2.3.1 Deconvolution or Transposed convolution 49](#_Toc130238353)

[2.3.2 DCGAN 54](#_Toc130238354)

[2.3.3 WGAN and WGAN-GP 56](#_Toc130238355)

[2.3.4 Training GANs with conditional generation 58](#_Toc130238356)

[3. Project Methodology 60](#_Toc130238357)

[References 61](#_Toc130238358)

**Table of Figures**

[Figure 1 64 random covid positive x-ray samples 1](#_Toc130238419)

[Figure 2 64 random normal x-ray samples 2](#_Toc130238420)

[Figure 3 64 random viral pneumonia samples 2](#_Toc130238421)

[Figure 4 A pie chart showcasing the class imbalance problem 3](#_Toc130238422)

[Figure 5 Class imbalance solved using GANs 4](#_Toc130238423)

[Figure 6 EfficientNet architecture (Google) 6](#_Toc130238424)

[Figure 7 ResNet-50 architecture (Raimi Karim) 7](#_Toc130238425)

[Figure 8 GoogLeNet architecture (Geeks for Geeks) 7](#_Toc130238426)

[Figure 9 Figure 37 DCGAN Generator architecture (Original Paper) 8](#_Toc130238427)

[Figure 10 Generator architecture (Original Paper) 8](#_Toc130238428)

[Figure 11 Functional decomposition diagram for MedicareAI 10](#_Toc130238429)

[Figure 12 Generator Architecture (Original Paper) 16](#_Toc130238430)

[Figure 13 CNN classifier's performance (Original Paper) 17](#_Toc130238431)

[Figure 14 Working mechanism of CovidGAN (Original paper) 19](#_Toc130238432)

[Figure 15 Performance of VGG15 model (Original paper) 19](#_Toc130238433)

[Figure 16 Performance of ResNet50 model (Original paper) 21](#_Toc130238434)

[Figure 17 WGAN discriminator model architecture (original paper) 23](#_Toc130238435)

[Figure 18 WGAN generator model architecture (original paper) 24](#_Toc130238436)

[Figure 19 Working mechanism of CycleGAN (Original paper) 26](#_Toc130238437)

[Figure 20 Applications of chest x-rays using deep learning (Original paper) 28](#_Toc130238438)

[Figure 21 Convolution operation (Irhum Shafkat) 30](#_Toc130238439)

[Figure 22 A 3\*4\*4 RGB image (Sumit Saha) 30](#_Toc130238440)

[Figure 23 Convolution operation in RGB image (Niklas Lang) 31](#_Toc130238441)

[Figure 24 Creating a 3\*5\*5 array 32](#_Toc130238442)

[Figure 25 Two convolution layers 32](#_Toc130238443)

[Figure 26 Output from convolutional layers 33](#_Toc130238444)

[Figure 27 Padding in CNN (Abhisek Kumar Pandey) 34](#_Toc130238445)

[Figure 28 Stride in CNN (Abhisek Kumar Pandey) 34](#_Toc130238446)

[Figure 29 Types of pooling (Sumit Saha) 35](#_Toc130238447)

[Figure 30 Average and Max pooling in PyTorch 36](#_Toc130238448)

[Figure 31 Minmax scaling (Martin Riva) 37](#_Toc130238449)

[Figure 32 Standard Scaling (Martin Riva) 38](#_Toc130238450)

[Figure 33 Batch Normalization formula (Martin Riva) 38](#_Toc130238451)

[Figure 34 Batch Normalization in PyTorch 39](#_Toc130238452)

[Figure 35 Fully connected layer in PyTorch example 39](#_Toc130238453)

[Figure 36 Forward pass in CNN 40](#_Toc130238454)

[Figure 37 Output from a CNN 41](#_Toc130238455)

[Figure 38 Cross entropy loss calculation 41](#_Toc130238456)

[Figure 39 Backpropagation in PyTorch 42](#_Toc130238457)

[Figure 40 Sigmoid activation (Sagar Sharma) 43](#_Toc130238458)

[Figure 41 Softmax activation (Thomas Wood) 44](#_Toc130238459)

[Figure 42 Softmax activation graph (Kajal Pawar) 44](#_Toc130238460)

[Figure 43 Tanh activation (Junxi Feng) 45](#_Toc130238461)

[Figure 44 ReLu activation (Sagar Sharma) 46](#_Toc130238462)

[Figure 45 Working of Leaky ReLu (Benjamin McCloskey) 46](#_Toc130238463)

[Figure 46 Leaky ReLu activation function(Sagar Sharma) 47](#_Toc130238464)

[Figure 47 GANs working mechanism (Jason Brownlee) 48](#_Toc130238465)

[Figure 48 Transposed convolution intuition (Dive into deep learning) 49](#_Toc130238466)

[Figure 49 creating a custom tensor and kernel matrix 50](#_Toc130238467)

[Figure 50 Stride and padding set to 1 and 0 in deconvolution 50](#_Toc130238468)

[Figure 51 Stride and padding set to 1 and 0 in deconvolution in copy 51](#_Toc130238469)

[Figure 52 Stride and padding set to 2 and 0 in deconvolution 51](#_Toc130238470)

[Figure 53 Stride and padding set to 2 and 0 in deconvolution in copy 52](#_Toc130238471)

[Figure 54 Stride and padding set to 3 and 0 in deconvolution 52](#_Toc130238472)

[Figure 55 Stride and padding set to 1 in deconvolution 53](#_Toc130238473)

[Figure 56 Stride and padding set to 2 in deconvolution 53](#_Toc130238474)

[Figure 57 DCGAN Generator architecture (Original Paper) 54](#_Toc130238475)

[Figure 58 Generator architecture (Original Paper) 54](#_Toc130238476)

[Figure 59 Summary of discriminator model 55](#_Toc130238477)

[Figure 60 DCGAN Generator summary 56](#_Toc130238478)

[Figure 61 WGAN discriminator conditional training 59](#_Toc130238479)

[Figure 62 WGAN generator conditional training 59](#_Toc130238480)

**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**](https://www.kaggle.com/datasets/tawsifurrahman/covid19-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

A picture containing curtain, indoor, furniture, different

Description automatically generated

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.

Chart, pie chart

Description automatically generated

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.

Chart, bar chart

Description automatically generated

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 WGAN-GP - 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 is capable of tolerating 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 a 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).

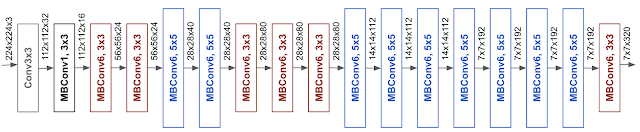


Figure 6 EfficientNet architecture ([Google](https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html))

#### 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.

Diagram

Description automatically generated

Figure 7 ResNet-50 architecture ([Raimi Karim](https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d))

#### 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.

Chart

Description automatically generated

Figure 8 GoogLeNet architecture ([Geeks for Geeks](https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/))

#### 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.

Diagram

Description automatically generated

Figure 9 Figure 37 DCGAN Generator architecture ([Original Paper](https://arxiv.org/pdf/1511.06434v2.pdf))

Diagram

Description automatically generated

Figure 10 Generator architecture ([Original Paper](https://doi.org/10.1016/j.knosys.2019.104927))

The DCGAN architecture suffers from the 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

1. Quality of synthetic image data generated will be properly accessed using **Fréchet inception distance (FID).**
2. Covid-19 diagnosis will be boosted and automated by implementing various classification algorithms like ResNet 50, EfficientNet and GoogLeNet.
3. A web application using Django, HTML, CSS, and JavaScript will be developed to for making real time predictions.

## 1.4 Artefact

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

Diagram

Description automatically generated

Figure 11 Functional decomposition diagram for MedicareAI

MedicareAI is a web application developed to enhance COVID-19 diagnosis through the use of 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 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. The project MedicareAI has been created as an effort of speeding up covid testing.

## 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 WGAN-GP.
* 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

# 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.

Diagram

Description automatically generated

Figure 12 Generator Architecture (Original Paper)

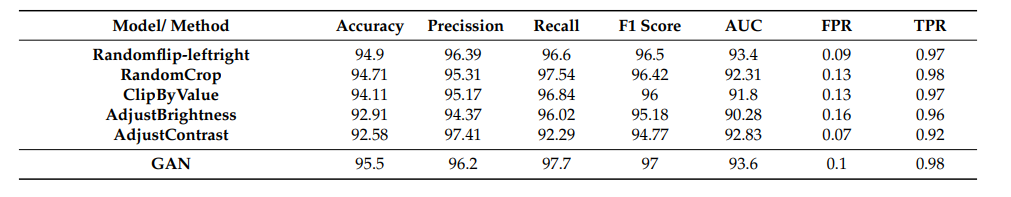


Figure 13 CNN classifier's performance ([Original Paper](https://www.mdpi.com/1999-5903/13/1/8))

### 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.

Diagram

Description automatically generated

Figure 14 Working mechanism of CovidGAN [(Original paper)](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9093842)

Table

Description automatically generated

Figure 15 Performance of VGG15 model [(Original paper)](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9093842)

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 covid-19 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 pre-trained 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.

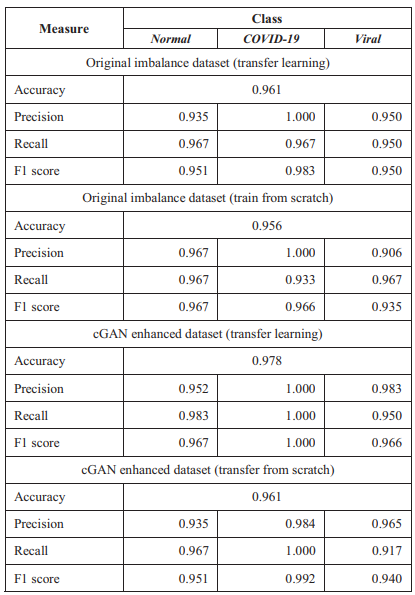


Figure 16 Performance of ResNet50 model [(Original paper)](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9313466)

This literature implements 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).

Diagram

Description automatically generated

Figure 17 WGAN discriminator model architecture [(original paper)](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9871519)

Diagram, text

Description automatically generated

Figure 18 WGAN generator model architecture [(original paper)](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9871519)

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 covid-19 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).

Following image below showcases the CycleGAN architecture implemented in this literature.

Diagram

Description automatically generated

Figure 19 Working mechanism of CycleGAN [(Original paper)](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9761334)

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).

Diagram

Description automatically generated

Figure 20 Applications of chest x-rays using deep learning [(Original paper)](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9736485)

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 Convolutional Neural Networks and Generative Adversarial 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.

### 2.2.1 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).

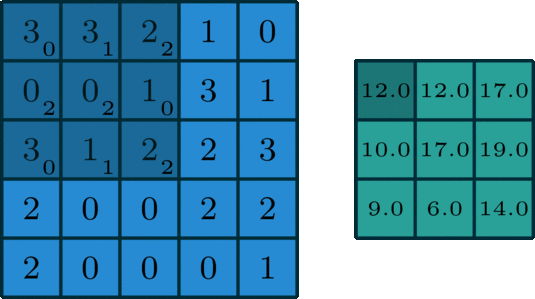


Figure 21 Convolution operation [(Irhum Shafkat](https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1/))

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 a following RGB image.

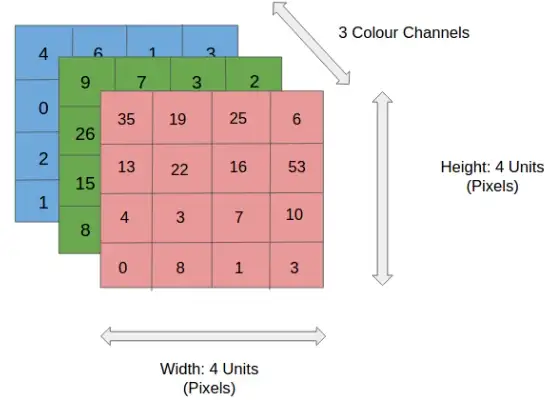


Figure 22 A 3\*4\*4 RGB image [(Sumit Saha)](https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)

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 as a image. It is rather called as a **feature map**.

Diagram

Description automatically generated

Figure 23 Convolution operation in RGB image ([Niklas Lang](https://towardsdatascience.com/using-convolutional-neural-network-for-image-classification-5997bfd0ede4))

A complete convolution operation is explained below using PyTorch.

Graphical user interface, text, application

Description automatically generated with medium confidence

Figure 24 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.

Graphical user interface, text, application

Description automatically generated

Figure 25 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.

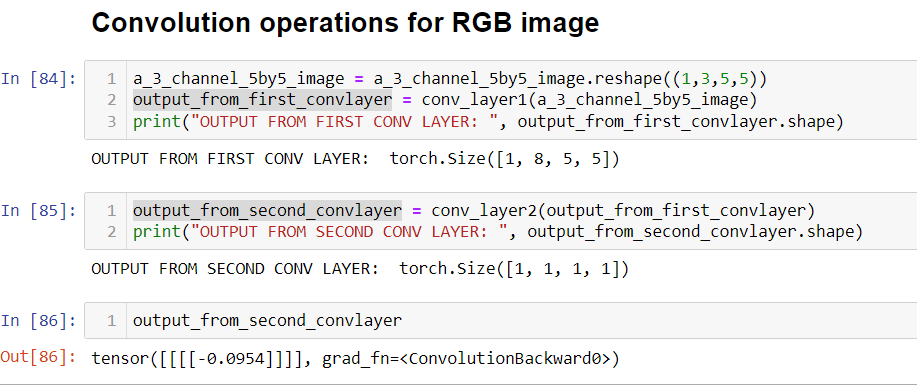


Figure 26 Output from convolutional layers

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.

### 2.2.2 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.

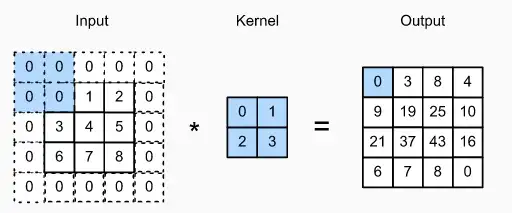


Figure 27 Padding in CNN [(Abhisek Kumar Pandey)](https://medium.com/analytics-vidhya/convolution-padding-stride-and-pooling-in-cnn-13dc1f3ada26)

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

A picture containing shoji, building

Description automatically generated

Figure 28 Stride in CNN [(Abhisek Kumar Pandey)](https://medium.com/analytics-vidhya/convolution-padding-stride-and-pooling-in-cnn-13dc1f3ada26)

### 2.2.3 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).

Diagram

Description automatically generated

Figure 29 Types of pooling [(Sumit Saha)](https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)

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

Graphical user interface, text, application, email

Description automatically generated

Figure 30 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.

### 2.2.4 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 value range of [0,1] respectively.

A picture containing text

Description automatically generated

Figure 31 Minmax scaling [(Martin Riva)](https://www.baeldung.com/cs/batch-normalization-cnn#:~:text=Normalization%20is%20a%20pre%2Dprocessing,and%20decrease%20its%20learning%20speed.)

Here, x refers to the sample in a feature, m refers to the mean of the feature, xmax and xmin 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)**.**

Text, letter

Description automatically generated

Figure 32 Standard Scaling [(Martin Riva)](https://www.baeldung.com/cs/batch-normalization-cnn#:~:text=Normalization%20is%20a%20pre%2Dprocessing,and%20decrease%20its%20learning%20speed.)

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 batches of data, the tensors are normalized using following formula.

Diagram

Description automatically generated

Figure 33 Batch Normalization formula [(Martin Riva)](https://www.baeldung.com/cs/batch-normalization-cnn#:~:text=Normalization%20is%20a%20pre%2Dprocessing,and%20decrease%20its%20learning%20speed.)

Here, mz and sz refer to batch mean and standard deviation.

Batch Normalization helps to speed up the training and help model 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.

Graphical user interface, text, application

Description automatically generated

Figure 34 Batch Normalization in PyTorch

### 2.2.5 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.

Graphical user interface, text, application

Description automatically generated

Figure 35 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.

### 2.2.6 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.

Text

Description automatically generated

Figure 36 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.

Graphical user interface, text, application, Word

Description automatically generated

Figure 37 Output from a CNN

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

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 38 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.

Graphical user interface, text, application, email

Description automatically generated

Figure 39 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 10th 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.

### 2.2.7 Some activation functions in CNN

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.

#### 2.2.7.1 Sigmoid or Logistic activation function

This activation function’s curve in a graph looks like 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 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).

Diagram

Description automatically generated

Figure 40 Sigmoid activation [(Sagar Sharma)](https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)

#### 2.2.7.2 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.

Text

Description automatically generated with medium confidence

Figure 41 Softmax activation [(Thomas Wood)](https://deepai.org/machine-learning-glossary-and-terms/softmax-layer)

Diagram

Description automatically generated

Figure 42 Softmax activation graph [(Kajal Pawar)](https://insideaiml.com/blog/SoftMaxActivation-Function-1034)

#### 2.2.7.3 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.

Diagram

Description automatically generated

Figure 43 Tanh activation ([Junxi Feng](https://paperswithcode.com/method/tanh-activation))

#### 2.2.7.4 Rectified Linear Unit (ReLu) activation function

ReLu is also another type of non-linear activation function that maps negative value to 0. It’s 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.

Chart, line chart

Description automatically generated

Figure 44 ReLu activation [(Sagar Sharma)](https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)

#### 2.2.7.5 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).

Diagram

Description automatically generated

Figure 45 Working of Leaky ReLu [(Benjamin McCloskey)](https://towardsdatascience.com/leaky-relu-vs-relu-activation-functions-which-is-better-1a1533d0a89f)

Diagram, schematic

Description automatically generated

Figure 46 Leaky ReLu activation function[(Sagar Sharma)](https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6)

## 2.3 Generative Adversarial Networks

Generative Adversarial Networks, GANs, are a class of deep learning generative models that are capable of mimicking 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 47 GANs working mechanism ([Jason Brownlee](https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/))

### 2.3.1 Deconvolution or Transposed convolution

Three GAN architectures are implemented in this project: **DCGAN**, **WGAN** and **WGAN**-**GP**. 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 oyt trainable upsampling (Deep Dive Into Deep Learning, 2023).The gif below showcases the working mechanism of a transposed convolution.

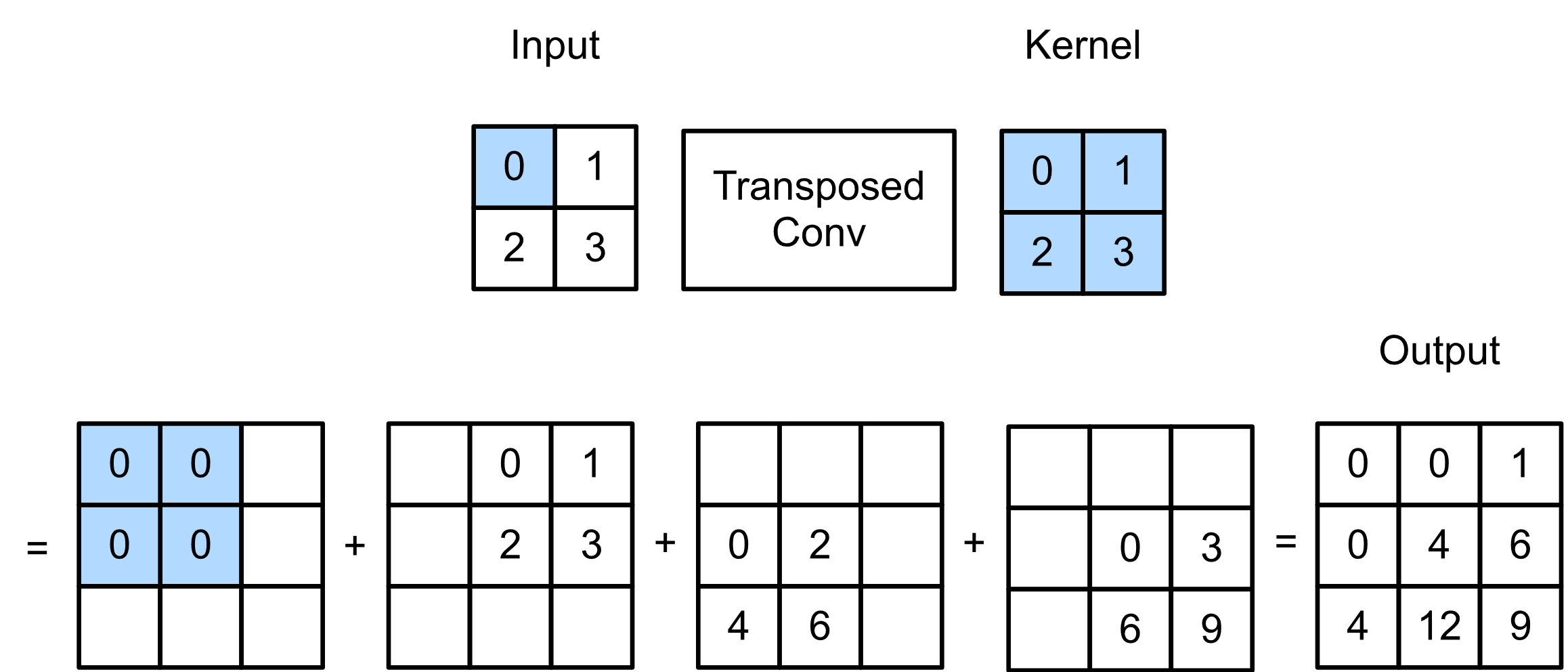


Figure 48 Transposed convolution intuition [(Dive into deep learning)](https://d2l.ai/chapter_computer-vision/transposed-conv.html)

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

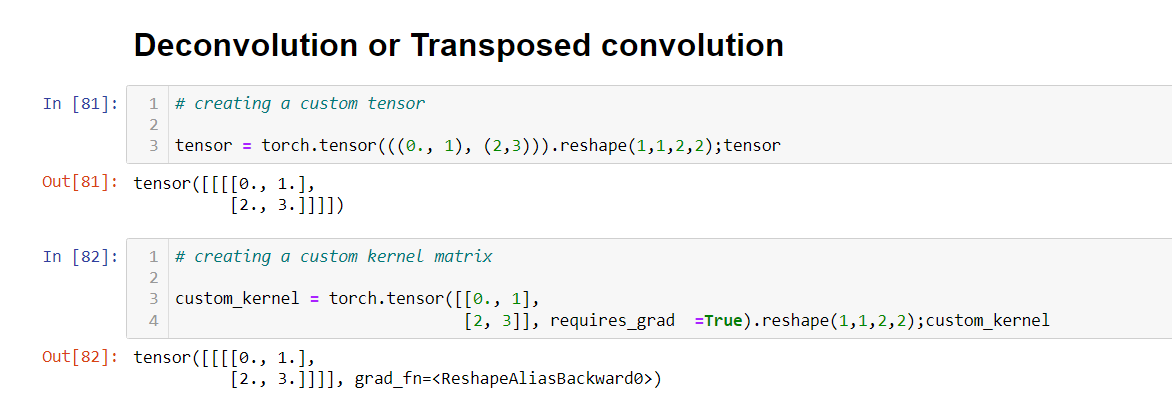


Figure 49 creating a custom tensor and kernel matrix

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.

#### 2.3.1.1 Stride and padding set to 1 and 0

Graphical user interface, application, Word

Description automatically generated

Figure 50 Stride and padding set to 1 and 0 in deconvolution

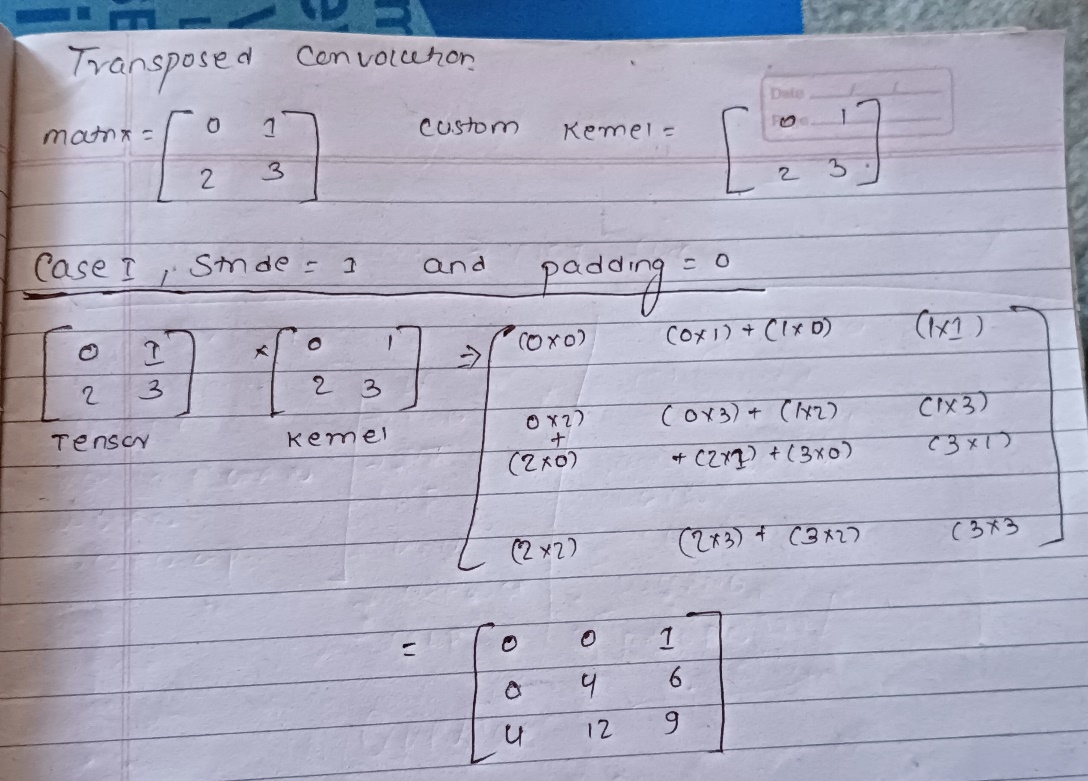


Figure 51 Stride and padding set to 1 and 0 in deconvolution in copy

#### 2.3.1.2 Stride and padding set to 2 and 0

Graphical user interface, text, application

Description automatically generated

Figure 52 Stride and padding set to 2 and 0 in deconvolution

Text, letter

Description automatically generated

Figure 53 Stride and padding set to 2 and 0 in deconvolution in copy

#### 2.3.1.3 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 same. The only difference is that the odd rows and columns added will have 0 values.

Graphical user interface, text, application, email

Description automatically generated

Figure 54 Stride and padding set to 3 and 0 in deconvolution

#### 2.3.1.4 Stride and padding set to 1

Graphical user interface, text, application, email

Description automatically generated

Figure 55 Stride and padding set to 1 in deconvolution

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

#### 2.3.1.5 Stride and padding set to 2

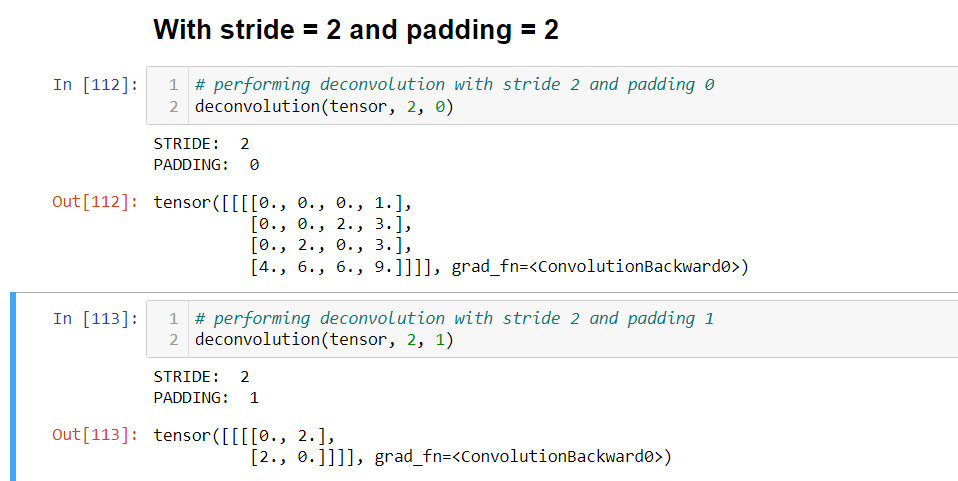


Figure 56 Stride and padding set to 2 in deconvolution

### 2.3.2 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.

Diagram

Description automatically generated

Figure 57 DCGAN Generator architecture [(Original Paper)](https://arxiv.org/pdf/1511.06434v2.pdf)

Diagram

Description automatically generated

Figure 58 Generator architecture [(Original Paper)](https://www.sciencedirect.com/science/article/abs/pii/S0950705119303740?via%3Dihub)

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

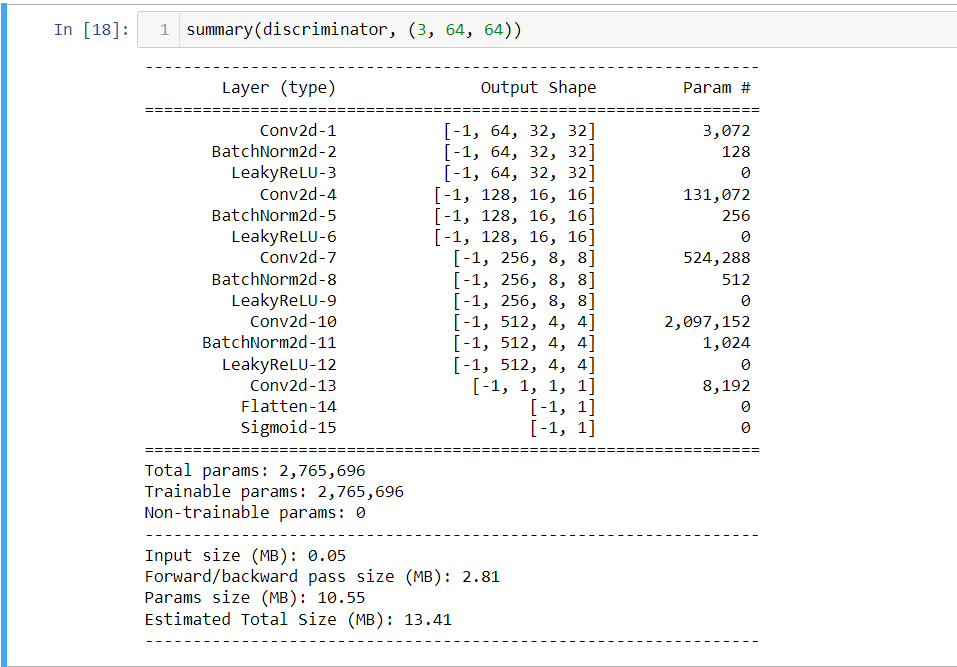


Figure 59 Summary of discriminator model

Graphical user interface, table

Description automatically generated

Figure 60 DCGAN Generator summary

### 2.3.3 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 approach 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).

### 2.3.4 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.

Text

Description automatically generated

Figure 61 WGAN discriminator conditional training

Graphical user interface, text, application

Description automatically generated

Figure 62 WGAN generator conditional training

# 3. Project Methodology

# References

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

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-and-strides.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.

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-about-pooling-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-deep-learning/neural-networks-from-scratch/  
[Accessed 16 January 2023].

IBM, 2022. *Generative adversarial networks explained.* [Online]   
Available at: https://developer.ibm.com/articles/generative-adversarial-networks-explained/  
[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-performance-caaa436f885b  
[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-network-matlab.html  
[Accessed 14 January 2023].

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-cnn-architecture/  
[Accessed 12 November 2022].

Riva, M., 2022. *Batch Normalization in Convolutional Neural Networks.* [Online]   
Available at: https://www.baeldung.com/cs/batch-normalization-cnn#:~: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.

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-deep-learning  
[Accessed 16 January 2023].

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

Waheed, A. et al., 2020. CovidGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved Covid-19 Detection. *IEEE,* Volume 8, pp. 1-8.

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