



Ollscoil  
Teicneolaíochta  
an Atlantaigh

Atlantic  
Technological  
University

# Automated Detection of COVID-19 using Convolutional Neural Networks and Generative Adversarial Networks

A thesis submitted

by

Ultan Kearns

in partial fulfillment of the requirements for the degree of

Master of Science in Computing in Big Data Analytics and Artificial Intelligence

Supervisor: Dr Paul Greaney

Submitted to Quality and Qualifications Ireland (QQI)

Dearbhú Cáilíochta agus Cáilíochtaí Éireann

April 2023

# Contents

<b>Declaration</b>	<b>4</b>
<b>Acknowledgements</b>	<b>5</b>
<b>Acronyms</b>	<b>6</b>
<b>List of Figures</b>	<b>7</b>
<b>List of Tables</b>	<b>14</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Explanation of Generative Adversarial Networks (GANs) . . . . .	1
1.2 Explanation of Artificial Neural Networks (ANNs) . . . . .	2
1.3 What is A Convolutional Neural Network? (CNN) . . . . .	3
1.4 Supervised Learning . . . . .	4
1.5 Unsupervised Learning . . . . .	5
1.6 Tensorflow . . . . .	5
1.7 Keras . . . . .	5
1.8 Background of Problem & Aims of This Paper . . . . .	6
1.9 Datasets . . . . .	8
1.9.1 COVID-19 Chest X-ray . . . . .	8
1.9.2 COVID-19 Radiography Database . . . . .	8
1.9.3 COVID-19 Pneumonia Normal Chest X-ray PA Dataset . . . . .	8
1.9.4 Extensive COVID-19 X-ray and CT Chest Images Dataset . . . . .	9
1.9.5 Use of datasets in This Project . . . . .	9
1.10 Structure of This Thesis . . . . .	9
1.10.1 Chapter 1 - Introduction . . . . .	9
1.10.2 Chapter 2 - Literature Review . . . . .	10
1.10.3 Chapter 3 - Implementation . . . . .	10
1.10.4 Chapter 4 - Results . . . . .	10

1.10.5 Chapter 5 - Further Research and Conclusions . . . . .	10
<b>2 Literature Review</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 Analysis of Existing Models for Automated COVID-19 Detection . . . . .	14
2.3 Challenges & Limitations of Using Artificial Intelligence in Automated Diagnosis Systems for COVID-19 . . . . .	21
2.4 Research into Data Augmentation And Convolutional Neural Networks Architectures . . . . .	21
2.5 Conclusion . . . . .	38
<b>3 Design and Implementation</b>	<b>39</b>
3.1 Introduction . . . . .	39
3.2 CNN Model Design and Comparison . . . . .	40
3.2.1 Baseline Models . . . . .	40
3.3 Transfer Learning CNN Baseline Models . . . . .	48
3.3.1 Radiography Dataset . . . . .	49
3.3.2 X-Ray COVID-19 Dataset . . . . .	52
3.3.3 Evaluation of TL models for The x-ray covid 19 dataset . . . . .	55
3.3.4 COVID-19 Chest X-Ray Dataset . . . . .	55
3.3.5 Extensive COVID-19 Dataset CT . . . . .	58
3.3.6 Extensive COVID-19 Dataset X-ray . . . . .	61
3.4 GAN Baseline Design and Comparison . . . . .	64
3.5 GANs for Radiography Dataset . . . . .	64
3.5.1 VAE(Variational Auto Encoder) . . . . .	65
3.5.2 DCGAN(Deep Convolutional GAN Network) . . . . .	65
3.6 GANs for COVID 19 X-ray dataset . . . . .	68
3.6.1 VAEs . . . . .	68
3.6.2 DCGANs . . . . .	69
3.7 GANs for Chest X-ray COVID-19 . . . . .	71
3.8 Extensive COVID-19 X-Ray / CT . . . . .	71
3.8.1 VAEs . . . . .	71
3.8.2 DCGANs . . . . .	72
3.9 Conclusion . . . . .	78
<b>4 Results</b>	<b>79</b>
4.1 Evaluation of Performance of The Augmented CNN Models . . . . .	79
4.1.1 Radiography CNN Models . . . . .	79
4.1.2 Extensive CNN Models . . . . .	84

4.1.3	X-ray COVID-19 dataset CNN Models . . . . .	94
4.2	Evaluation of GAN Models . . . . .	99
4.2.1	Radiography GAN Models . . . . .	99
4.2.2	Extensive X-Ray GAN Models . . . . .	101
4.2.3	Extensive CT GAN Models . . . . .	102
4.2.4	X-ray COVID-19 dataset GAN Models . . . . .	103
4.2.5	Summary of Results . . . . .	104
4.3	Conclusion . . . . .	105
<b>5</b>	<b>Conclusions</b>	<b>107</b>
5.1	Limitations . . . . .	107
5.1.1	Computational Resources Offered by Google Colab Pro . . . . .	107
5.1.2	Run time Limits in Google Colab Pro . . . . .	108
5.1.3	Lack of Data . . . . .	108
5.1.4	Bias within Original and Augmented Datasets . . . . .	109
5.1.5	Time . . . . .	109
5.1.6	Financial Limitations . . . . .	109
5.2	Future Research . . . . .	109
5.2.1	Suggestions for Future Research . . . . .	110
5.3	Conclusion of Work . . . . .	110
5.3.1	Issues Faced and How They Should be Mitigated in Future Research . .	110
5.3.2	Final Words . . . . .	111

# Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Master of Science in Computing in Big Data Analytics and Artificial Intelligence, is entirely my own work and has not been taken from the work of others except and to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution. I understand that it is my responsibility to ensure that I have adhered to LYIT's rules and regulations.

I hereby certify that the material on which I have relied on for the purpose of my assessment is not deemed as personal data under the GDPR Regulations. Personal data is any data from living people that can be identified. Any personal data used for the purpose of my assessment has been pseudonymised and the data set and identifiers are not held by LYIT. Alternatively, personal data has been anonymised in line with the Data Protection Commissioners Guidelines on Anonymisation.

I give consent for my work to be held for the purposes of education assistance to future Computing students at LYIT and it will not be shared outside the Department of Computing at LYIT. I understand that my assessment may be shared with any other third party and will be held securely in LYIT in line with the Institute's Records Retention Policy.

Signed: Ultan Kearns

Date: Wednesday 12<sup>th</sup> April, 2023

# Acknowledgements

I would first of all like to thank my supervisor during this project Dr. Paul Greaney, he was a fantastic help throughout the course of writing this thesis and this work could not have been completed without his input and help.

I would also like to thank the lecturers who taught me so much during my postgraduate course, both Doctors Karla Muñoz-Esqueival and Shagufta Henna provided a fantastic introduction into many areas of Artificial Intelligence and the knowledge they imparted has helped me a lot throughout the course of conducting this research.

I would also like to thank Andrew Ng, his deep learning courses provided a great foundation into the realm of deep learning and Artificial Intelligence.

I would like to thank Francois Chollet whose book “Deep Learning with Python” was an indispensable resource and helped me get more familiar with the Keras library as well as the best AI practices and concepts such as transfer learning.

Finally I would like to thank my parents and friends for supporting me throughout the course of this masters.

# Acronyms

Acronym	Stands For
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
GAN	Generative Adversarial Network
CT	Computed Topography
LSTM	Long Short Term Memory
AUC	Area Under Curve
DCNN	Deep Convolutional Neural Network
RCNN	Regions with CNN Features
VAE	Varational Auto Encoder
DCGAN	Deep Convolutional Generative Adversarial Network

Table 1: Acronyms used in this thesis

# List of Figures

1.1	Basic Example of Generative Adversarial Network . . . . .	2
1.2	Diagram of a Basic Artificial Neural Network with 1 input layer, 1 output layer, and 1 hidden layer . . . . .	3
1.3	Graph of COVID-19 Statistics by age-range Ireland from October 2021 - December 2021 Courtesy of CSO[16] . . . . .	6
1.4	Cumulative cases of the COVID-19 virus worldwide courtesy of Our World in Data[17] . . . . .	7
2.1	Estimated dates of first COVID-19 cases around the World, Image courtesy of Roberts, Rossman and Jarić[29] . . . . .	12
2.2	Examples of Overfitting, Underfitting and Optimal models, Image courtesy of Abhishek Shrivastava[31] . . . . .	13
2.3	Examples of CT Qualitative images lung segmentation[32] . . . . .	15
2.4	Examples of CT qualitative images infection masks[32] . . . . .	16
2.5	Overview of a Typical LSTM Network[33] . . . . .	18
2.6	Perfomance of CNN model[33] . . . . .	19
2.7	Performance of CNN LSTM Model[33] . . . . .	20
2.8	Example of SMOTE[37] . . . . .	22
2.9	Results from Experiment One (Data Augmentation Using GANs)[36] . . . . .	25
2.10	Figure of learning framework for lychee Classification Model[39] . . . . .	30
2.11	Figure of TransGAN (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	31
2.12	Figure of SSD-MobileNet V2 Architecture (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	31
2.13	Figure of Faster RCNN Architecture (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]	32

2.14	Figure of Faster RCNN Res Block and Inception Module (a) Res block; (b) Inception Module. (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]	33
2.15	Figure of Mean Average Precision of Models. (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]	36
2.16	Figure of Speed of Models in classifying lychee. (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]	37
3.1	Figure of Swish and ReLU activation functions(Image courtesy of Madhura Ingahalikar)[45]	41
3.2	Figure of Train and Validation Accuracy of X-ray COVID-19 dataset CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the accuracy	42
3.3	Figure of Train and Validation Loss of X-ray COVID-19 dataset CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the loss	42
3.4	Train and Validation Accuracy of Radiography CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the accuracy	43
3.5	Figure of Train and Validation Loss of Radiography CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the loss	44
3.6	Figure of Train and Validation Accuracy of COVID-19 chest X-ray CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the accuracy	45
3.7	Figure of Train and Validation Loss of COVID-19 chest X-ray CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the loss	45
3.8	Figure of Train and Validation accuracy of Extensive COVID CNN Baseline Model CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy	47
3.9	Figure of Train and Validation Loss of Extensive COVID CNN Baseline Model CT - The X-Axis shows the epoch number and the Y-axis shows the loss	47
3.10	Figure of Extensive COVID-19 X-ray CNN Baseline Model Train and Validation Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy	48
3.11	Figure of Extensive COVID-19 X-ray CNN Baseline Model Train and Validation Loss - The X-Axis shows the epoch number and the Y-axis shows the loss	48
3.12	Transfer Learning Xception CNN Baseline Train and Validation Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy	49

---

*LIST OF FIGURES*

---

3.13 Transfer Learning Xception CNN Baseline Train and Validation Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	50
3.14 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	50
3.15 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	51
3.16 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	52
3.17 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	52
3.18 Transfer Learning Xception CNN Baseline Train and Validation Accuracy X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	53
3.19 Transfer Learning Xception CNN Baseline Train and Validation Loss X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	53
3.20 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	54
3.21 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	54
3.22 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	55
3.23 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	55
3.24 Transfer Learning Xception CNN Baseline Train and Validation Accuracy Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	56
3.25 Transfer Learning Xception CNN Baseline Train and Validation Loss X-Ray Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	56
3.26 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	57
3.27 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss X-Ray Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	57

---

***LIST OF FIGURES***

3.28 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	58
3.29 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	58
3.30 Transfer Learning Xception CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	59
3.31 Transfer Learning Xception CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	59
3.32 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	60
3.33 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	60
3.34 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	61
3.35 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	61
3.36 Transfer Learning Xception CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	62
3.37 Transfer Learning Xception CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	62
3.38 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	63
3.39 Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	63

3.40 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	64
3.41 Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	64
3.42 Example of Mode Collapse COVID Masks Radiography Dataset . . . . .	65
3.43 Possible Mode Collapse COVID X-ray Extensive Dataset . . . . .	71
3.44 Possible Mode Collapse Non-Covid CT Extensive Dataset . . . . .	72
4.1 Radiography Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	80
4.2 Radiography Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	80
4.3 Radiography Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	81
4.4 Radiography Augmented Xception Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	81
4.5 Radiography Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	82
4.6 Radiography Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	82
4.7 Radiography Augmented EfficientNetV2S Model DCGAN Accuracy The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	83
4.8 Radiography Augmented EfficientNetV2S Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	83
4.9 Extensive CT Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	85
4.10 Extensive CT Augmented Baseline Model DCGAN - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	85
4.11 Extensive CT Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	86
4.12 Extensive CT Augmented Xception Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	86
4.13 Extensive CT Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	87
4.14 Extensive CT Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	87

---

***LIST OF FIGURES***

4.15	Extensive CT Augmented EfficientNetV2S Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	88
4.16	Extensive CT Augmented EfficientNetV2S Model DCGAN - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	88
4.17	Extensive X-ray Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	89
4.18	Extensive X-ray Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	90
4.19	Extensive X-ray Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	91
4.20	Extensive X-ray Augmented Xception Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	91
4.21	Extensive X-ray Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	92
4.22	Extensive X-ray Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	92
4.23	Extensive X-ray Augmented EfficientNetV2S Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	93
4.24	Extensive X-ray Augmented EfficientNetV2S Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	93
4.25	X-ray COVID-19 Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	95
4.26	X-ray COVID-19 Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	95
4.27	X-ray COVID-19 Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	96
4.28	X-ray COVID-19 Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	96
4.29	X-ray COVID-19 Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	97
4.30	X-ray COVID-19 Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	97
4.31	X-ray COVID-19 Augmented EfficientNetV2S Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy . . . . .	98
4.32	X-ray COVID-19 Augmented EfficientNetV2S Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss . . . . .	98
4.33a	Real COVID-19 Radiography Mask Example . . . . .	99
4.33b	Generated COVID-19 Radiography Mask Example DCGAN . . . . .	99

---

***LIST OF FIGURES***

4.34a	Synthetically generated COVID 19 mask with Artefacts(DCGAN) . . . . .	100
4.34b	Malformed Image of synthetically generated COVID 19 Mask(DCGAN) . . . . .	100
4.35a	Real COVID-19 Radiography X-ray Example . . . . .	100
4.35b	Generated COVID-19 Radiography X-ray Example DCGAN . . . . .	100
4.36a	Malformed COVID-19 Radiography X-ray Example Number 1 Radiography DCGAN . . . . .	100
4.36b	Malformed COVID-19 Radiography X-ray Example Number 2 Radiography DCGAN . . . . .	100
4.37a	Generated Pneumonia Radiography Mask Example DCGAN . . . . .	101
4.37b	Real Pneumonia Radiography Mask Example . . . . .	101
4.38a	Real Pneumonia Radiography X-ray Example . . . . .	101
4.38b	Generated Pneumonia Radiography X-ray Example DCGAN . . . . .	101
4.39a	Real COVID X-ray Example Extensive . . . . .	102
4.39b	Synthetically Generated COVID X-ray Example Extensive DCGAN . . . . .	102
4.40a	Real Non COVID X-ray Example Extensive . . . . .	102
4.40b	Synthetically Generated Non COVID X-ray Example Extensive DCGAN	102
4.41a	Real COVID CT Example Extensive . . . . .	103
4.41b	Synthetically Generated COVID CT Example Extensive DCGAN . . . . .	103
4.42a	Synthetically Generated non COVID CT Example Extensive DCGAN . . . . .	103
4.42b	Real non COVID CT Example Extensive . . . . .	103
4.43a	Real Normal X-ray X-ray Dataset COVID-19 . . . . .	103
4.43b	Synthetically Generated Normal X-ray X-ray Dataset COVID-19 DCGAN	103
4.44a	Real Pneumonia X-ray X-ray Dataset COVID-19 . . . . .	104
4.44b	Synthetically Generated Pneumonia X-ray X-ray Dataset COVID-19 DCGAN . . . . .	104

# List of Tables

1	Acronyms used in this thesis . . . . .	6
2.1	Results of Standard CNN Network - A combined deep CNN-LSTM network for the detection of COVID-19 using X-ray images . . . . .	19
2.2	Results of CNN - A combined deep CNN-LSTM network for the detection of COVID-19 using X-ray images . . . . .	20
2.3	GAN Architectures used for experiments in[36]) . . . . .	24
2.4	Results of Cancer data set using different GAN Architectures Experiment 1 (Data Augmentation using GANs)[36] . . . . .	25
2.5	Results and label distribution of Diabetes data set using different GAN Architectures Experiment 1 (Data Augmentation using GANs)[36] . . . . .	26
2.6	Classification results on imbalanced test set experiment 2 (Data Augmentation using GANs)[36] . . . . .	27
2.7	Classification results on balanced test set experiment 2 (Data Augmentation using GANs)[36] . . . . .	27
2.8	Comparison of distribution of data augmented vs original(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	29
2.9	Results of models before Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	34
2.10	Improvement of model's mean average precision after Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	34
2.11	Mean average precision before Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	35
2.12	Mean average precision after Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	35

2.13 Comparison of accuracy of base models vs models with data augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39] . . . . .	37
3.1 X-ray COVID-19 dataset CNN baseline model architecture . . . . .	40
3.2 X-ray COVID-19 dataset CNN baseline model hyperparameters . . . . .	41
3.3 X-ray COVID-19 dataset CNN baseline model results . . . . .	41
3.4 Radiography CNN baseline model architecture . . . . .	43
3.5 Radiography CNN baseline model hyperparameters . . . . .	43
3.6 Radiography CNN baseline results . . . . .	43
3.7 COVID-19 chest X-ray CNN baseline model architecture for COVID-19 Chest X-ray Dataset . . . . .	44
3.8 COVID-19 chest X-ray CNN baseline model hyperparameters for COVID-19 Chest X-ray Dataset . . . . .	45
3.9 COVID-19 chest X-ray CNN baseline model results for COVID-19 Chest X-ray Dataset . . . . .	45
3.10 Extensive COVID-19 CT Dataset CNN baseline model architecture . . . . .	46
3.11 Extensive CT CNN baseline model hyperparameters . . . . .	46
3.12 Extensive CT CNN baseline model results . . . . .	46
3.13 Extensive COVID-19 X-Ray CNN Baseline Model Architecture . . . . .	47
3.14 Extensive COVID-19 X-Ray CNN baseline model hyperparameters . . . . .	48
3.15 Extensive COVID-19 X-Ray CNN baseline model results . . . . .	48
3.16 DCGAN for Producing Synthetic COVID-19 Mask Data From Radiography Dataset . . . . .	67
3.17 DCGAN for Producing Synthetic COVID-19 X-Ray Data From Radiography Dataset . . . . .	68
3.18 DCGAN for Producing Synthetic Normal Class Data for X-ray COVID19 dataset	70
3.19 DCGAN for Producing Synthetic X-ray COVID Class Data for Extensive COVID 19 Dataset . . . . .	73
3.20 DCGAN for Producing Synthetic CT COVID Class Data for Extensive COVID 19 Dataset . . . . .	76
3.21 DCGAN for Producing Synthetic CT Non COVID Class Data for Extensive COVID 19 Dataset . . . . .	78

## **Abstract**

This paper aims to analyze the applications of generative adversarial networks or GANs in overcoming issues of data-shortages in relation to developing convolutional neural networks to automate the diagnosis of COVID-19 in patients. There have been many COVID-19 datasets compiled but many suffer from lack of data-quality, imbalance between classes and data shortages[1][2]. In this paper we aim to create and train multiple convolutional neural networks or CNNs to analyze X-rays of patients lungs to automate the detection of COVID-19. The CNN will be trained with a number of images generated from different GAN architectures to determine which will prove most efficient in automating the detection of COVID-19. In the results section of this Thesis we will compare the non-augmented CNN models with the augmented models to see if there was any effect on accuracy and loss. The aim of this paper is to see if existing COVID models could be improved by use of synthetic data to increase the model's generalisation ability.

# Chapter 1

## Introduction

### 1.1 Explanation of Generative Adversarial Networks (GANs)

A generative adversarial network or GAN for short first appeared in a 2014 paper by Ian Goodfellow et al[3]. In this paper Goodfellow et al propose a new way to generate data via an adversarial process. The GAN essentially works as follows: two models are trained, a generative model  $G$  which will generate the synthetic data from the real data and another model  $D$  which will be the discriminator, judging if the data was created by the model or if it came from the dataset. The goal of this training is to ensure data generated from  $G$  is realistic enough to make the discriminator  $D$  believe that the generated content came from the training set, this is achieved by training the model for a number of epochs and adjusting the models weights to improve the quality of the generated image. It is in this way that we can create realistic "fake" data from the generative model.

There are a number of GAN architectures which are useful in different scenarios, such as CycleGans[4] which are useful for translating images from a source domain  $X \rightarrow Y$  in which  $Y$  is the target domain, StyleGan, which was created by NVIDIA which allows more control over the generative process[5] and PixelRNN, which can recreate images when given a fraction of the original and can generate new images based on probability[6].

This dissertation examines a number of different generative adversarial network architectures and their ability to produce meaningful data when trained on the datasets which will be mentioned in a later section.

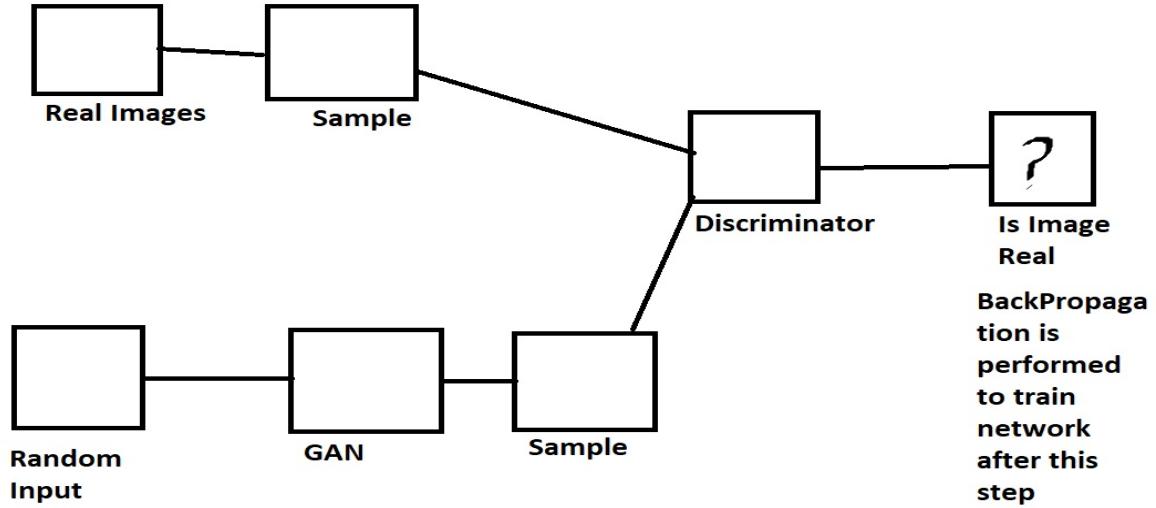


Figure 1.1: Basic Example of Generative Adversarial Network

As we can see from the image above 1.1, we start the process by taking a sample of real images from the training data, then passing it to the discriminator. We also take a sample from the GAN created images and pass that to the discriminator which will then determine if the images are real or fake. After the discriminator guesses if the image generated is good enough to be considered real in terms of quality then backpropagation is performed to train the model so that it can differentiate better between samples that came from the training set and those which came from the generator  $G$ .

## 1.2 Explanation of Artificial Neural Networks (ANNs)

Artificial Neural Networks, or ANNs for short, comprise of a network of neurons or nodes(both terms can be used interchangeably) which are used for training a model to perform a certain task. They are made up of an input layer,  $N$  hidden layers, and finally an output layer. Each layer has its own activation function and will adjust its weights and biases to determine the final output of the model.[7] These networks are heavily inspired by biological processes which occur in the brain.

Artificial neural networks are a general-purpose model used to solve a number of common problems.

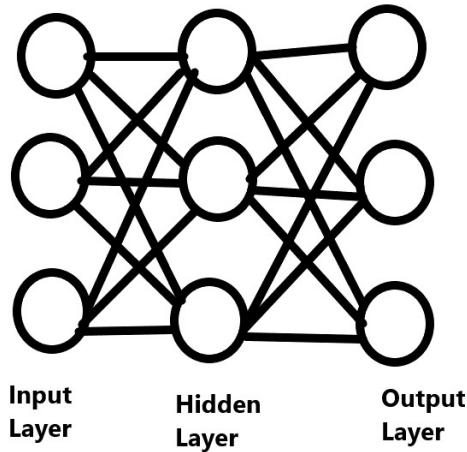


Figure 1.2: Diagram of a Basic Artificial Neural Network with 1 input layer, 1 output layer, and 1 hidden layer

A basic example of an artificial neural network is shown in Figure 1.2. As shown in the figure, the network has an input layer, a hidden layer, and an output layer. Generally when creating these networks we determine the number of neurons in both the input and the output layers based on the number of classes we are trying to predict. The above network could be used to predict if an image is of a cat, a dog, or a fish for example. There can multiple hidden layers in an ANN and the number of neurons in each layer can be adjusted. In reality, artificial neural networks will typically be far bigger than the example given above in terms of neurons and hidden layers but for illustrative purposes, the above diagram will suffice. Each neuron will also have its own weight, bias, and an activation function which will determine whether a neuron fires or not. There are many common activation functions such as swish, ReLU (Rectified Linear Units), sigmoid, and tanh the use of which depends on a number of factors.

### 1.3 What is A Convolutional Neural Network? (CNN)

A convolutional neural network, or CNN for short, is a type of neural network which is primarily used for tasks involving image and pattern recognition[7]. The structure is similar to an ANN in which we have an input layer,  $N$  hidden layers, and finally an output layer. As with the Artificial Neural Network each of these layers will have an activation function and its own weights and biases to determine the final output for a given input. The model will take an image as input, the image is made up of vectors (RGB) or a similar format and from that image the model will determine certain patterns. For example, the output might be a classification of whether or not COVID-19 is present. This application will be discussed in more detail later in the dissertation.

There are a few ways in which CNNs differ from ANNs, in that they are comprised of three types of layers which are the convolutional layers, pooling layers, and fully connected layers[7]. The convolutional layer is responsible for extracting features from an image and generating a  $2D$  activation map, the pooling layer will reduce the parameters of a given input by means of downsampling, and finally the fully connected layers will then determine and classify the output for a given input. The convolutional layer's parameters utilize learnable kernels (a kernel acts as a filter used to extract features from images), and this layer also produces a  $2D$  activation map which will be used to determine if a neuron fires or not for a given input. We can adjust hyper parameters in the convolutional layer to greatly reduce the complexity of the model through optimization, which can be achieved by adjusting the following hyper parameters: depth, stride and zero padding.

Depth is related to the output volume produced by the convolutional layers in the model which can be manually set by adjusting the number of neurons in each layer. Reducing the depth of the model can greatly decrease the training time but at the expense of performance. Stride is related to the spatial dimensionality of the input which will determine the receptive field (this is an area where each neuron in the network is connected to a small region of the input - this area of the network is called the receptive field[7]), if the stride is set to a low integer we will produce extremely large activations, and if it is set too high the network won't produce enough activations. The stride can also be interpreted as sliding a window across the image / video, where only a section of the image or video is input into the network at a time. Finally, zero-padding will pad the border of the images ingested by the model with 0s, reducing their dimensionality. Padding can prove to be useful in increasing the accuracy of the model as it can possibly eliminate areas of the image which are not useful for the model and can also improve training time times in some use cases[8].

Through the adjustment of the hyperparameters mentioned above, and through the utilization of different activation functions, the accuracy of the convolutional neural network can be improved through a process of trial and error.

## 1.4 Supervised Learning

Supervised learning is a methodology of machine-learning involving the use of labeled data to train the model[9]. The data is typically labeled manually by a data scientist, which can be a long and laborious process depending on a number of factors (size of the data, number of classes, etc.), but offers many benefits when it comes to training models. Supervised learning performs extremely well at tasks involving classification (classifying data into a given category), and regression (understanding the relationships between independent and dependent variables).

## 1.5 Unsupervised Learning

Unsupervised learning is a methodology of machine-learning which involves using unlabelled data to train machine learning models[9]. This type of machine learning requires no human intervention since the data is unlabelled and the model will detect relationships between data based on the raw data fed in to the model. This type of machine learning is used for tasks such as: clustering (grouping data together based on shared characteristics or features), association (finding relationships between features), and dimensionality reduction (reducing the number of features in a given dataset without compromising the integrity of said data). The key differences between supervised and unsupervised learning are: labeled versus unlabeled datasets, and finding relationships in data (unsupervised) or trying to predict and classify data (supervised).

## 1.6 Tensorflow

Tensorflow is an open-source library which allows developers to access a number of functions to make machine learning easier and allows developers to build models more quickly[10]. Tensorflow provides numerous modules and classes which form the foundation for building both the generative adversarial network and the convolutional neural network. There have been numerous case studies proving the efficacy of Tensorflow in solving many AI / ML problems and the library is used by research teams in organisations such as Google, Airbnb, ARM, Coca-Cola, Intel, and many more[11].

Given the reputation and widespread use of Tensorflow, and the vast amount of documentation around the framework, it seems an ideal library for the implementation of GANs and CNNs for this study.

## 1.7 Keras

Keras is a deep-learning framework for Python developed by Francois Chollet which provides a number of helpful functions and methods for creating AI models[12]. Keras is built on top of Tensorflow and simplifies data loading, pre-processing and the overall building of the model. Keras is commonly used by data-scientists and researchers due to the powerful methods it offers and the time it saves. The additional classes and modules Keras provides on top of Tensorflow will help to reduce the time taken to build and develop both the convolutional neural network and the generative adversarial network.

Like Tensorflow, Keras has been used by a number of companies and is well recognised in the Artificial Intelligence community. The framework has a range of uses and has proven beneficial when developing AI models in a number of areas and fields[13].

## 1.8 Background of Problem & Aims of This Paper

COVID-19 is a highly transmissible virus which has caused a worldwide pandemic and has claimed many lives. There have been 616,951,418 cases worldwide and 6,530,281 deaths as of the 4th of October 2022[14]. During the pandemic, Ireland alone was subject to a total of 1.6 million confirmed cases and nearly 8,000 deaths[15]. This has led many researchers to pursue the goal of automating the detection of COVID-19 to partially relieve the immense pressure put on medical staff throughout the pandemic.

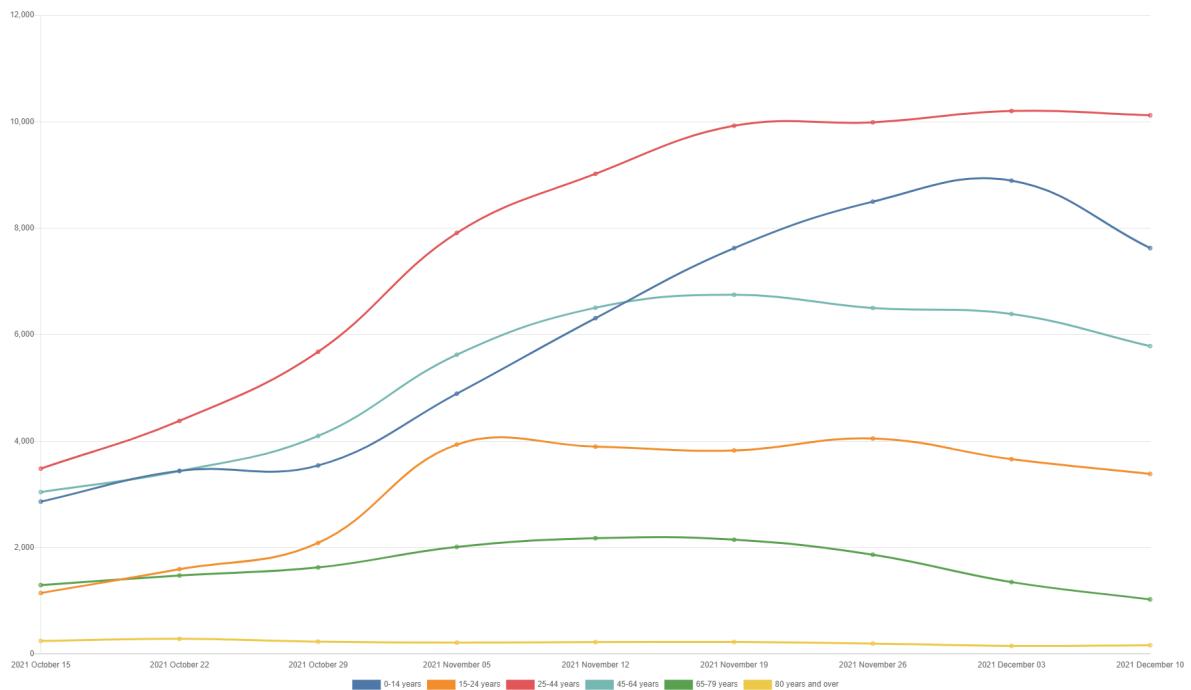
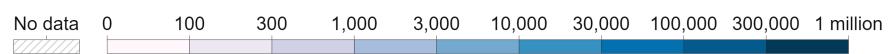


Figure 1.3: Graph of COVID-19 Statistics by age-range Ireland from October 2021 - December 2021 Courtesy of CSO[16]

Cumulative confirmed COVID-19 cases per million people, Oct 9, 2022  
Due to limited testing, the number of confirmed cases is lower than the true number of infections.

Our World  
in Data



Source: Johns Hopkins University CSSE COVID-19 Data

CC BY

Figure 1.4: Cumulative cases of the COVID-19 virus worldwide courtesy of Our World in Data[17]

The main objective of this research is to develop a robust model which can accurately analyze X-rays of patients and determine from said X-rays if the patient is afflicted with COVID-19. This will be achieved by utilizing a number of different GAN architectures which will create realistic "fake data" which will then be used to train a number of models. From this training we plan to compare and contrast the results when generating data with different architectures to determine the best configuration for data generation to train the CNN model. There has been some success in utilizing convolutional neural networks to automate the detection of the virus[18][19]. Through the use of synthetic data generated by utilizing a variety of GAN architectures, it is our hope that such convolutional models will be improved upon and made more accurate.

We plan on utilizing existing data sets which are listed in the next section when training the Generative Adversarial Models, through trial and error we plan on determining the best architecture of GANs to use for training the model for this use-case.

## 1.9 Datasets

Before beginning the training of the model it is important to explore and understand each of the datasets. There are a total of three datasets which will be used in the course of this research, we will explain more about these datasets below.

### 1.9.1 COVID-19 Chest X-ray

The COVID-19 Chest X-ray data set is a data set which is comprised of labeled X-ray Images taken from a number of patients. This dataset contains 357 X-ray images of COVID positive patients, and Chest X-rays of those afflicted with another disease (MERS, SARS, and ARDS). This dataset also includes a metadata file listing the diagnosis of the patient along with a number of other features[20]. In total this dataset contains 11 classes, the images do not have a consistent resolution which may cause issues as resizing each image may lead to a loss in image quality. The loss in quality will cause problems when evaluating the model's accuracy.

### 1.9.2 COVID-19 Radiography Database

The COVID-19 Radiography Database is made up of 3,616 images of chest X-rays taken from COVID positive patients, 10,192 Images of lung X-rays taken from healthy patients, and 1,345 X-ray images of viral pneumonia positive patients. All images in this dataset are PNG (Portable Network Graphic) images and are at a resolution of height 299 pixels and width 299 pixels eliminating the need for preprocessing of the images, the dataset also includes metadata for each of the images in this dataset showing a number of features with the diagnosis of the patient as well. The data in this dataset was gathered by a team of researchers from Qatar University, Doha, Qatar, and the University of Dhaka, Bangladesh along with their collaborators from Pakistan and Malaysia[21].

### 1.9.3 COVID-19 Pneumonia Normal Chest X-ray PA Dataset

The COVID-19 Pneumonia Normal Chest X-ray PA dataset is comprised of a train set containing 74 Normal X-ray Images taken from healthy Patients and afflicted with Pneumonia and a test set containing a Normal set containing 20 chest X-rays taken from healthy patients and a Pneumonia set containing 20 images. The images in this dataset are unlabelled and no metadata is offered, however, the images are segregated into separate files listing the diagnosis[22].

#### **1.9.4 Extensive COVID-19 X-ray and CT Chest Images Dataset**

This dataset was added fairly late in the project due to data limitations in both the COVID-19 Chest X-ray and the COVID-19 Pneumonia Normal Chest X-ray PA Dataset. This dataset is comprised of 17099 X-ray and CT images which were generated with various augmentation techniques. Some of the images in this dataset come from the previously mentioned datasets namely the COVID-19 Pneumonia Normal Chest X-ray PA Dataset and the COVID-19 Chest X-ray. Given the large number of images in this dataset it may prove useful when training GANs to reproduce images as some of the other datasets have proven to lack the data needed to reasonably train a GAN to reproduce the X-rays.

The dataset is broken up into two folders containing X-rays and CT Scans respectively. Both folders contain images which are categorized into two further subfolders one containing COVID Positive X-ray and CT-Scans and the other containing COVID negative X-ray and CT scans[23].

#### **1.9.5 Use of datasets in This Project**

we plan to use each of these datasets to train and test the model and use data augmentation to increase the train and test sets by utilizing Generalized Adversarial Networks. When using these datasets in conjunction it is my hope that the GAN will have enough data to be effective when generating new sample images to train the final model.

### **1.10 Structure of This Thesis**

This thesis is broken into 5 chapters in total, this section will include the headings of the chapters and a brief summary of each chapter below:

#### **1.10.1 Chapter 1 - Introduction**

This chapter will offer the reader of this thesis a brief introduction to a number of core concepts which will be necessary to understand before diving deeper into this thesis. It is important that the reader has a basic understanding of generative adversarial networks, convolutional neural networks, artificial neural networks, supervised & unsupervised learning, and the overall question that this research proposes before discussing the implementation or discussing pertinent literature in this field.

In this section we will frame the research question, explain what a generative adversarial network is, its function, and how it works, We will also explain artificial neural networks and convolutional neural networks, and we will discuss the basic methodologies relating to the implementation of this project. We will also discuss the libraries used to implement the

practical artifact, datasets used to train the model and give the reader of this thesis a clear understanding of the key aims of this research.

#### **1.10.2 Chapter 2 - Literature Review**

In this section we will review pertinent literature related to the problem domain and discuss the ideas and concepts presented in these papers. We will also review the results from the research conducted in these papers and use them as a metric to gauge the performance of my own model. The papers will also be compared and contrasted and we will discuss the findings and how useful these papers were when conducting my own research. It is very important to understand the problem domain before beginning the implementation of this project to ensure that we are not "reinventing the wheel". This section will also provide the reader of this thesis with the most up-to-date progress made within the problem domain.

#### **1.10.3 Chapter 3 - Implementation**

In this section we will discuss the architecture of the convolutional model, the various architectures of generative adversarial networks implemented, how the models were trained and the overall design of the code implemented, and the rationale behind certain design choices. we will also show the results from training the models and discuss how through trial and error we were able to improve the various models and will include code samples so that the models can be reviewed by the reader or re-implemented by them.

#### **1.10.4 Chapter 4 - Results**

In this section, we will review the results achieved from training the best models and suggest how they may possibly be improved. we will be showing lots of graphs/tables in this section to gauge each model's test/dev set errors and we will also be comparing and contrasting the effects of the different GAN architectures implemented as well as discussing the results of the convolutional model.

#### **1.10.5 Chapter 5 - Further Research and Conclusions**

In this section, we will discuss further research that may need to be done by any researchers who would like to build upon this research. we will also review where the models could be improved and what we'd do differently if we were to conduct this research again. we will also discuss common issues we faced during the implementation of this project and how we overcame them. This section will be a summary of all the research conducted, the code, and my experience overall throughout the writing of this thesis.

This will be the final section of the paper and will tie the entire thesis together.

# **Chapter 2**

## **Literature Review**

### **2.1 Introduction**

The first reported cases of COVID-19 occurred in Wuhan, China on December 12th, 2019[24], when a number of patients began to exhibit "symptoms of an atypical pneumonia-like illness that does not respond well to standard treatments". It was not until December 31, 2019, that the World Health Organization(WHO) Country Office in China was informed of several more cases of this strange virus described as "pneumonia of unknown etiology", the symptoms of this new virus were shortness of breath and fever. All the initial cases observed seem to have been connected to a market called Huanan Seafood Wholesale Market. On January 1st of 2020, the Huanan Seafood Wholesale Market was shut down amid concerns over the spread of this new virus. On January 3rd the Government of China alerted the World Health Organization that they had identified over 40 new cases of this pneumonia-like disease, and on the 5th of January, Chinese public health officials shared the genetic sequence of the new virus with the world through a database that could be accessed by the public. Following the release of this information the CDC (Centers for Disease Control and Prevention), which is a US Government funded healthcare research agency, began an investigation into the origins of this new virus. The origins of COVID-19 are not clear and are still being researched. The most likely explanation offered by scientists is that it originated in the Huanan Seafood Wholesale Market from animals sold there - a likely culprit is the Raccoon Dog which is used for fur and food in China[25], but other theories suggest that a lab leak at a biological weapons facility[26] may be responsible for the creation of the virus. Some researchers are currently suggesting that blood samples taken from animals sold at the Hunan Market and samples from the people who sold them may lead to definitive evidence of the disease's origins[25]. Although the origins of COVID-19 still remain up for debate it is clear that when studying the virus, its high transmission rate and the speed at which it can spread made it one of the deadliest viruses in human history.[27][28]

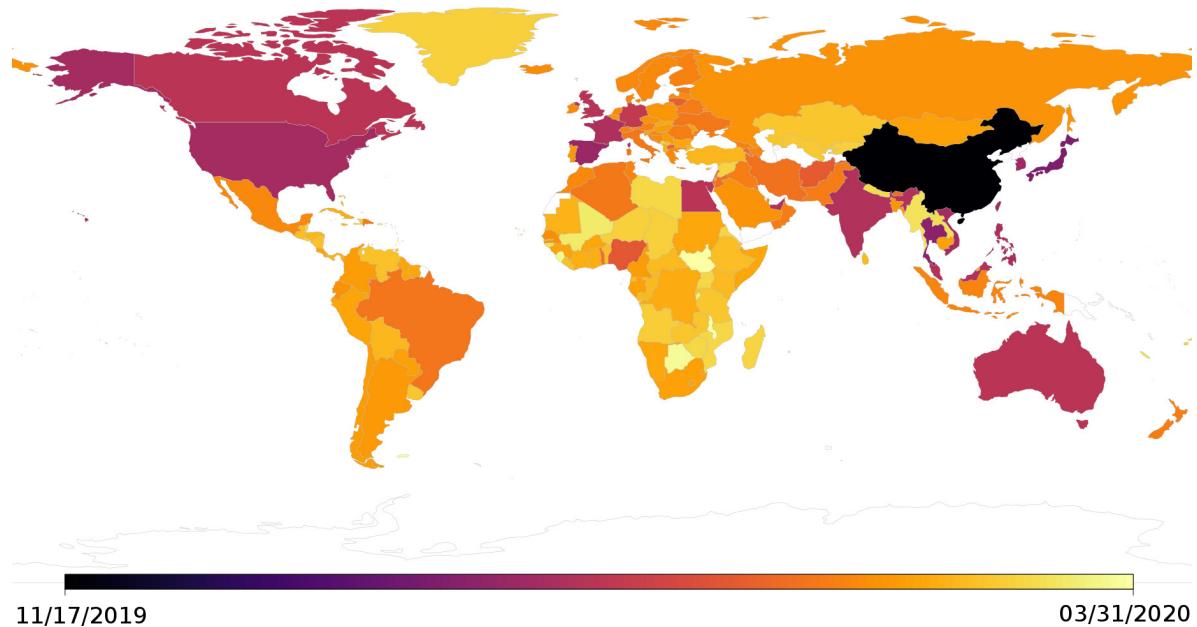


Figure 2.1: Estimated dates of first COVID-19 cases around the World, Image courtesy of Roberts, Rossman and Jarić[29]

Due to this rapid spread of the virus, many governments were unprepared for dealing with such an outbreak. The artificial intelligence community had published many papers and conducted much research into designing automated tools which could relieve medical professionals of the extreme stress they were under. Unfortunately, most of the models trained were of no use to medical professionals and some were even deemed harmful[30]. There were many limitations when it came to training automated diagnostic tools for COVID-19, such as incorrect assumptions about the data, lack of data quality, and lack of data in general. Due to the poor / limited quality of data and the urgent need for diagnostic tools, many of the models were trained using poor-quality data or incorrect data. Such poor models would have had drastic effects if patients who were COVID positive were diagnosed as negative by the model, and models suffering from high false negative rates would have drastic consequences for the patients afflicted with COVID.

The models trained on what have been termed as "Frankenstein Datasets" suffered immensely, as some of the data came from the same source, meaning the same data from the training set could have been present in the test set. This would severely impact the performance of the model, as it would have to overfit the data from which it was trained. These models which were overfitted on the data would seem to have high accuracy but ultimately would perform poorly on real-world data.

	<b>Underfitting</b>	<b>Just right</b>	<b>Overfitting</b>
<b>Symptoms</b>	<ul style="list-style-type: none"> <li>• High training error</li> <li>• Training error close to test error</li> <li>• High bias</li> </ul>	<ul style="list-style-type: none"> <li>• Training error slightly lower than test error</li> </ul>	<ul style="list-style-type: none"> <li>• Very low training error</li> <li>• Training error much lower than test error</li> <li>• High variance</li> </ul>
<b>Regression illustration</b>			
<b>Classification illustration</b>			
<b>Deep learning illustration</b>			
<b>Possible remedies</b>	<ul style="list-style-type: none"> <li>• Complexify model</li> <li>• Add more features</li> <li>• Train longer</li> </ul>		<ul style="list-style-type: none"> <li>• Perform regularization</li> <li>• Get more data</li> </ul>

Figure 2.2: Examples of Overfitting, Underfitting and Optimal models, Image courtesy of Abhishek Shrivastava[31]

Figure 2.2 above shows how a model's performance can be analyzed. Underfitting yields a high training error and high bias meaning that the model will perform poorly on the training, test, and dev sets. Overfitting leads to a very low training error which will be lower than the test, and dev set error and wouldn't be fit for purpose when analyzing real-world data. The optimal model has a training error that is slightly lower or in and around the same accuracy as the test and dev sets.

The lack of medical experience also played a role in the poor performance of these models as many of the AI researchers training these models would be unfamiliar with flaws in the data. The bias of the radiologist labeling the X-rays of patients also played a role as the radiologist could have inaccurately diagnosed the patient as COVID positive or negative. Private Artificial Intelligence companies also played a role in poor model development as published models from researchers tied to the company also showed that these models had a

high risk of bias.[30]

As the pandemic progressed more and more data was made available to researchers which was able to mitigate some of the problems stated above, leading to more accurate and robust models which we will explore in the later sections.

## **2.2 Analysis of Existing Models for Automated COVID-19 Detection**

In a paper by Mahmoudi, Benamour et al [32] researchers investigated a deep-learning approach to creating a diagnostic tool for COVID-19. The research involved utilizing data taken from computed topography scans. These scans segmented the infected regions of a patient's lungs to determine if said patient was afflicted with the COVID-19 virus. The researchers also used a technique called "Contrast Limited Adaptive Histogram Equalization" which is a preprocessing method that removes noise and intensity to create a homogeneous dataset. The researchers also removed black slices from the images so that only the region of interest was highlighted, with the goal of further improvement of the performance of the model. U-Net architecture is an architecture that is based on convolutional neural network encoders and decoders, this architecture was used in the creation of this model to allow for more timely and accurate segmentation of images and to generate the segmentation models of the lung and infection x-rays. Four-fold cross-validation (this is where the dataset is sliced into four equal parts depending on the size for odd datasets there may be set with the remainder of values if not equally divisible by four), then the models are trained on one or multiple sections, and tested with another section. The final model is taken from the model with the best performance and was then used to analyze the performance of the model along with a three-layered CNN architecture which was comprised of added fully-connected layers which was connected to a softmax output layer, the output layer was used for classification of the images.

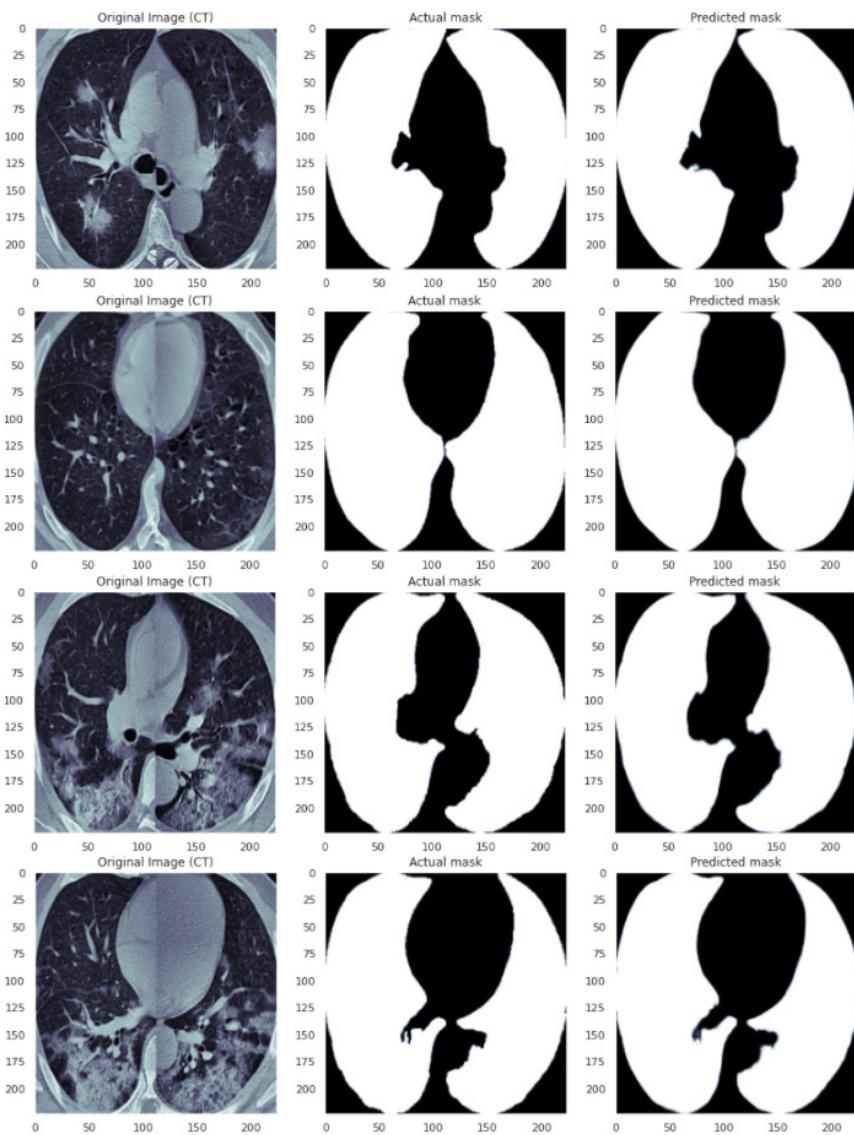


Figure 2.3: Examples of CT Qualitative images lung segmentation[32]

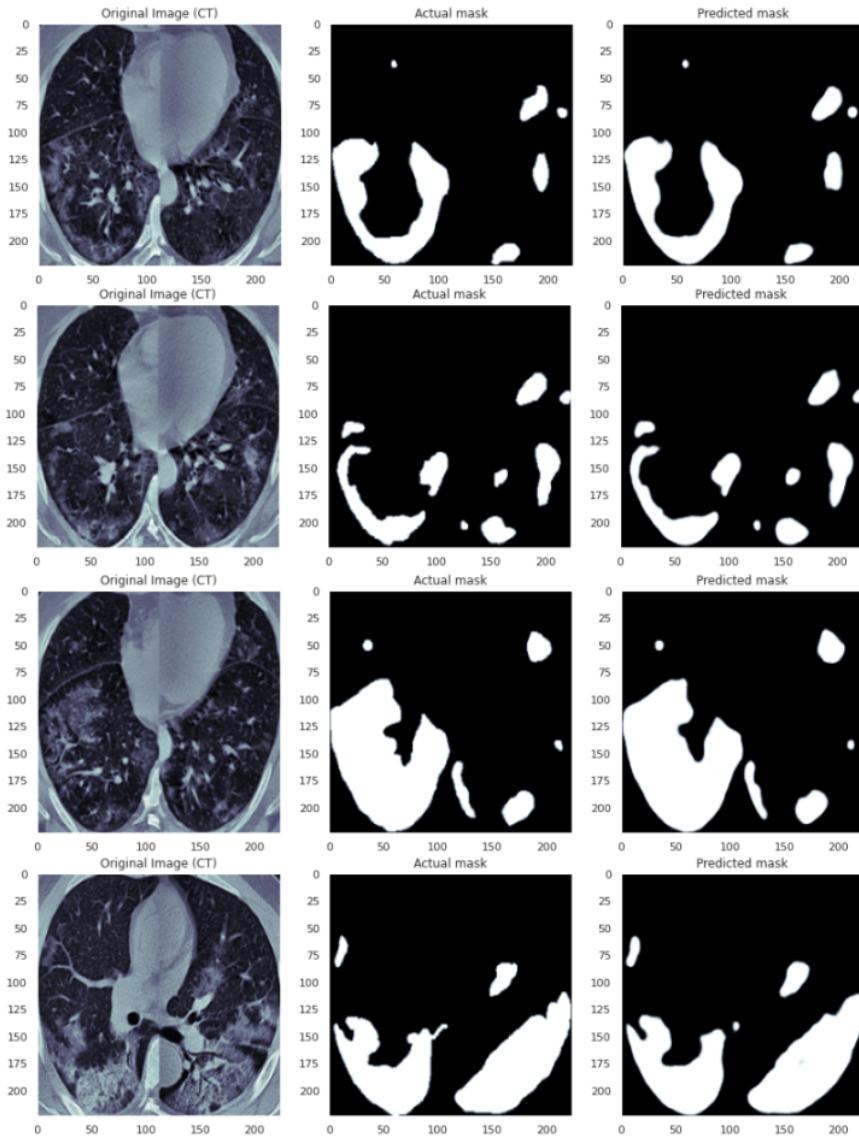


Figure 2.4: Examples of CT qualitative images infection masks[32]

Figure 2.3 shows the lung segmentation results, one column containing the ground truth and the other containing the researchers' proposed four slice model, which is compromised of four slices taken from different CT scans. The first column shows the original CT scan, the second column shows ground truth, and finally, the third column shows predicted lung masks. Figure 2.4 we can observe the qualitative comparison between the researchers' infection segmentation results which is made up of four slices from different CT scans and the ground truth. The first column shows the original CT scan, the second column shows the ground truth, and finally, the third column shows predicted infection masks. Utilizing a 70% - 30% training set and validation set split, the researchers were able to demonstrate that the system proposed attained a dice score (a value ranging from 0 - 1, used to

gauge performance by comparing the results of the output of the model to that of the ground truth, where 1 is a perfect overlap and 0 is no overlap) of 98% and 91% for lung and infection segmentation tasks. Additionally, the system accurately diagnosed patients afflicted with COVID-19 98% of the time. The development of this model suffered from a lack of data, as only 20 CT scans were used to train and test the model. The limited data set used suggests the researcher's model may have possibly been overfitting the training data. The researchers mention as much in the conclusion section of this paper. They discuss how the main limitation of the study is the use of a small but sufficient amount of training data. The restrictions on data collection coupled with the high cost of labelling the data meant that the researchers were only able to utilize the 20 CT scans for both training and testing the model. From the conclusion section of this paper, it is clear that there is a high potential for bias in the data set used by the researchers. The possibly mislabelled images links back to the "Frankenstein Datasets" which I mentioned in the introduction section of this literature review.

In another paper by Islam, Islam, and Asraf[33] we see a new method being used by researchers to develop an automated diagnostic tool. The researchers used a combination of a convolutional neural network with LSTM (long short-term memory), they used the convolutional neural network for deep feature extraction and LSTM for detection of COVID-19 using an extracted feature. They also used a dataset containing 4575 X-ray images which included 1525 images of COVID-19 X-rays. The experimental results of the system are as follows: 99.4% accuracy, AUC (area under curve) accuracy of 99.9%, specificity of 99.2%, sensitivity of 99.3% , and finally an F1 score of 98.9%. The researchers suggest that this system could be further improved in the abstract if more data were available to the researchers. As we can see this study also appears to suffer from a lack of data as per the previous paper discussed. The lack of data is clearly visible but by utilizing an LSTM network the researchers further improved upon the diagnostic model discussed in the previous paper by Mahmoudi et al. The dataset used was also greater in size than the dataset used in the prior paper, this would lead to a more robust model with a greater ability to generalize.

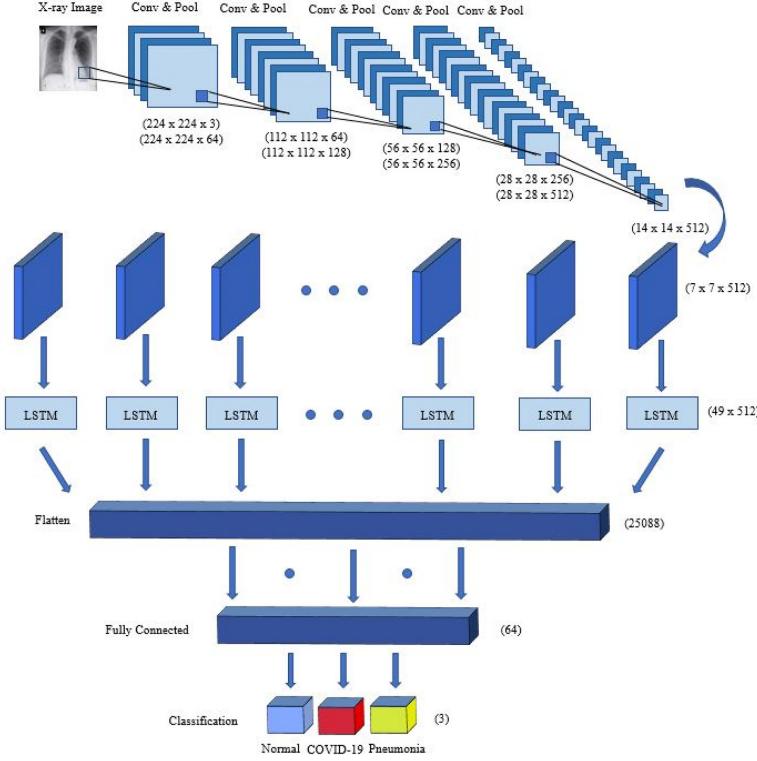


Figure 2.5: Overview of a Typical LSTM Network[33]

The model benefited from the use of LSTM, as an LSTM network has an internal memory that is utilized to learn from experience with long-term states. LSTM is based on recurrent neural networks, and improves upon them by using memory blocks instead of conventional RNN units, this helps to solve the vanishing and exploding gradient problem[34]. In addition to the memory blocks, there is also a cell state which saves long-term states, the cell states being the main difference between recurrent neural networks and LSTM. The network is capable of remembering and connecting previous information to present data[35]. The LSTM Network is comprised of three gates the input gate which is termed the "forget gate", an update gate, and finally an output gate. These gates essentially determine which data is worth remembering and which data can be forgotten.

The results achieved by the various models are shown in figure 2.6 and figure 2.7

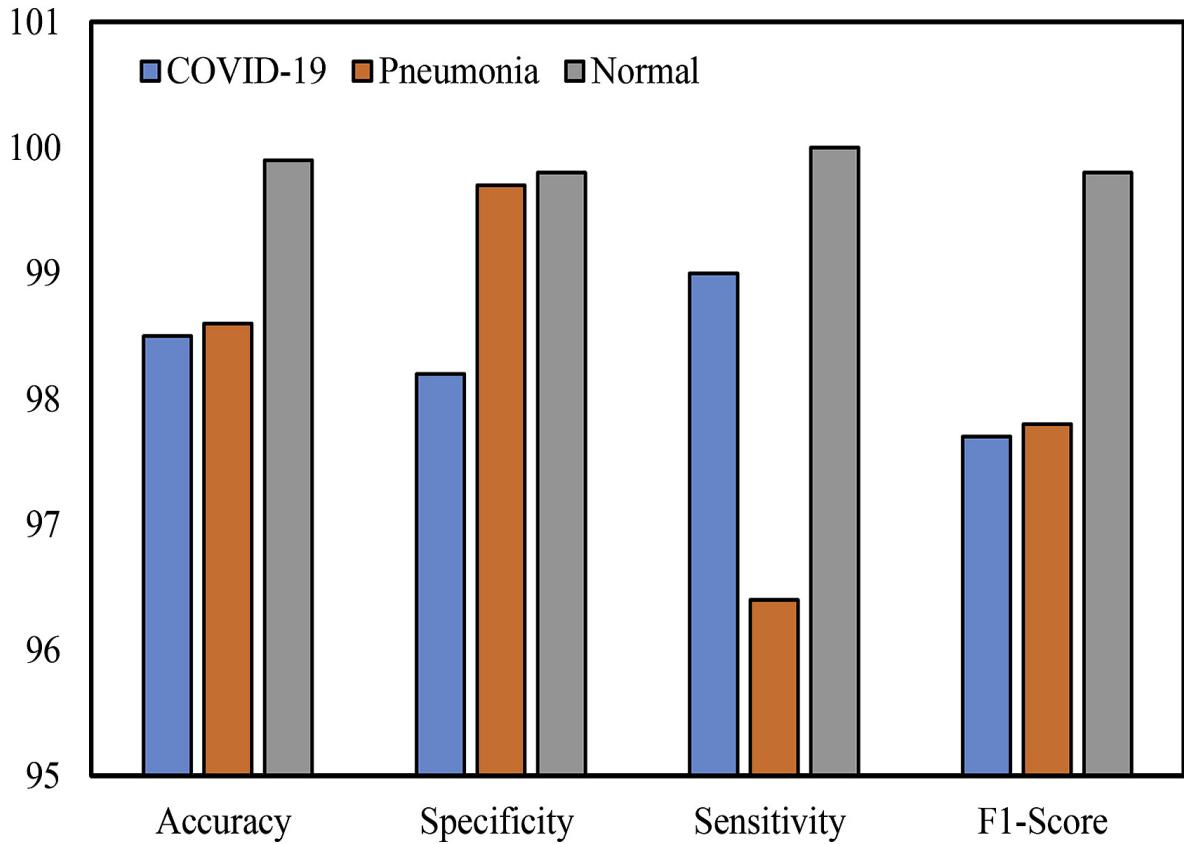


Figure 2.6: Perfomance of CNN model[33]

Figure 2.6 shows the performance of a standard CNN model the accuracy ratings are as follows:

Class	Accuracy	Specificity	Sensitivity	F1-Score
COVID-19	98.5	98.2	99.0	97.7
Pneumonia	98.6	99.7	96.4	97.8
Normal	99.9	99.8	100.0	99.8

Table 2.1: Results of Standard CNN Network - A combined deep CNN-LSTM network for the detection of COVID-19 using X-ray images

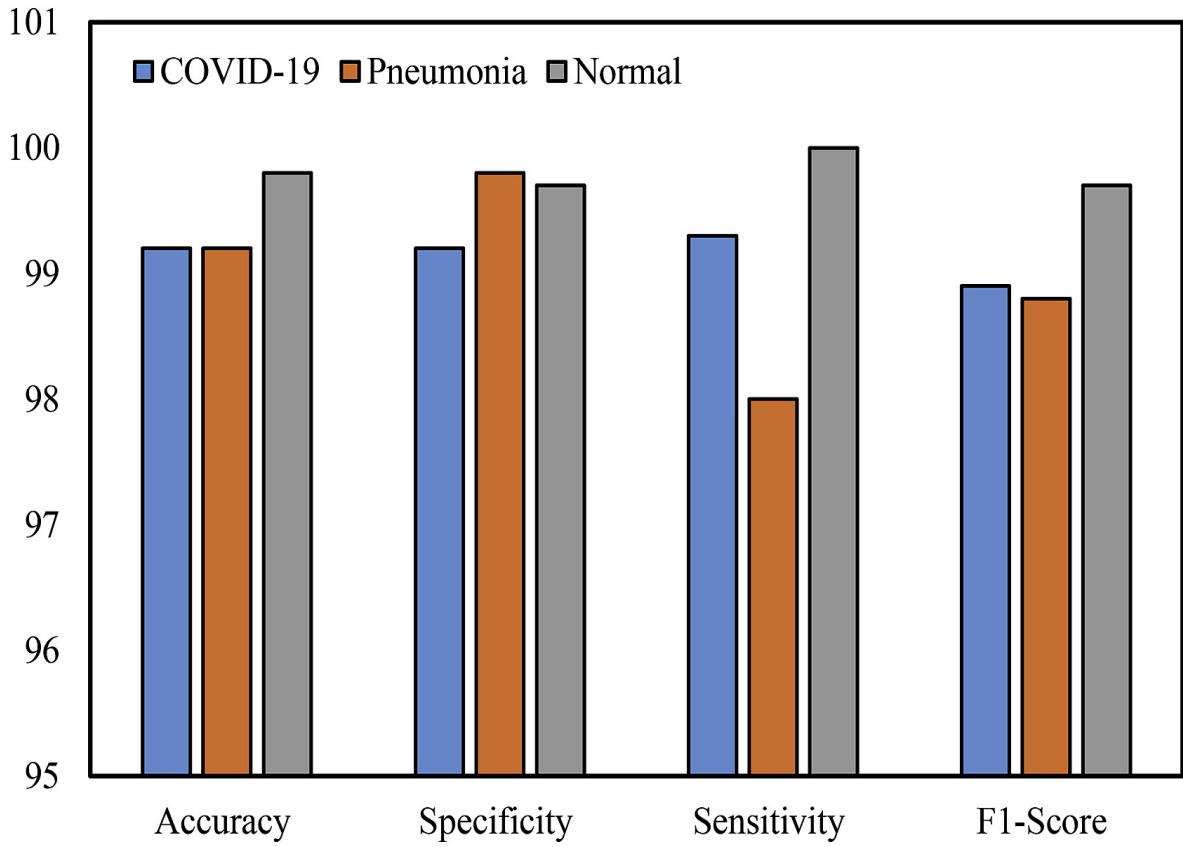


Figure 2.7: Performance of CNN LSTM Model[33]

Class	Accuracy	Specificity	Sensitivity	F1-Score
COVID-19	99.2	99.2	99.3	98.9
Pneumonia	99.2	99.8	98.0	98.8
Normal	99.8	99.7	100.0	99.7

Table 2.2: Results of CNN - A combined deep CNN-LSTM network for the detection of COVID-19 using X-ray images

From figure 2.7 it is clear that the model utilizing LSTM outperformed the basic CNN model on almost all fronts (with the exception of classification of normal patients which experienced a small decrease in accuracy), yielding a higher accuracy, specificity, sensitivity and F1 score for identifying COVID-19, and Pneumonia from X-rays.

Despite the results achieved by the model, the final model suffers from lack of data which the researchers address in the conclusion section of this paper. The other shortcoming of this model is that it focuses on posterior-anterior view of X-Rays meaning that it cannot diagnose X-Rays which are in other formats. The authors of the paper also address that X-Rays, where the patient is afflicted with multiple diseases, cannot be efficiently classified by the model and the model's accuracy was not compared with that of radiologists. Data augmentation may

prove useful for such a model using a combination of CNN and LSTM.

### **2.3 Challenges & Limitations of Using Artificial Intelligence in Automated Diagnosis Systems for COVID-19**

In a paper by Huang, Yang and others, researchers offer an analysis of the challenges of developing Artificial Intelligence to assist medical professionals in the identification and diagnosis of COVID-19. As mentioned previously in this thesis the challenges include: lack of data, lack of data quality and the use of poorly merged data sets termed as Frankenstein data sets when training models. There are, however, more challenges that are faced when developing diagnostics tools, as discussed in the previously cited paper[33] it is very difficult to find people who are COVID-19 positive and asymptomatic due to them not getting treatment as no symptoms are apparent. The labelling of data is also an issue as the X-rays of patients may only show moderate signs of COVID-19 which yields a risk of mislabelled data by clinicians. There is also a risk of false positives and false negatives when developing a diagnostic tool. False positives would cause a patient to unnecessarily be quarantined and false negatives could cause a patient to inadvertently spread COVID-19 to others. Some patients who have already been infected with the virus may show no signs on CT images which also yield high false negative rates making it difficult to distinguish COVID positive patients from COVID negative patients.

To mitigate these challenges the researchers suggest that when developing an Artificially Intelligent diagnostic system the developer should combine chest imaging, exposure history and laboratory tests when training and testing the model. Such data, however, is hard to come by as there are multiple laws concerning data-collection and ethical questions regarding the patient's right to privacy.

### **2.4 Research into Data Augmentation And Convolutional Neural Networks Architectures**

Data augmentation allows artificial intelligence researchers to artificially inflate the size of the amount of data, this is done by utilizing existing data and detecting patterns in the data. From the original data, new data is produced using various methods such as rotating images, applying filters, altering various aspects of the image (such as padding, cropping, and zooming), etc. There are numerous techniques and methodologies for using data augmentation but for the purpose of this thesis we will be using Generative Adversarial Networks to create new data<sup>1.1</sup> for the purpose of improving upon existing automated diagnostic models for COVID-19. There are multiple different types of GAN architecture that have been covered

in the introduction section of this thesis. The key advantages of using data augmentation are as follows: larger training set to train models on, reduces overfitting, helps to prevent underfitting and improve the accuracy of the model, reduces the cost / time associated with the collection of new data, and increases the trained model's ability to generalize. There are however some challenges when it comes to using data augmentation such as: inability to reduce bias of new data(if there is bias in the existing data the new data will also contain bias), hard to generate discrete data such as text, and the data generated will need to be evaluated.

In a paper by Tanaka and Aranha[36] the researchers outline two algorithms to oversample the minority class within the data with the aim of balancing the data set. These two methodologies are called SMOTE (Synthetic Minority Over-sampling Technique)[37] and ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning)[38].

SMOTE works by creating artificial data which takes into account the data's position, a random point in the least represented class in the dataset is selected and SMOTE identifies members of the same class within the data by using the  $k$ -nearest neighbour algorithm which is a form of unsupervised learning. SMOTE then generates an entirely new point in the vector for each pair which is situated between the two pieces of data, the new point is then positioned at a random percentage away from the initial point chosen.[36]

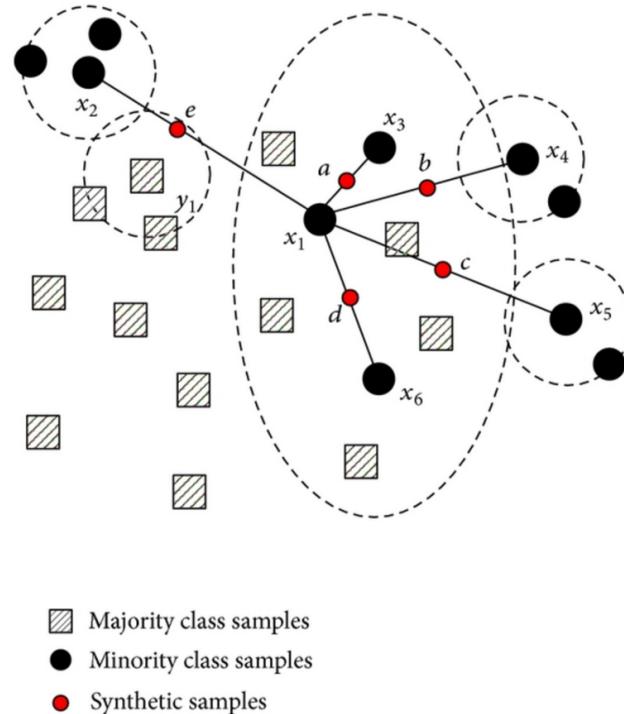


Figure 2.8: Example of SMOTE[37]

As we can see from figure2.8 the algorithm functions by detecting points between minority class samples, the points being determined by the  $k$ -nearest neighbour algorithm. This ensures that the newly generated data will be similar to the already existing data of the minority class. This may prove useful when developing the Generative Adversarial Network to create synthetic COVID-19 X-Rays.

ADASYN works in a similar way to SMOTE and was originally based on SMOTE. Both function in much the same way but the key difference lies in ADASYN adding a random small bias value to the points, breaking linear correlation to their parents. The bias that ADASYN adds helps to increase the amount of variance within the synthetic data. In this research paper the authors decided to evaluate the performance of the GANs ability to generate synthetic numerical data in two domains, one domain is concerned with training a classifier which was trained only using synthetic data and the other domain to created a balanced data set by oversampling the minority class using synthetic data. The first domain's performance was measured by comparing the overall performance of the classifier on the original dataset with the overall performance of the classifier on the synthetic data set which were created by variations in the GAN architecture. The researchers gauged the performance of the second domain by comparing the clasifier's performance on an imbalanced dataset oversampled with a standard GAN, SMOTE, and ADASYN as well as the original data set which is not oversampled. SMOTE and ADASYN produce desirable results but the drawback is that they do not generalize well with sparse data and outliers according to the researchers.

In both experimental domains, the researchers used the following GAN architecture to generate the synthetic data:

- Leaky ReLU as activation function with a negative slope of 0.2
- batch size of 5
- learning rate of  $2 \times 10^{-4}$
- use of dropout in the GAN generator with a probability of 0.3
- Binary cross-entropy as loss function
- Adam as the optimizer
- No convolution layers
- If the generator has more than one layer, they are ordered in ascending size
- In the discriminator, layers are ordered in descending size if there is more than 1 layer

[36] They also used the following architectures to generate the data:

Data Set Name	Architecture of GAN
Original Data	The first 70% of the original database
256/512/1024	Generated by a GAN with 3 hidden layers with size 256, 512 and 1024
256/512	Generated by a GAN with 2 hidden layers with size 256 and 512
256	Generated by a GAN with 1 hidden layer with size 256
128/256/512	Generated by a GAN with 3 hidden layers with size 128, 256 and 512
128/256	Generated by a GAN with 2 hidden layers with size 128 and 256
128	Generated by a GAN with 1 hidden layer with size 128

Table 2.3: GAN Architectures used for experiments in[36])

choices in the above architecture are standard within the literature in this area. The performance of the GAN was then tested on 3 data sets which are listed below.

- Pima Indians Diabetes data Database
- Breast Cancer Wisconsin Data Set (Diagnostic)
- Credit Card Fraud Detection

Using these data sets the researchers conducted a number of experiments to judge the performance of data augmentation when testing the classifier. The first experiment involved the training of a classifier which was trained on synthetic data generated by the GAN. The GAN was trained on the original data set for 1500 epochs. After training, the GAN was then used to generate synthetic data containing the same amount of data as the original data set. The classification label used by the GAN is a continuous value between 0 and 1, the value is then made discrete (either 0 or 1) which is determined by the process of rounding the integer to the nearest value. The synthetic data generated was then utilized to train a classification tree and the tree was then tested by using a test subset which was created from the original data set. The GAN was trained using labeled class data also, this allows the synthetic data to have any class and the GAN will determine of which class the data is a member. Tests for experiment one were conducted on both the diabetes and cancer data sets, these data sets were not very unbalanced in terms of classes. The findings of this experiment are visible in figure 2.9 where the researchers compared classes in the newly generated synthetic data with data in the original data set.

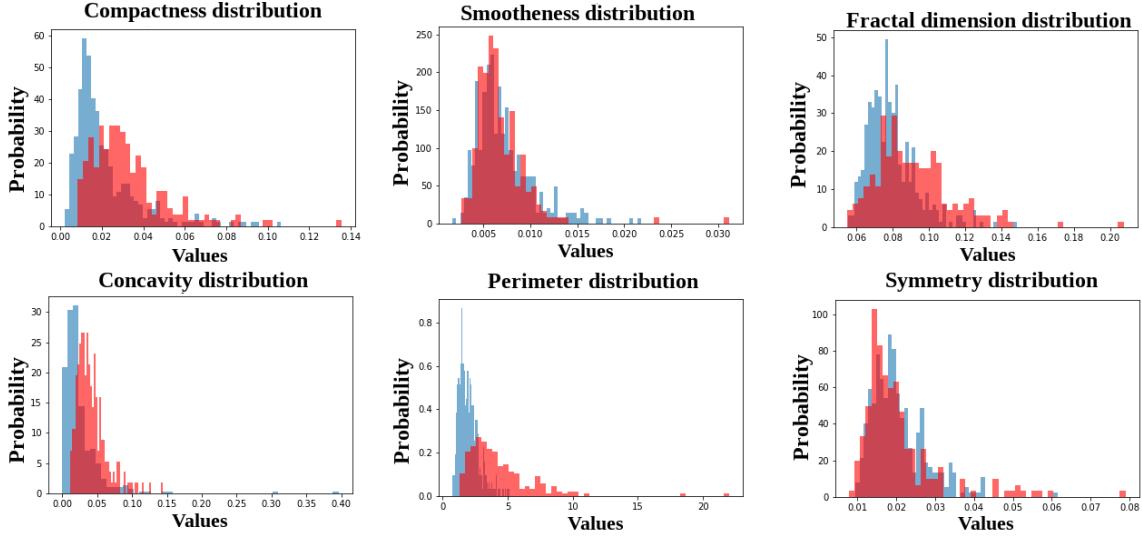


Figure 2.9: Results from Experiment One (Data Augmentation Using GANs)[36]

The blue region in the image 2.9 represents the distribution of features in the original data set where as the red represents the distribution of features in the newly created synthetic data. Figure 2.9 shows that the synthetic data has a much more varied distribution. From using the synthetic data the researchers were able to obtain the following results shown in table 2.4

Database	Label Proportion	Test Accuracy
Original Data Set	56.53/43.47	0.888
256/512/1024	52.26/47.74	0.818
256/512	56.28/43.72	0.941
256	56.78/43.22	0.906
128/256/512	54.02/45.98	0.953
128/256	58.04/41.96	0.935
128	54.27/45.73	0.912

Table 2.4: Results of Cancer data set using different GAN Architectures Experiment 1 (Data Augmentation using GANs)[36]

These results indicate that classifiers trained with the synthetic data generated by the various GAN architectures had superior results than the classifier trained with the original data set. The only exception to this was the GAN with an architecture of 256/512/1024 which showed a small reduction in test accuracy.

Database	Label Proportion	Test Accuracy
Original Data Set	64.8/35.2	0.748
256/512/1024	71.69/28.31	0.7
256/512	67.23/32.77	0.548
256	67.6/32.4	0.748
128/256/512	60.15/39.85	0.661
128/256	65.18/34.82	0.739
128	54.27/45.73	0.697

Table 2.5: Results and label distribution of Diabetes data set using different GAN Architectures Experiment 1 (Data Augmentation using GANs)[36]

The results of classifiers trained on the GAN architectures did not perform as well as those trained on the original data set when using the Diabetes data set. As is shown from the table above the classifiers trained using synthetic data struggled to match the performance of classifiers trained with the original data set. The GAN with the architecture of 1 hidden layer consisting of 256 units did manage to match the classifier trained on the original data set in terms of performance, the rationale for using the synthetic data would therefore be to increase the model's ability to generalise when encountering new data.

In the second experiment conducted by the researchers, they tested the oversampling of the minority class in the data set by using both SMOTE and ADASYN. The study proceeded as follows: The training set was separated based on the class of the target. The GANs were then trained only on minority-class data. The GAN was then used to add new synthetic data to the data set thus increasing the number of instances of the minority class in the dataset, making this minority class more numerous within the data. The researchers augmented the dataset with this new synthetic data creating new instances of the minority class until the data set was balanced. The synthetically augmented data set was then used when training a new classifier. The new classifier was then tested on both data sets, the original set, and the newly created synthetically augmented version which was created by allowing the majority class to be undersampled. The results obtained from this experiment are shown in table 2.6

Database	Accuracy	Precision	Recall
Original	0.999	0.896	0.556
SMOTE	0.958	0.026	0.861
ADASYN	0.958	0.026	0.861
128	0.798	0.051	0.806
256	0.986	0.077	0.789
128 / 256	0.974	0.045	0.82
256 / 512	0.964	0.033	0.808

Table 2.6: Classification results on imbalanced test set experiment 2 (Data Augmentation using GANs)[36]

Database	Accuracy	Precision	Recall
Original	0.782	1.0	0.565
SMOTE	0.912	0.959	0.861
ADASYN	0.921	0.979	0.861
128	0.807	0.89	0.806
256	0.894	0.998	0.789
128 / 256	0.902	0.981	0.82
256 / 512	0.888	0.962	0.808

Table 2.7: Classification results on balanced test set experiment 2 (Data Augmentation using GANs)[36]

It's clear from the tables above the use of SMOTE and ADASYN underperformed in accuracy on the imbalanced test set but had higher recall when compared with the original. In the second table, when tested on the balanced test set, SMOTE and ADASYN outperformed the original in terms of accuracy and recall. This is due to the original data set being imbalanced, the tree trained on this data set predicts almost all samples as negative. The GAN using a hidden layer of 128 performed very poorly in both instances when compared with other GAN architectures and this is called out by the researchers in the paper. Oversampling the minority seemed to increase the recall score of the classifier but at the expense of precision. The results shown above will prove useful in guiding the development of the architecture used in the GANs to generate synthetic COVID data, and to over sample the minority of the classes within the various datasets I plan on using to train the classifier, this will be discussed further in later chapters.

Through this research, the researchers found that when dealing with very unbalanced test sets, the GAN outperformed both SMOTE and ADASYN when it came to accuracy and precision but had a lower overall recall score. Depending on the context of the problem domain

accuracy and precision may be preferred over recall or recall may be preferred over accuracy and precision.

When developing the GANs for generating synthetic data to train COVID-19 classifier, there are questions that will need to be answered when deciding to use SMOTE and ADASYN over a traditional GAN architecture. A higher accuracy and precision would be useful in diagnosing COVID-19 in all patients while reducing the risk of unnecessary quarantining due to false positives. However, a higher recall would yield a higher overall identification of COVID positive patients but at the expense of precision, this would help to reduce the transmission of the virus but at the expense of causing unnecessary quarantines of patients.

There are a few limitations the researchers of this paper addressed in this study. The research was conducted using only three data sets it is unclear if the results found in the paper will be similar when utilizing other data sets. There are also many other considerations to take into account when using GANs, such as mislabelled data, the size of the dataset, amongst other factors which may influence the creation of the synthetic data.

In another paper, by Wang and Xiao[39] a convolutional neural network was employed in order to discern defects in harvested lychee fruit. The data set used was then augmented with synthetic data generated with a GAN. To train the classifier and the GAN, researchers created a data set of 3743 samples which were divided into 3 categories: mature, defects, and rot. The data set created by the researchers suffered from an imbalance much like the data set in the paper previously discussed[36]. To address the imbalance within the data set the researchers used a transformer-based GAN to augment the data and create a more diverse and balanced data set which was used to train the classifier to classify the lychee fruit. The researchers created a number of deep convolutional neural network models which incorporated a number of architectures such as: SSD-MobileNet V2, Faster RCNN-ResNet50, and Faster RCNN-Inception-ResNet V2. The models were trained with different hyper-parameters to evaluate and contrast their performance. The researchers found from the evaluations of the models that the data augmentation did in fact increase the performance of the classifiers.

There is much need for automation within this particular domain as human fatigue can affect the classification of lychee fruit and incorrect classification of lychee is costly to businesses both in monetary terms and in terms of reputation. This problem domain within artificial intelligence has been extensively studied, commonly used methods for detecting defects within fruit are: region growing method, minimum outer rectangle method, threshold segmentation, edge detection, *k*-mean clustering, and contour finding[39]. Recent progress made in the field of deep learning has demonstrated superior results in a wide range of computer vision tasks among other tasks. Through the use of a DCNN the authors of this paper [39] hope to show superior performance in comparison to traditional machine learning methods currently in use. The researchers decided to use samples of black leaf lychee which were purchased from the Jiangbei Fruit Wholesale Market in Huzhou city which is located in Guangdong, China when

creating the data set, the black leaf lychee were purchased in batches of two. The first batch contained 2042 mature lychees, which contained 1216 sample which had cracks but had no signs of rot. To gather rot samples for the data set the researchers stored the 625 cracked lychee samples in an environment which was dry and they were stored at room-temperature so they rotted naturally as opposed to artificially. 495 additional samples were rotted and dried by being exposed to sunlight. The second batch contained 865 lychees that were utilized as a test sample to ensure the tests were not biased. The two batches of lychee were then stored in boxes alongside ice packs and shipped back to the researcher's lab to maintain the fruit's freshness while in transit. Data diversity was accomplished by gathering lychee images at six locations in Huizhou College at various times, the images were also taken from different angles. A total of 5014 images were collected for the data set by the researchers.

When training the neural network the researchers synthetically augmented the dataset in an attempt to reduce the overall risk of overfitting the model and to improve the model's generalisation ability by exposing the model to a far larger dataset. The training set was augmented with synthetic data generated by the GAN below is a table of the distribution of categories from the original data set which the researchers were using to train the model.

Category	Original	Original Percentage	Training	Test	Generated	Augmented Training set
Mature	1648	44.03%	1298	350	102	1400
Defects	964	25.75%	800	164	600	1400
Rot	1331	30.22%	896	235	504	1400
Total	3743	100.00%	2994	749	1206	4200

Table 2.8: Comparison of distribution of data augmented vs original(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

As shown in the table above the augmented data offers a much more balanced dataset with all classes of lychee being represented in equal proportion. Classifiers trained on the original dataset may have possibly created a model which would create a high bias for the most represented class of lychee in the dataset.

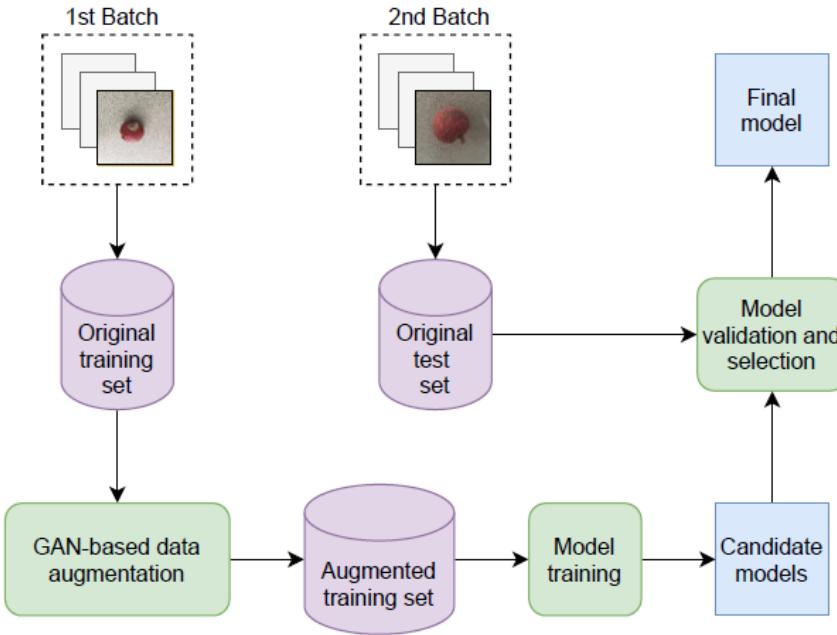


Figure 2.10: Figure of learning framework for lychee Classification Model[39]

Figure 2.10 shows the overall configuration of how the models for lychee surface defect detection were trained. The original training set was used to train the GAN and from the newly augmented training set the DCNN models were trained to detect defects. After the training of the DCNN models they were validated on a test set to compare each model's performance in an unbiased manner.

In this paper, the researchers decided to use a variation of GAN known as TransGAN. This version of a GAN is based on transformers and did not use convolutions. This version of a GAN consisted of a transformer encoder which is made up of a "multi-head self-attention module stacked by a feed-forward multilayer perceptron"[39]. This version of a GAN has traditionally been used for natural language processing but has also found applications in computer vision. In this version of a GAN the generator  $G$  alongside the discriminator,  $D$  are created using transformer encoder blocks. TransGAN has a multi-stage mechanism that will adjust the image resolution by upscaling and downscaling the image accordingly to prevent excessive memory consumption. There is also a grid self-attention module which is contained within the first partition, this is used to reduce the computational load. TransGAN has shown superior results in generative modeling and hence was adopted by the researchers.[39][40]

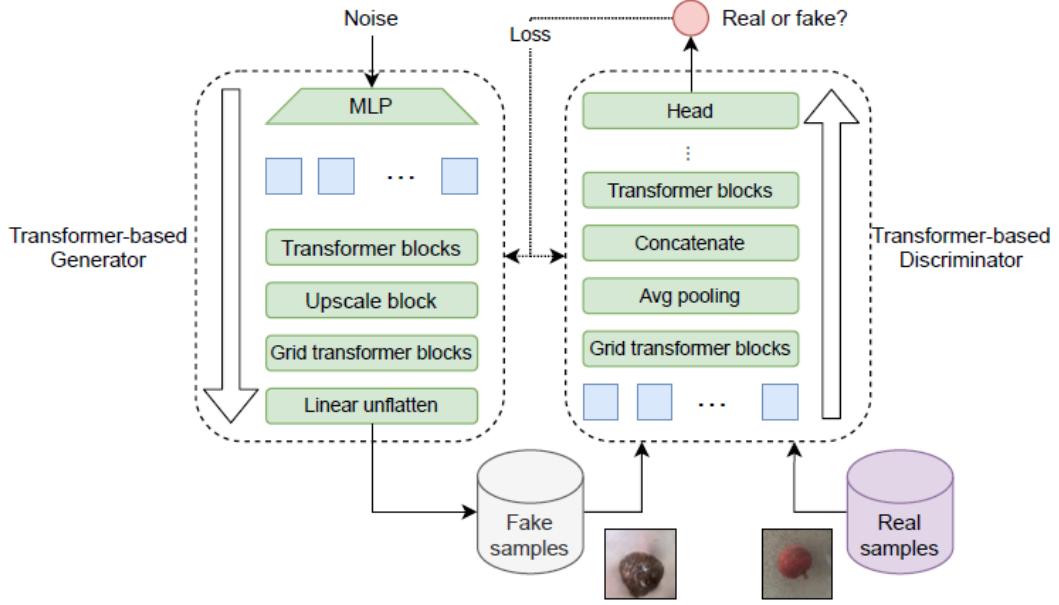


Figure 2.11: Figure of TransGAN (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

There are a variety of DCNNs used in this paper, we will show diagrams used by the researchers and explain each before further investigating the results of this paper as it's important to understand how each DCNN functions and compare each model's advantages and disadvantages.

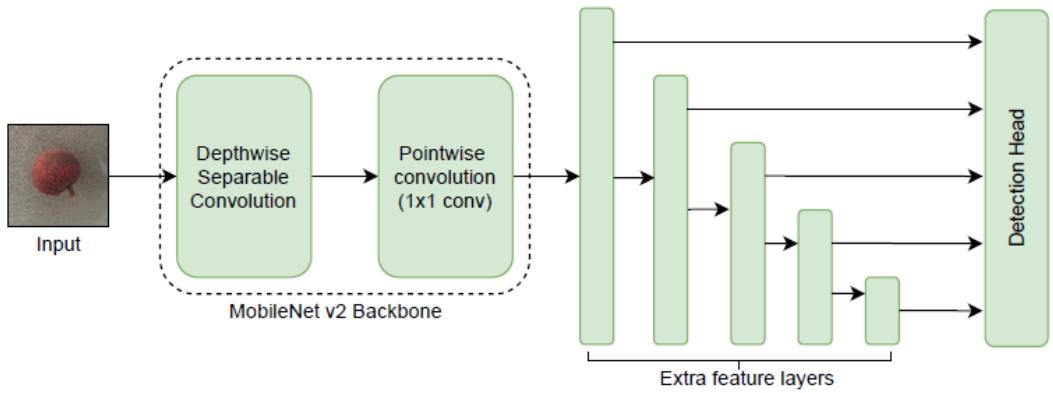


Figure 2.12: Figure of SSD-MobileNet V2 Architecture (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

Figure 2.12 shows an example of an SSD-MobileNet V2 DCNN as shown the input goes through a depthwise separable convolution and a pointwise convolution. The goal of SSD is to

be able to perform object localization alongside object classification in only one forward pass of the network. SSD uses multi-scale feature mapping which allows the neural network which mimics the human eye when detecting and classifying objects. MobileNet is a lightweight deep neural network which has been identified as being efficient at performing a number of tasks. The model consists of two hyper-parameters which include both, a resolution multiplier, and a width multiplier. These hyperparameters can be tuned to yield a higher latency or a higher accuracy for speed. Each convolutional layer uses both batch normalization alongside a ReLU activation function. The original version of the MobileNet (V1) consisted of an input layer, 13 convolutional layers, an average pooling layer and a fully connected layer[39]. In MobileNetV2 two new features were added these included a bottleneck which is linear between each layer and a connection shortcut between bottlenecks in the model which allowed for more efficient training.

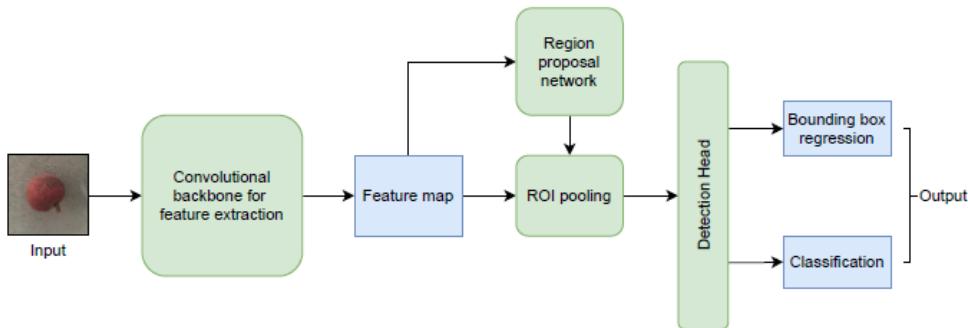


Figure 2.13: Figure of Faster RCNN Architecture (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

Figure 2.13 shows the architecture of an RCNN-ResNet50 DCNN, this model was originally proposed by Girshick et al in a 2014 paper [41]. This proposed DCNN model is utilized to perform search which is selective to extract 2000 regions from a given image, the regions extracted are termed as “region proposals”[41]. Candidate region proposals are converted into a square shape and then passed into a CNN that generates a 4096 dimensional feature vector, this feature vector is then input into an SVM(Support Vector Machine) to discern the data’s classification. This model is not suited for real time detection as the researchers found it took approximately 47 seconds to classify a single image, if using this architecture in the COVID-19 diagnostics CNN model there might possibly be a trade-off in terms of the time needed to classify data and the model’s accuracy. Due to the long time taken to classify an image, the researchers proposed a new architecture aptly termed ”Faster RCNN” which would eliminate the selective search and instead input the entire image into the CNN to produce the convolutional feature map. This architecture does not have to process 2000 proposals every time it classifies an image, which reduces the amount of time taken by the model to classify

the image.

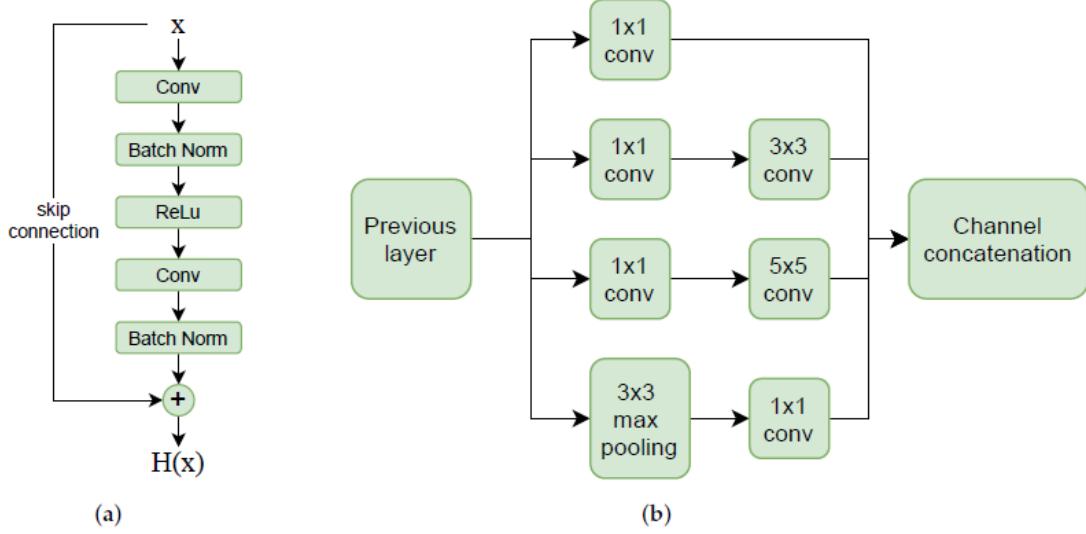


Figure 2.14: Figure of Faster RCNN Res Block and Inception Module (a) Res block; (b) Inception Module. (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

The final model proposed by the researchers, aims to include an inception model to offer "superior local topology for the neural network"[39]. The module (b) which is shown in figure 2.14 aims to perform several convolutional operations on the input image, each operation occurs in parallel, and combines all the results into a deep feature map. The model uses a numerous array of different filters to perform such convolutional operations on the input and obtains an array of information about the image. The methods used when designing this model allows for the model to gain a much deeper understanding of the image.

The researchers used the following configuration to train the TransGAN to generate synthetic data

- Learning rate of  $1 \times 10^{-4}$
- Adam as an optimizer
- batch size of 64 for both the generator and discriminator models
- The training ran for 220 epochs

The DCNNs were then trained on both the original and augmented training sets, training 6 different models in total. Every model which was trained took an image as input and then showed a box over the image for each detected lychee along with a predicted category along

with a score of how confident the model was in classifying the lychee. The hyperparameter settings for each model were as follows:

- Weight decay of  $5 \times 10^{-4}$
- Momentum of  $8 \times 10^{-1}$
- Verification Period of 5000
- Batch Size of 32
- Learning Rate of  $5 \times 10^{-3}$
- And ran for a total of 1500 epochs

The results before augmentation for the models are shown in the table below 2.9

Name of Model	Mean Average Precision
SSD-MobileNet V2	88.95%
Faster RCNN-ResNet50 V2	91.57%
Faster RCNN-Inception-ResNet V2	91.25%

Table 2.9: Results of models before Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

The researchers found that training on the GAN-augmented data that the performance increased by the following amounts for each of the models.

Name of Model	Performance Gain
SSD-MobileNet V2	2.86%
Faster RCNN-ResNet50 V2	1%
Faster RCNN-Inception-ResNet V2	0.58%

Table 2.10: Improvement of model's mean average precision after Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

As we can see from the above table 2.10 SSD-MobileNet V2 had the most gains in terms of performance. Due to the large imbalance between classes in the dataset the performance gap is quite large. The mean average precision performance gaps between classes before augmentation is shown in the table below 2.11

Name of Model	Mean Average Precision Performance Gap
SSD-MobileNet V2	9.45%
Faster RCNN-ResNet50 V2	6.12%
Faster RCNN-Inception-ResNet V2	7.77%

Table 2.11: Mean average precision before Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

After the augmentation process, the researchers found that the mean average precision performance gaps between classes were reduced for each of the three models to the values shown in table 2.12

Name of Model	Mean Average Precision Performance Gap
SSD-MobileNet V2	1.78%
Faster RCNN-ResNet50 V2	4.45%
Faster RCNN-Inception-ResNet V2	2.35%

Table 2.12: Mean average precision after Augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

As we can see from the above values the augmentation process yielded better quality models that can better differentiate between the three classes of fruit (rotten, defective, and mature). The model which has shown the most improvement in terms of mean average precision was Faster RCNN-ResNet50, however the researchers found that Faster RCNN-Inception-ResNet V2 was the most accurate in detecting rotten samples.

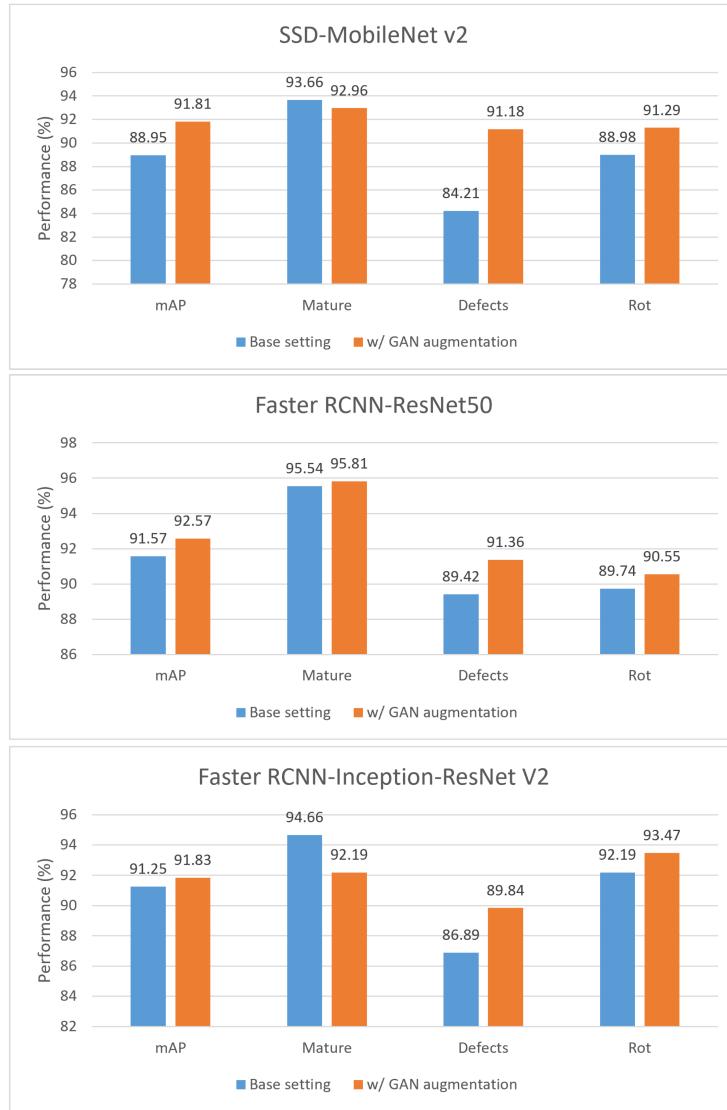


Figure 2.15: Figure of Mean Average Precision of Models. (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

Given that detecting defects in lychee is a time-sensitive job the researchers compared each of the three models in terms of detection speed. This is also a significant factor when developing the diagnostic models for detecting COVID-19 as the sooner the virus can be detected the sooner it can be treated effectively.

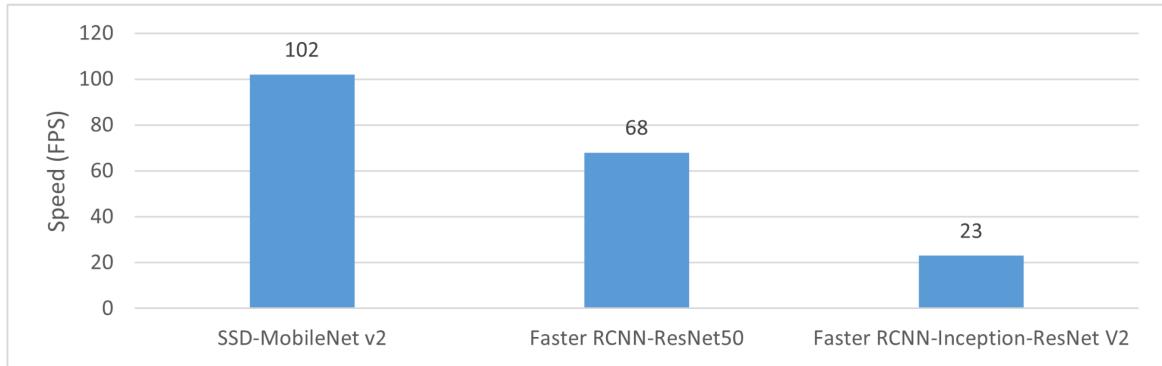


Figure 2.16: Figure of Speed of Models in classifying lychee. (lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

As we can see above SSD-MobileNet V2 classifies the lychee faster than the other two models although all 3 models met the researcher's requirement for lychee defect detection. These results are of particular interest and may prove useful when designing the automated diagnosis tool for COVID-19. SSD-MobileNet V2 could perhaps diagnose the patients faster than the medical professionals analyzing the patient, thus freeing up time for medical professionals to assist other patients.

I will list the accuracy of each of the models with and without data augmentation below to compare and contrast the classification performance.

Model	Setting	Acc	Rec	Spe	F1
SSD-MobileNet V2	Base setting	89.81%	90.08%	89.89%	89.46%
	GAN Augmentation	91.96%	92.06%	91.99%	91.92%
Faster RCNN-ResNet50	Base Setting	91.82%	92.23%	91.95%	91.72%
	GAN Augmentation	92.76%	92.96%	92.80%	92.55%
Faster RCNN-Inception-ResNet V2	Base Setting	91.96%	92.07%	91.98%	91.54%
	GAN Augmentation	92.36%	91.74%	92.22%	91.86%

Table 2.13: Comparison of accuracy of base models vs models with data augmentation(lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation)[39]

As shown in the above table 2.13 all of the models had better accuracy, recall(with the exception of Faster RCNN Inception model), specificity and F1 score with the data augmented data set. This shows that the classifiers were more accurate when classifying the fruit when they were trained on a more balanced data set.

## 2.5 Conclusion

From analyzing existing models for the automated detection of COVID-19 it appears that data quality and data shortage are key areas where improvements could be made to improve the overall accuracy and usability of the existing models. From the analysis of the current paradigms in convolutional neural networks and data augmentation of this thesis it is clearly shown that data augmentation has proven very useful in a wide range of applications which range from detecting defects in lychee to identifying credit card fraud. Given the positive results shown in the given problem domains, it seems a reasonable conjecture that the use of data augmentation would also prove useful for COVID-19 detection. In the next sections of this thesis, I will discuss the implementation methods used when creating both the convolutional models and the data augmentation models, the results of the models implemented, and further research which could be conducted into this area. It will be interesting to see if the findings of the papers explored above are transferable to this new problem domain.

## Chapter 3

# Design and Implementation

### 3.1 Introduction

Initially, when starting the development of this model, we looked at various tools and options to implement the model in code. We settled on using Jupyter Notebooks along with a number of libraries to help make the development of this model easier and faster. The useful thing about Jupyter Notebooks is that Jupyter Notebooks can be opened in a browser and all the code can be run from a single page. We will detail the development of our proposed model both in this thesis and include notes in the notebook itself to explain my rationale behind implementing the model in a certain way. During the initial phase of implementation, I used both the Keras documentation [12] and Tensorflow documentation [10] as references to ensure that the model's development was following standard practices and to ensure that the model was optimized to allow training in a timely manner.

Due to the limited support for AMD graphics cards (currently the GPU in our system has a 6700XT which does not have RoCM support[42]) in a variety of popular AI frameworks/libraries at the time of writing this thesis, we decided it was best to use Google Colab Pro when training both the CNNs and the GANs this may offer some limitations in terms of memory and computational power. Google Colab Pro, however, does offer a lot of advantages when it comes to quickly setting up an environment in which to train these models, it is for this reason that we have chosen to use it for training the models.

For the purpose of reproducible results, we included the following lines of code `np.random.seed(9)` and set the random seed of Keras to 10 so that other researchers can reproduce similar results (achieving the same results depends on a number of factors) and build upon this study. All the datasets are loaded and split using a seed of 1337 also so that the train/test split is the exact same every time. Each model also uses a train / validation / test split of 70% for train, 10% for validation and 20% for test(bar the chest X-ray COVID 19 dataset). The rationale for having double the amount of data for the test set when compared with the validation

set was made so that the test set could have more variety and therefore a higher degree of accuracy when comparing / contrasting the models.

## 3.2 CNN Model Design and Comparison

In this section we will compare and contrast each CNN's architecture and design when evaluating on both the original datasets and augmented datasets. The goal of this section is to determine which architecture works best when creating the automated diagnostic system and whether or not the augmented dataset is increasing the model's generalization ability and the model's accuracy. Initially when training the models we thought about using early-stopping to improve the accuracy and reduce the loss of these models, we decided against this due to the harm it may cause the model's generalization ability. All the models listed below use the entire data available when they are being trained.

### 3.2.1 Baseline Models

When starting the implementation phase, we decided to use the following resource to develop baseline CNN models [43]. We plan on modifying this resource to achieve a relatively high training/validation accuracy when training on the original dataset. We plan on using these models to get a metric with which we can compare models generated on the original dataset to the models which are generated on the synthetic dataset. It is in this way we can accurately compare the effects of the synthetic dataset on the accuracy of the implemented models.

After this initial comparison is done with the models trained on the original dataset versus the models trained on the synthetic dataset. We then plan on focusing on which architectures would work best when developing the CNN and how the models trained on the synthetic dataset can be improved.

To start we decided to use the following settings when developing a CNN to be used when training on the x-ray COVID-19 dataset. This dataset is made up of images that are labeled either 1 or 0 with 1 being COVID-positive and 0 being COVID-negative. We have included the architecture of the layers of the model in the table below 3.1

Layer Number	Layer Type	Layer Size	Kernel Size	Strides	Padding	Activation
1	Conv2D Layer	16	(3,3)	2	Same	Swish
2	SeparableConv2D Layer	32	(3,3)	None	Same	Swish
3	SeparableConv2D Layer	64	(3,3)	None	Same	Swish
4	MaxPooling2D	2	2	None	Same	None
5	Residual	64	(3,3)	2	Same	Swish
6	SeparableConv2D	128	(3,3)	None	Same	Swish
7	GlobalAveragePooling2D	1	None	None	None	Sigmoid

Table 3.1: X-ray COVID-19 dataset CNN baseline model architecture

For the padding the keyword "same" means that the input is padded with 0s evenly, both up and down and left and right of the image. The input was also scaled to normalize the data using the following line of code "1.0 / 255)(inputs)" After each layer batch normalization was performed excluding the residual, max pooling, and global average pooling 2D layers. The use of the activation function "swish" was chosen due to studies showing it's performance matched or outperformed ReLU for certain tasks[44]. Swish differs slightly in comparison to ReLU in that there isn't a sharp rise as the weight approaches 0.

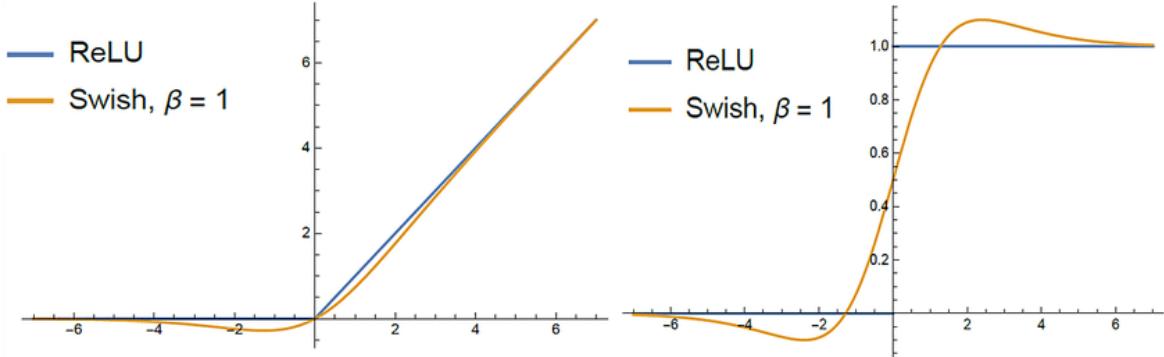


Figure 3.1: Figure of Swish and ReLU activation functions(Image courtesy of Madhura Ingahalikar)[45]

The model uses a dropout of 0, given the small size of the dataset we didn't want to drop neurons from the network. We also used the following settings when using `model.compile()`

Optimizer	Loss Function	Metric	Batch Size	Steps Per Epoch	Number of Epochs
Adam with a learning rate of $1 \times 10^{-3}$	Binary CrossEntropy	Accuracy	16	10	10

Table 3.2: X-ray COVID-19 dataset CNN baseline model hyperparameters

The model was trained for a total of 37 epochs with 1 step per epoch (again due to the limitations in the size of the dataset) and achieved the following results.

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Test Set Loss	Test Set Accuracy
0.2651	0.9122	0.7918	0.5000	0.8173	0.4688

Table 3.3: X-ray COVID-19 dataset CNN baseline model results

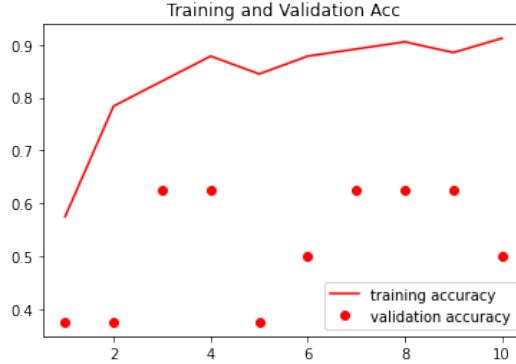


Figure 3.2: Figure of Train and Validation Accuracy of X-ray COVID-19 dataset CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 3.3: Figure of Train and Validation Loss of X-ray COVID-19 dataset CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the loss

As is shown from the results above 3.3 the model appears to be overfitting. The model has a training accuracy which is 0.4122 higher than the validation accuracy. The training loss is also much lower at 0.2651 when compared with the validation loss which is 0.7918. The model overfitting the training set is expected given the small size of the dataset. The model also performs very poorly on the test set and has a comparatively high loss and low accuracy when compared with the training set, which again is due to the limited data available.

After finishing the CNN implementation for the X-ray COVID-19 dataset we then moved on to designing the model with the radiography dataset. This dataset is much larger than the original dataset, the radiography dataset contains a total of 30,306 image files broken into three classes. In comparison, the X-ray COVID-19 dataset only contains 188 images belonging to two classes. When designing this Convolutional network more thought had to be given to the split and which activation function to use for output, given that there are multiple classes. When designing the CNN we decided to implement a much larger neural network given the amount of data available. We tried using the initial network which was used for the X-ray

COVID-19 dataset but the results were poor, increasing the size of the network led to better results. Due to computational limitations we have decided to opt for ReLU instead of Swish as an activation function.

Layer Number	Layer Type	Layer Size	Kernel Size	Strides	Padding	Activation
1	Conv2D Layer	64	(3,3)	2	Same	ReLU
2	SeparableConv2D Layer	128	(3,3)	2	Same	ReLU
3	SeparableConv2D Layer	256	(3,3)	2	Same	ReLU
4	SeparableConv2D Layer	512	(3,3)	2	Same	ReLU
5	MaxPooling2D	3	2	None	Same	None
6	Residual	512	(3,3)	2	Same	ReLU
7	SeparableConv2D	1024	(3,3)	None	Same	ReLU
8	GlobalAveragePooling2D	3	None	None	None	Softmax

Table 3.4: Radiography CNN baseline model architecture

Optimizer	Loss Function	Metric	Batch Size	Steps Per Epoch	Number of Epochs
Adam with a learning rate of $1 \times 10^{-3}$	sparse categorical crossentropy	Accuracy	32	663	10

Table 3.5: Radiography CNN baseline model hyperparameters

In this model we achieved a higher accuracy when using ReLU as opposed to swift. The final results of the model are as follows:

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Test Set Loss	Test Set Accuracy
0.3446	0.8611	0.4120	0.8306	0.3800	0.8434

Table 3.6: Radiography CNN baseline results

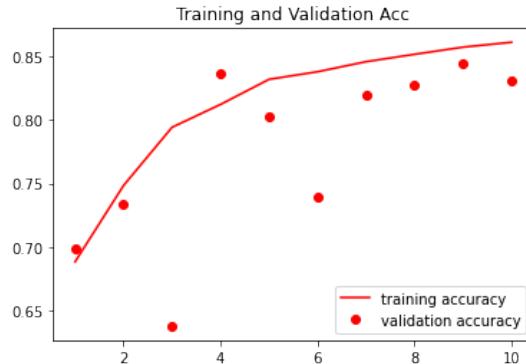


Figure 3.4: Train and Validation Accuracy of Radiography CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the accuracy

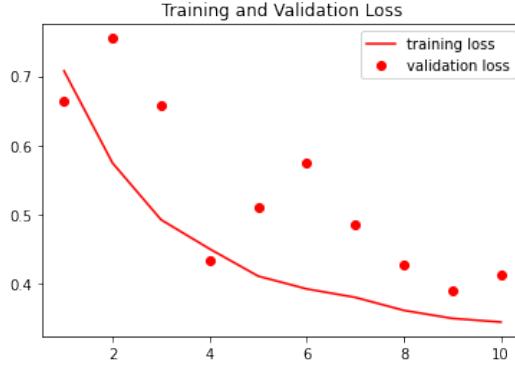


Figure 3.5: Figure of Train and Validation Loss of Radiography CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the loss

As shown in the table 3.6 above we can see that the model has better results when compared with the first baseline model. One of the reasons for this is that we are dealing with a much larger dataset so the model has more images that it can learn features from, this yields a much higher accuracy when compared with the first baseline model which was trained on a much more limited dataset.

The third baseline CNN model was trained using the COVID-19 chest X-ray dataset. This dataset didn't have a standardised resolution for images so the images had to be resized which could possibly lead to lack of data quality and consistency when resized. For these models no test set was included as the some classes had very little data and it was impossible to include all classes when randomly sampling the data, If data was manually included this would introduce bias so the test set was excluded from the model's evaluation. When training the model there was a high degree of validation loss which is to be expected given the dataset size. The architecture of this model is as follows:

Layer Number	Layer Type	Layer Size	Kernel Size	Strides	Padding	Activation
1	Conv2D Layer	32	(3,3)	2	Same	Swish
2	SeparableConv2D Layer	64	(3,3)	2	Same	Swish
3	SeparableConv2D Layer	128	(3,3)	2	Same	Swish
4	SeparableConv2D Layer	256	(3,3)	2	Same	Swish
5	SeparableConv2D Layer	512	(3,3)	2	Same	Swish
6	MaxPooling2D	3	2	None	Same	None
7	Residual	512	(3,3)	2	Same	Swish
8	SeparableConv2D	128	(3,3)	None	Same	Swish
9	GlobalAveragePooling2D	11	None	None	None	Softmax

Table 3.7: COVID-19 chest X-ray CNN baseline model architecture for COVID-19 Chest X-ray Dataset

Optimizer	Loss Function	Metric	Batch Size	Steps Per Epoch	Number of Epochs
Adam with a learning rate of $1 \times 10^{-3}$	categorical crossentropy	Accuracy	4	79	10

Table 3.8: COVID-19 chest X-ray CNN baseline model hyperparameters for COVID-19 Chest X-ray Dataset

Due to the small size of the dataset the batch size was set to a low number. The steps per epoch and number of epochs were also relatively low when compared with the other datasets due to the limited amount of data present. The model's performance is shown in the table below:

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
0.8372	0.7975	7.2848	0.6571

Table 3.9: COVID-19 chest X-ray CNN baseline model results for COVID-19 Chest X-ray Dataset

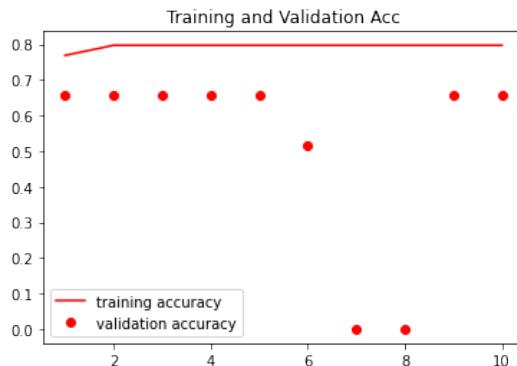


Figure 3.6: Figure of Train and Validation Accuracy of COVID-19 chest X-ray CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the accuracy

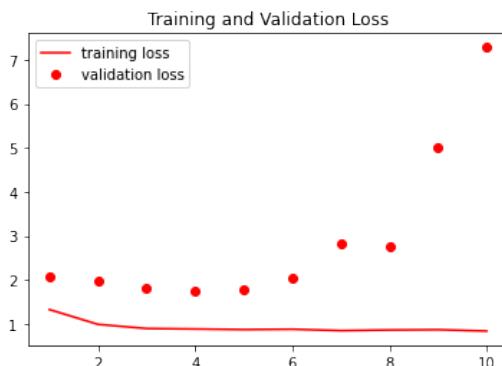


Figure 3.7: Figure of Train and Validation Loss of COVID-19 chest X-ray CNN Baseline Model - The X-Axis shows the epoch number and the Y-axis shows the loss

As shown in table 3.9 the model has a high degree of loss and the accuracy isn't very good on either the training or the validation set. This is to be expected as the dataset is quite small and contains a large number of classes. The model suffers from insufficient data and has performed poorly due to data imbalance as well as lack of standardised image resolutions. The fourth and fifth baseline models were trained using the Extensive COVID Dataset. This dataset is comprised of two different categories of images, one category of images being X-ray images and the other being CT scans. Two different CNNs were trained, one was trained using the X-ray images and the other using the CT images. The first baseline model for the Extensive COVID Dataset was trained on the CT images and the second baseline model was trained using the X-ray images.

When creating the baseline model we trialled a number of different architectures the best performing of these architectures is shown in table 3.10

Layer Number	Layer Type	Layer Size	Kernel Size	Strides	Padding	Activation
1	Conv2D Layer	32	(3,3)	2	Same	ReLU
2	SeparableConv2D Layer	64	(3,3)	2	Same	ReLU
3	SeparableConv2D	128	None	None	Same	ReLU
4	SeparableConv2D	256	(3,3)	None	Same	ReLU
5	SeparableConv2D	512	(3,3)	None	Same	ReLU
6	residual	512	(3,3)	2	Same	ReLU
7	SeparableConv2D	1024	(3,3)	None	None	ReLU
8	activation layer	1	None	None	None	Sigmoid

Table 3.10: Extensive COVID-19 CT Dataset CNN baseline model architecture

Optimizer	Loss Function	Metric	Batch Size	Steps Per Epoch	Number of Epochs
RMSprop with a learning rate of $1 \times 10^{-3}$ and a momentum of $1 \times 10^{-3}$	binary crossentropy	Accuracy	32	177	10

Table 3.11: Extensive CT CNN baseline model hyperparameters

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Test Set Loss	Test Set Accuracy
0.2286	0.9023	0.3837	0.8150	0.3574	0.8319

Table 3.12: Extensive CT CNN baseline model results

The training / validation accuracy and loss for this model are visible in the images included below

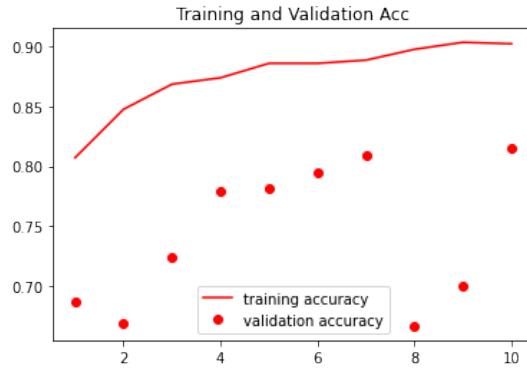


Figure 3.8: Figure of Train and Validation accuracy of Extensive COVID CNN Baseline Model CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy

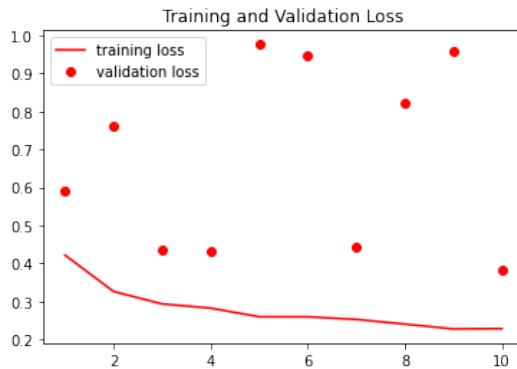


Figure 3.9: Figure of Train and Validation Loss of Extensive COVID CNN Baseline Model CT - The X-Axis shows the epoch number and the Y-axis shows the loss

When creating the baseline model we trialled a number of different architectures but found the following architecture detailed in table 3.13

Layer Number	Layer Type	Layer Size	Kernel Size	Strides	Padding	Activation
1	Conv2D Layer	32	(3,3)	2	Same	ReLU
2	SeparableConv2D Layer	64	(3,3)	2	Same	ReLU
3	SeparableConv2D	128	None	None	Same	ReLU
4	SeparableConv2D	256	(3,3)	None	Same	ReLU
5	SeparableConv2D	512	(3,3)	None	Same	ReLU
6	residual	512	(3,3)	2	Same	ReLU
7	SeparableConv2D	1024	(3,3)	None	None	ReLU
8	activation layer	1	None	None	None	Sigmoid

Table 3.13: Extensive COVID-19 X-Ray CNN Baseline Model Architecture

Optimizer	Loss Function	Metric	Batch Size	Steps Per Epoch	Number of Epochs
RMSprop with a learning rate of $10^{-3}$ and a momentum of $10^{-3}$	binary crossentropy	Accuracy	32	209	10

Table 3.14: Extensive COVID-19 X-Ray CNN baseline model hyperparameters

Training Loss	Training Accuracy	Validation Loss	Validation Accuracy	Test Set Loss	Test Set Accuracy
0.1948	0.9278	0.3482	0.8587	0.3203	0.8698

Table 3.15: Extensive COVID-19 X-Ray CNN baseline model results

The training / validation accuracy and loss for this model are visible in the images included below

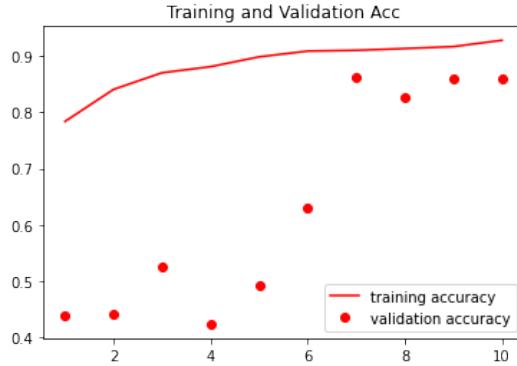


Figure 3.10: Figure of Extensive COVID-19 X-ray CNN Baseline Model Train and Validation Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

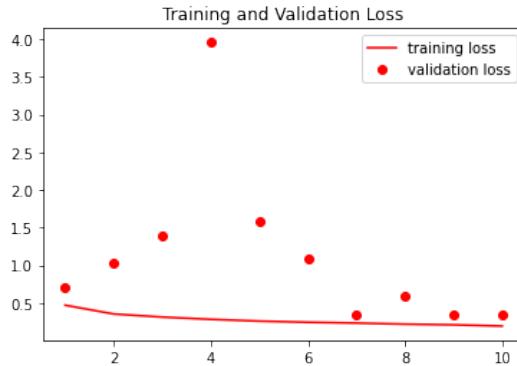


Figure 3.11: Figure of Extensive COVID-19 X-ray CNN Baseline Model Train and Validation Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### 3.3 Transfer Learning CNN Baseline Models

Next we will compare the transfer learning baseline models. Keras offers a large number of pretrained models[46]. For the sake of time we have chosen the following three pretrained

models to compare and contrast their effectiveness when automating COVID-19 diagnosis, Xception, ResNet50V2, and EfficientNetV2S were chosen as the architectures. When choosing the pretrained models we had to carefully consider both their performance and size for use in this project. Some models had parameters in excess of 100 million parameters which would not be usable in Colab Pro due to computational resource limitations. Thus the models chosen had a reasonable number of parameters to avoid crashes when training the CNNs, the model with the highest number of parameters was ResNet50V2 which has 25.6M trainable parameters and the other two models have approximately 20 - 25 million parameters. The weights used with all the transfer models come from models trained using the ImageNet dataset[47]. The ImageNet dataset is comprised of 14,197,122 images and contains 1,000 classes.

### 3.3.1 Radiography Dataset

#### Xception

This model contains 21,386,795 parameters including a layer of 256 ReLU units and an additional layer of 3 softmax units which were appended to the model. The model ran for a total of 10 epochs with 663 steps per epoch. The model also used sparse categorical cross entropy for the loss function and for the optimizer used Adam with a learning rate of  $1 \times 10^{-3}$ . The model achieved a final result of 0.9543 training accuracy with 0.1162 loss and had a validation accuracy of 0.8964 with a validation loss of 0.3867. The model also has a test set accuracy of 0.9001 alongside a test set loss of 0.3833. This is a relatively small but significant improvement in comparison to the original baseline models. Figure 3.12 shows the training and validation accuracy and figure 3.13 shows the training and validation loss.

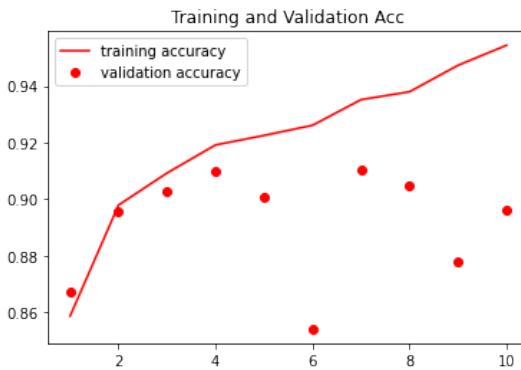


Figure 3.12: Transfer Learning Xception CNN Baseline Train and Validation Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

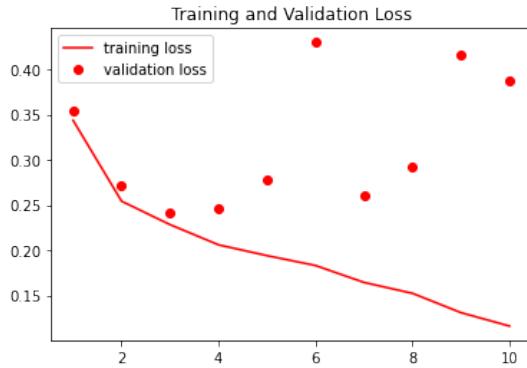


Figure 3.13: Transfer Learning Xception CNN Baseline Train and Validation Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### ResNet50V2

This model had a total of 24,090,115 parameters including a layer of 256 ReLU units and an additional layer of 3 softmax units which were appended to the model. This model had the same number of epochs and steps per epoch as exception(10 epochs with 663 steps) and also used the same optimizer with the same learning rate(Adam with  $1 \times 10^{-3}$  as the learning rate). The model performed slightly worse than Xception and finished with a training accuracy of 0.9172 and a training loss of 0.2024 and had a validation accuracy of 0.8695 with a validation loss of 0.3944. The model also has a test set accuracy of 0.8809 and a test set loss of 0.3942. Figure 3.14 shows the training and validation accuracy and figure 3.15 shows the training and validation loss.

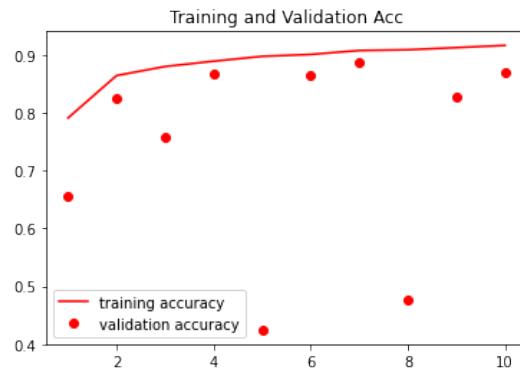


Figure 3.14: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

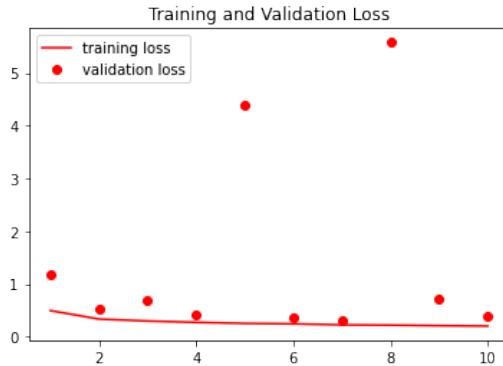


Figure 3.15: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

As shown in the figures above Xception outperforms this model when it comes to both the training accuracy and validation accuracy as well as the loss.

### EfficientNetV2S

EfficientNetVS2 is a model with a total of 20,660,067 parameters including a layer of 256 ReLU units and an additional layer of 3 softmax units which were appended to the model. Like the previous models this model was trained for a total of 10 epochs with 663 steps within each epoch. It uses Adam as an optimizer with a learning rate of  $1 \times 10^{-3}$ . The model performed worse than Xception but better than ResNet50V2 on the training,test, and validation sets. The model had a similar performance as Xception but managed to have a slightly lower train, test and validation accuracy and slightly lower losses for both the test and validation sets but a higher loss for the training set. The model has a training accuracy of 0.9383 and a training loss of 0.1519 and a validation accuracy of 0.8844 and a validation loss of 0.3362. The model also has a test loss of 0.3217 and a test accuracy of 0.8811. Figure 3.16 shows the training and validation accuracy and figure 3.17 shows the training and validation loss.

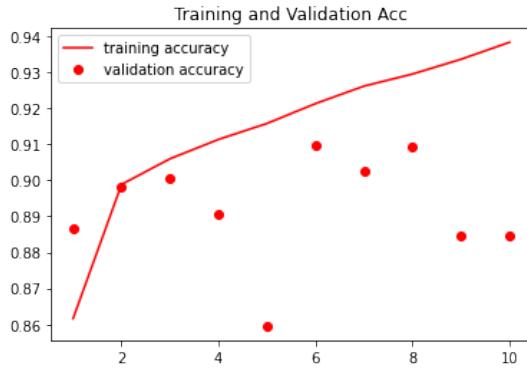


Figure 3.16: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy  
- The X-Axis shows the epoch number and the Y-axis shows the accuracy

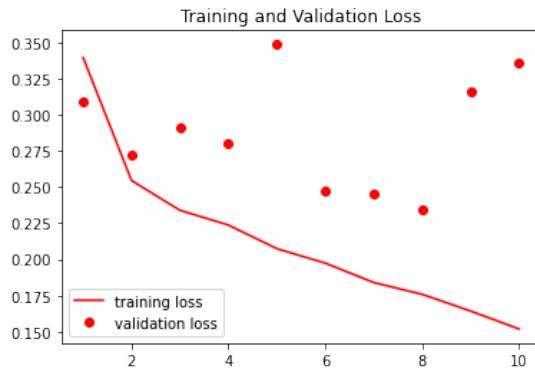


Figure 3.17: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss -  
The X-Axis shows the epoch number and the Y-axis shows the loss

### 3.3.2 X-Ray COVID-19 Dataset

#### Xception

This model has a total of 22,960,681 parameters including a layer of 1024 ReLU units and a layer of 1 Sigmoid unit which were appended to the model. The model uses binary crossentropy as a loss function and uses Adam as an optimizer with a learning rate of  $1 \times 10^{-3}$ . The model ran for a total of 10 epochs with 10 steps per epoch. The model achieved a training accuracy of 0.9662 with a training loss of 0.0951 and a validation accuracy of 0.5000 with a validation loss of 5.4697. The model also had a test set loss of 7.5530 and a test set accuracy of 0.3750. As with the baseline model this model is overfitting the training set which is to be expected given the small size of the dataset. Figure 3.18 shows the training and validation accuracy and figure 3.19 shows the training and validation loss.

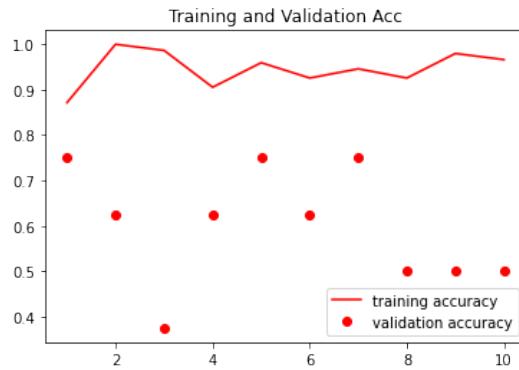


Figure 3.18: Transfer Learning Xception CNN Baseline Train and Validation Accuracy X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the accuracy

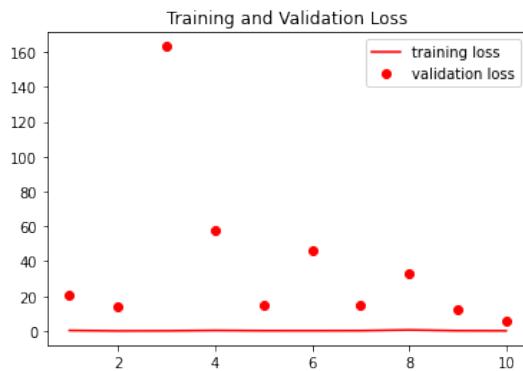


Figure 3.19: Transfer Learning Xception CNN Baseline Train and Validation Loss X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the loss

## ResNet50V2

This model has a total of 25,664,001 parameters with an additional layer of 1024 ReLU units and another layer of 1 sigmoid unit which was appended to the model. The model uses binary crossentropy as a loss function and Adam with a learning rate of  $1 \times 10^{-3}$  as an optimizer. The model ran for a total of 10 epochs with 10 steps per epoch and achieved a training accuracy of 0.9257 and a training loss of 0.2622 along with a validation accuracy of 0.3750 and a validation loss of 29.2193. The model also has a high test loss of 23.9440 and an accuracy of 0.4688. As with both the baseline and Xception models this model has overfit the training set. Figure 3.20 shows the training and validation accuracy and figure 3.21 shows the training and validation loss.

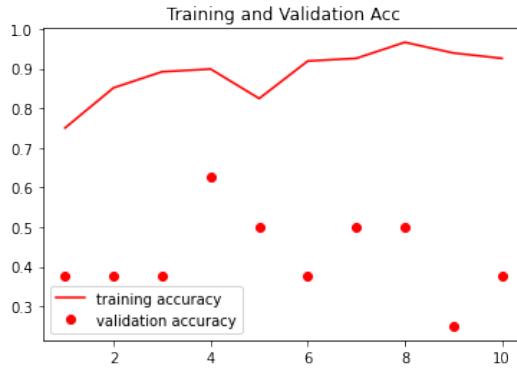


Figure 3.20: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the accuracy

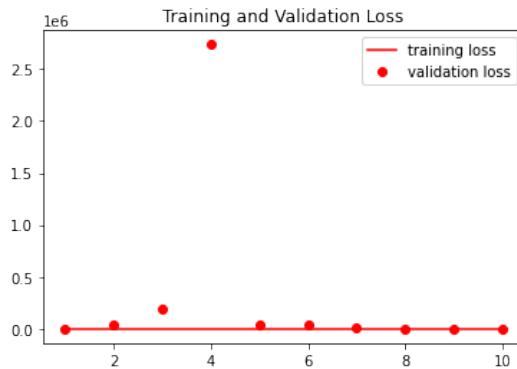


Figure 3.21: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the loss

## EfficientNetV2S

This model has a total of 21,644,129 parameters with an additional layer of 1024 ReLU units and another layer of 1 sigmoid unit. The model uses binary crossentropy as a loss function and Adam as an optimizer with a learning rate of  $1 \times 10^{-3}$ . The model runs for a total of 10 epochs with 7 step per epoch. The model achieved a training accuracy of 1.000 with a training loss of 0.0063 and a validation accuracy of 1.0000 and a validation loss of 0.0011. The model also achieved a test set loss of 0.2317 and a test set accuracy of 0.9375. Figure 3.22 shows the training and validation accuracy and figure 3.23 shows the training and validation loss. This model outperformed the baseline, Xception and ResNet50V2 models in terms of both loss and accuracy.

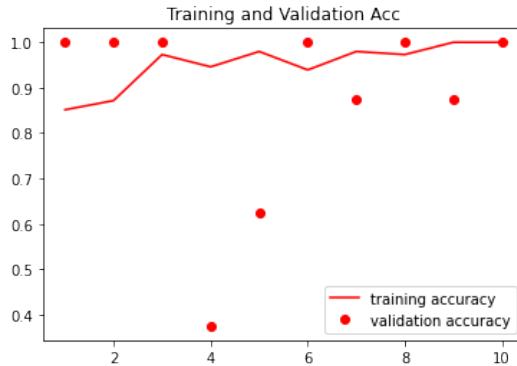


Figure 3.22: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the accuracy

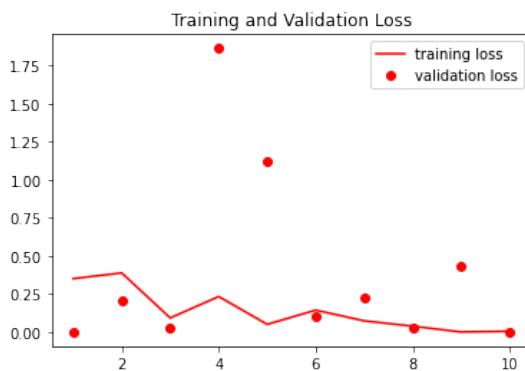


Figure 3.23: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss X-Ray COVID19 - The X-Axis shows the epoch number and the Y-axis shows the loss

### 3.3.3 Evaluation of TL models for The x-ray covid 19 dataset

The X-ray covid 19 dataset has a lack of data as it only comprises of 188 images. It is not surprising to see that the models are greatly overfitting the training set. Due to its limited size this dataset may not be suitable for training GANs or the output of said GANs may lack diversity.

### 3.3.4 COVID-19 Chest X-Ray Dataset

#### Xception

This model has a total of 22,970,931 parameters including an additional layer of 1024 ReLU units and another layer of 11 softmax units which were appended to the model. The model uses categorical crossentropy as a loss function along with Adam with a learning rate of  $1 \times 10^{-3}$  as an optimizer. The model runs for a total of 10 epochs with 79 step per epoch and achieved a training accuracy of 0.7975 and a training loss of 0.6109 along with a validation accuracy of

0.6571 and a validation loss of 1.6387. Figure 3.24 shows the training and validation accuracy and figure 3.25 shows the training and validation loss.

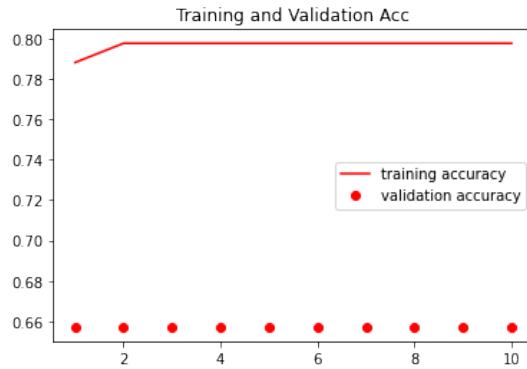


Figure 3.24: Transfer Learning Xception CNN Baseline Train and Validation Accuracy Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 3.25: Transfer Learning Xception CNN Baseline Train and Validation Loss X-Ray Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the loss

## ResNet50V2

This model has a total of 25,674,251 parameters including an additional layer of 1024 ReLU units and another layer of 11 softmax units which were appended to the model. The model uses categorical crossentropy as a loss function along with Adam with a learning rate of  $1 \times 10^{-3}$  as an optimizer. The model runs for a total of 10 epochs with 79 step per epoch and achieved a training accuracy of 0.7975 and a training loss of 0.9676 along with a validation accuracy of 0.6571 and a validation loss of 1.5587. Figure 3.26 shows the training and validation accuracy and figure 3.27 shows the training and validation loss.

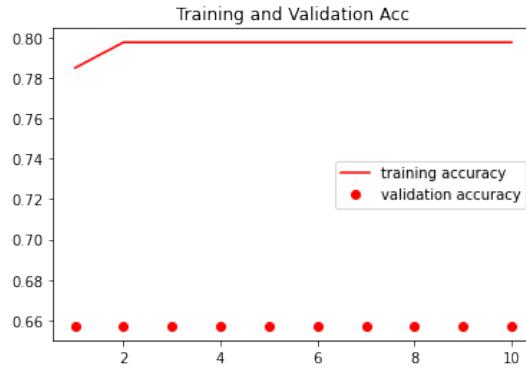


Figure 3.26: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy

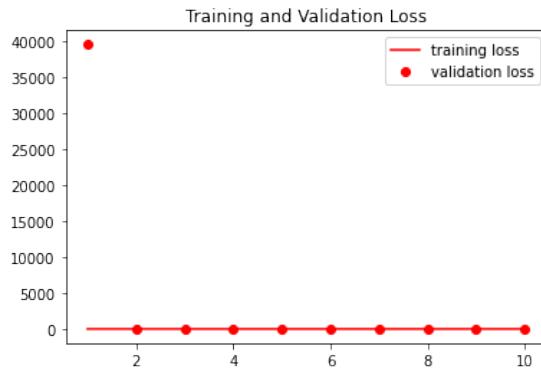


Figure 3.27: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss X-Ray Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the loss

### EfficientNetV2S

This model has a total of 21,654,379 parameters including an additional layer of 1024 ReLU units and another layer of 11 softmax units which were appended to the model. The model uses categorical crossentropy as a loss function along with Adam with a learning rate of  $1 \times 10^{-2}$  as an optimizer. The model runs for a total of 10 epochs with 79 step per epoch and achieved a training accuracy of 0.7975 and a training loss of 1.0280 along with a validation accuracy of 0.6571 and a validation loss of 1.7049. Figure 3.28 shows the training and validation accuracy and figure 3.29 shows the training and validation loss.

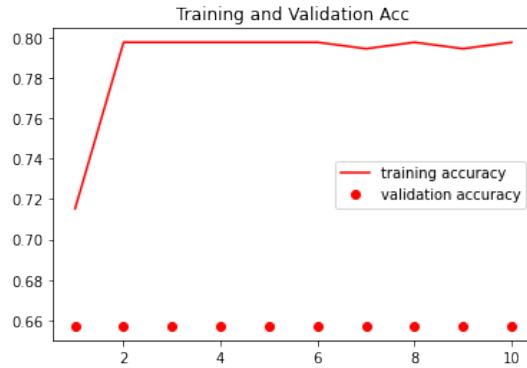


Figure 3.28: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 3.29: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss Chest X-Ray - The X-Axis shows the epoch number and the Y-axis shows the loss

### 3.3.5 Extensive COVID-19 Dataset CT

#### Xception

This model is comprised of 21,386,281 parameters in total including an additional layer of 256 neurons using ReLU activation and an output layer of 1 neuron using a sigmoid activation function. The model was trained for a total of 10 epochs with a total of 177 steps per epoch and uses RMSprop as an optimizer with a learning rate of  $1 \times 10^{-3}$  and a momentum of  $1 \times 10^{-3}$ . The model achieved a training accuracy of 0.9904 and a training loss of 0.0238 along with a validation accuracy of 0.9534 and a validation loss of 0.2250. The model also had a test set loss of 0.1771 and a test accuracy of 0.9537. Figure 3.30 and figure 3.31 shows the training and validation loss.

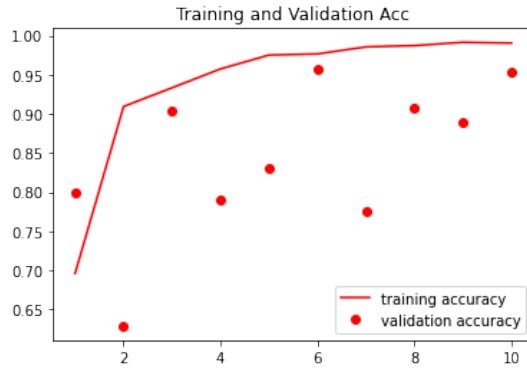


Figure 3.30: Transfer Learning Xception CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 3.31: Transfer Learning Xception CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the loss

## ResNet50V2

This model is comprised of 24,089,601 parameters in total including an additional layer of 256 neurons using ReLU activation and an output layer of 1 neuron using a sigmoid activation function. The model was trained for a total of 10 epochs with 177 steps per epoch and uses RMSprop as an optimizer with a learning rate of  $1 \times 10^{-3}$  and a momentum of  $1 \times 10^{-3}$ . The model achieved a training accuracy of 0.9604 and a training loss of 0.1140 along with a validation accuracy of 0.9350 and a validation loss of 0.2078. The model also has a test set accuracy of 0.9262 and a test set loss of 0.1993. Figure 3.32 shows the training and validation accuracy and figure 3.32 shows the training and validation loss.

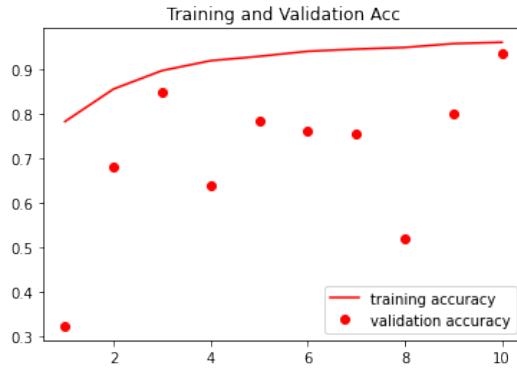


Figure 3.32: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 3.33: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the loss

### EfficientNetV2S

This model is comprised of 20,659,553 parameters in total including an additional layer of 256 neurons using ReLU activation and an output layer of 1 neuron using a sigmoid activation function. The model was trained for a total of 10 epochs with 177 steps per epoch and uses RMSprop as an optimizer with a learning rate of  $1 \times 10^{-3}$  and a momentum of  $1 \times 10^{-5}$ . The model achieved a training accuracy of 0.9901 and a training loss of 0.0291 along with a validation accuracy of 0.9375 and a validation loss of 0.2638. The model also has a test set accuracy of 0.9438 and a test set loss of 0.2352. Figure 3.34 shows the training and validation accuracy and figure 3.34 shows the training and validation loss.

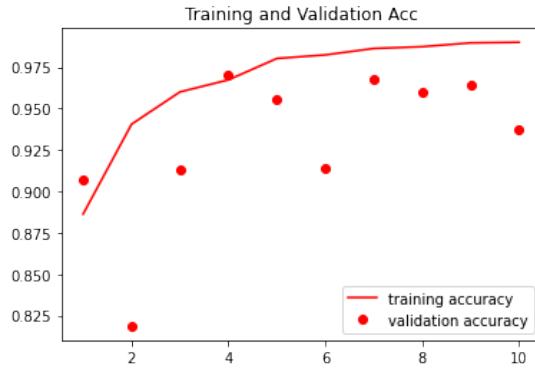


Figure 3.34: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the accuracy

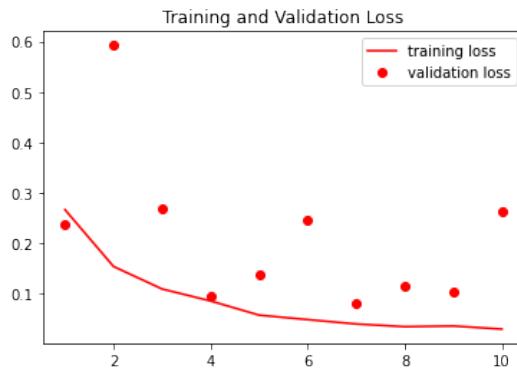


Figure 3.35: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset CT - The X-Axis shows the epoch number and the Y-axis shows the loss

### 3.3.6 Extensive COVID-19 Dataset X-ray

#### Xception

This model is comprised of 21,386,281 parameters in total including an additional layer of 256 neurons using ReLU activation and an output layer of 1 neuron using a sigmoid activation function. The model was trained for a total of 10 epochs with a total of 177 steps per epoch and uses RMSprop as an optimizer with a learning rate of  $1 \times 10^{-3}$  and a momentum of  $1 \times 10^{-3}$ . The model achieved a training accuracy of 0.9876 and a training loss of 0.0277 along with a validation accuracy of 0.9416 and a validation loss of 0.2814. The model also had a test set loss of 0.1947 and a test set accuracy of 0.9578. Figure 3.36 shows the training and validation accuracy and figure 3.36 shows the training and validation loss.

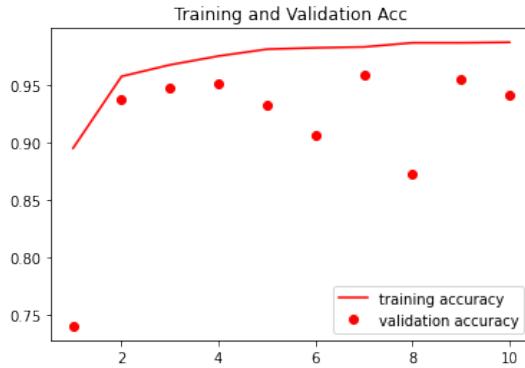


Figure 3.36: Transfer Learning Xception CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 3.37: Transfer Learning Xception CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the loss

### ResNet50V2

This model is comprised of 24,089,601 parameters in total including an additional layer of 256 neurons using ReLU activation and an output layer of 1 neuron using a sigmoid activation function. The model was trained for a total of 10 epochs with 209 steps per epoch and uses RMSprop as an optimizer with a learning rate of  $1 \times 10^{-3}$  and a momentum of  $1 \times 10^{-3}$ . The model achieved a training accuracy of 0.9649 and a training loss of 0.0907 along with a validation accuracy of 0.9139 and a validation loss of 0.2842. The model also has a test set accuracy of 0.9224 and a test set loss of 0.2379. Figure 3.38 shows the training and validation accuracy and figure 3.39 shows the training and validation loss.

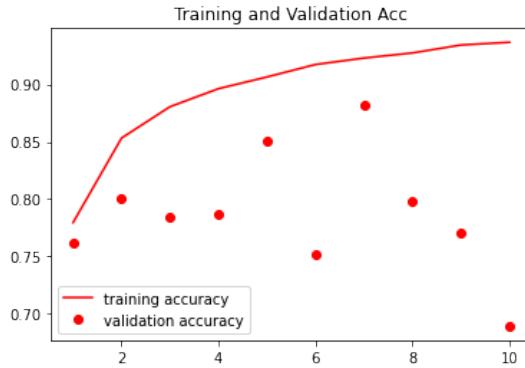


Figure 3.38: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy

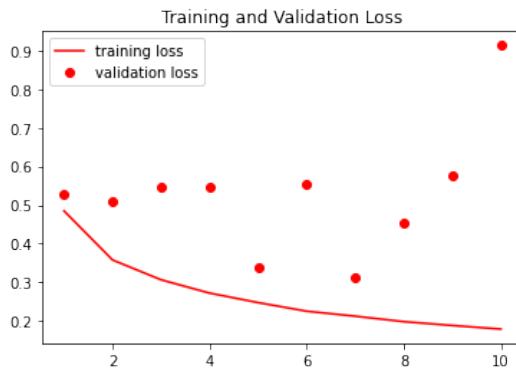


Figure 3.39: Transfer Learning ResNet50V2 CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the loss

### EfficientNetV2S

This model is comprised of 20,659,553 parameters in total including an additional layer of 256 neurons using ReLU activation and an output layer of 1 neuron using a sigmoid activation function. The model was trained for a total of 10 epochs with 209 steps per epoch and uses RMSprop as an optimizer with a learning rate of  $1 \times 10^{-3}$  and a momentum of  $1 \times 10^{-3}$ . The model achieved a training accuracy of 0.9840 and a training loss of 0.0396 along with a validation accuracy of 0.9702 and a validation loss of 0.2095. The model also has a test set accuracy of 0.9766 and a test set loss of 0.1513. Figure 3.40 shows the training and validation accuracy and figure 3.41 shows the training and validation loss.

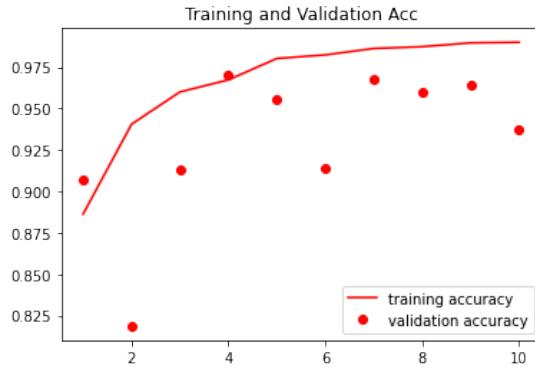


Figure 3.40: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Accuracy Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the accuracy

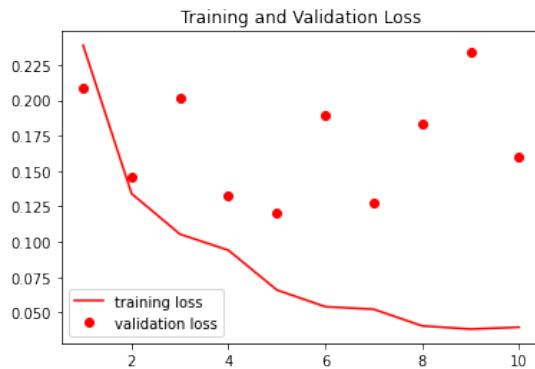


Figure 3.41: Transfer Learning EfficientNetV2S CNN Baseline Train and Validation Loss Extensive COVID 19 Dataset X-ray - The X-Axis shows the epoch number and the Y-axis shows the loss

### 3.4 GAN Baseline Design and Comparison

In this section we will detail the designs of the GANs and their architectures when augmenting classes in each database. We will also compare the different results and effects each GAN architecture had when producing synthetic data.

### 3.5 GANs for Radiography Dataset

Due to a large imbalance between classes in the dataset we decided to explore the use of GANs to create synthetic data for both the COVID positive images which comprise 7,232 images in this dataset and the Pneumonia positive images which comprise 2,690 images. The normal class (healthy patients) is very over-represented in the data as it is comprised of

20,384 images, due to this imbalance the CNNs trained from this dataset will be heavily biased towards identifying the normal patients. For this reason we have chosen to use a number of Generative Adversarial Architectures to synthetically augment the classes lacking in data to balance the dataset and increase the generalization and robustness of the CNN models.

### 3.5.1 VAE(Variational Auto Encoder)

A few issues were encountered when running the variational auto encoder models for the radiography dataset. When training on the masks we frequently ran into the issue of mode collapse, this is when the model only produces a single type of image. Mode collapse occurred with both the COVID and Pneumonia datasets and can be seen below

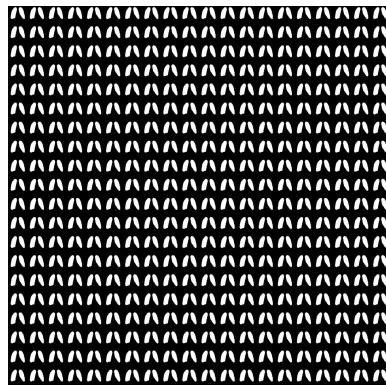


Figure 3.42: Example of Mode Collapse COVID Masks Radiography Dataset

There were also issues faced when using a VAE on the X-ray data of both COVID and pneumonia images, the VAE tended to not produce any results when trained upon these X-rays and instead only showed a blank grid as output. After a number of attempts to train both models it was found that the mask model suffered greatly from mode collapse and the X-ray model failed to produce any results for either class.

### 3.5.2 DCGAN(Deep Convolutional GAN Network)

#### COVID-19 Mask Class Augmentation

When designing the DCGAN we experimented with a number of architectures, some of these architectures led to the GAN only producing black squares which was a sign of mode collapse. Mode collapse occurs when the discriminator gets stuck at a local minimum and the generator learns to only produce the same type of image over and over again to fool the discriminator. We found switching from an ADAM optimizer to RMSPROP and experimenting with the learning rate and momentum led to far better results. The model produced some promising results, the following architecture was used to create the generator and discriminator:

```

1  # based off example:
2   ↵ https://keras.io/examples/generative/dcgan_overriding_train_step/
3  discriminator = keras.Sequential(
4      [
5          keras.Input(shape=(128, 128, 3)),
6          layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
7          layers.LeakyReLU(alpha=0.5),
8          layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
9          layers.LeakyReLU(alpha=0.5),
10         layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
11         layers.LeakyReLU(alpha=0.5),
12         layers.Flatten(),
13         layers.Dropout(0.2),
14         layers.Dense(1, activation="sigmoid"),
15     ],
16     name="discriminator",
17 )
18
19 # Create the generator.
20 generator = keras.Sequential(
21     [
22         keras.Input(shape=(latent_dim,)),
23         layers.Dense(8 * 8 * 128),
24         layers.Reshape((8, 8, 128)),
25         layers.Conv2DTranspose(256, kernel_size=4, strides=2,
26             ↵ padding="same"),
27         layers.LeakyReLU(alpha=0.2),
28         layers.Conv2DTranspose(512, kernel_size=4, strides=2,
29             ↵ padding="same"),
30         layers.LeakyReLU(alpha=0.2),
31         layers.Conv2DTranspose(1024, kernel_size=4, strides=2,
32             ↵ padding="same"),
33         layers.LeakyReLU(alpha=0.2),
34         layers.Conv2DTranspose(64, kernel_size=4, strides=2, padding="same"),
35         layers.LeakyReLU(alpha=0.2),
36         layers.Conv2D(3, kernel_size=5, padding="same",
37             ↵ activation="sigmoid"),

```

---

```

34     ] ,
35     name="generator",
36 )
37 generator.summary()

```

The design of this GAN was based off of a Keras tutorial and the code was refactored for the purposes of this project[48]. The following hyper parameters were used when training the DCGAN model to generate synthetic COVID-19 mask images:

Parameter	Value
Latent Dimension	100
Generator Optimizer	RMSProp
Discriminator Optimizer	RMSProp
Generator Learning Rate	$1 \times 10^{-5}$
Discriminator Learning Rate	$1 \times 10^{-5}$
Generator Momentum	0
Discriminator Momentum	0
Steps per Epoch	452
Batch Size	8
Number of Epochs	100

Table 3.16: DCGAN for Producing Synthetic COVID-19 Mask Data From Radiography Dataset

With this model architecture we was able to achieve a final loss of 0.5476 for the discriminator and 1.0187 for the generator.

### COVID-19 X-Ray Class Augmentation

Given the success achieved with the mask DCGAN we decided to reuse the architecture. The results at first were blurry and had little resemblance to the actual data. we decided to experiment with a number of different hyper parameters and found the parameters in the table below worked best:

Parameter	Value
Latent Dimension	256
Generator Optimizer	RMSProp
Discriminator Optimizer	RMSProp
Generator Learning Rate	$1 \times 10^{-5}$
Discriminator Learning Rate	$1 \times 10^{-5}$
Generator Momentum	0
Discriminator Momentum	0
Steps per Epoch	452
Batch Size	8
Number of Epochs	100

Table 3.17: DCGAN for Producing Synthetic COVID-19 X-Ray Data From Radiography Dataset

With this model architecture we was able to achieve a final loss of 0.6859% for the discriminator and 0.8015% for the generator.

### Pneumonia Mask Class Augmentation

The design of this DCGAN was based off the above COVID-19 DCGAN for generating masks and shares the same architecture the only difference being this model ran with 169 steps per epoch. The design yielded relatively similar results, although the losses for the pneumonia mask DCGAN was lower. The model finished training with a loss of 0.9474 and a loss of 0.5936 for the discriminator.

### Pneumonia X-ray Class Augmentation

The X-ray DCGAN for the pneumonia class is a copy of the above COVID-19 DCGAN the only difference being the Pneumonia X-Ray DCGAN uses a latent space of 256 and ran with 169 steps per epoch. The model achieved a final loss of 0.6901 for the discriminator and a loss of 0.7534 for the generator.

## 3.6 GANs for COVID 19 X-ray dataset

### 3.6.1 VAEs

The decision was made not to train variational auto-encoders on this dataset given the poor performance of VAEs on the radiography dataset which is a lot larger than the X-ray COVID-19 dataset. It appears that VAEs need much more data to train to recreate a similar image to the input data than the DCGAN models need.

### 3.6.2 DCGANs

There were some setbacks when training GANs on this dataset in particular due to the very limited amount of data it contained. To remind the reader this dataset only contains 94 images of COVID-19 Positive X-Rays and 94 images of Pneumonia X-Rays. When beginning the training of the GANs we merged both the train / test data for each class into two folders one marked Normal and the other marked Pneumonia each containing the 94 images of their respective class. We merged both the test and the training data into the two files previously mentioned in order to utilize all the data available for augmenting the respective class. Most researchers suggest a minimum of 50k to 100k images to train a high quality GAN as mentioned on NVIDIA's website[49]. We attained some success with this DCGAN model surprisingly and augmented both the normal and pneumonia classes with 1,000 new images.

The design of the DCGAN is as follows:

```

1 # based off example:
2   ↳ https://keras.io/examples/generative/dcgan_overriding_train_step/
3 discriminator = keras.Sequential(
4   [
5     keras.Input(shape=(128, 128, 3)),
6     layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
7     layers.LeakyReLU(alpha=0.5),
8     layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
9     layers.LeakyReLU(alpha=0.5),
10    layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
11    layers.LeakyReLU(alpha=0.5),
12    layers.Flatten(),
13    layers.Dropout(0.4),
14    layers.Dense(1, activation="sigmoid"),
15  ],
16  name="discriminator",
17 )
18 discriminator.summary()
19
20 # Create the generator.
21 generator = keras.Sequential(
22   [
23     keras.Input(shape=(latent_dim,)),
24     layers.Dense(8 * 8 * 128),
25     layers.Reshape((8, 8, 128)),

```

```

25   layers.Conv2DTranspose(256, kernel_size=4, strides=2,
26     ↪ padding="same"),
27   layers.LeakyReLU(alpha=0.2),
28   layers.Conv2DTranspose(512, kernel_size=4, strides=2,
29     ↪ padding="same"),
30   layers.LeakyReLU(alpha=0.2),
31   layers.Conv2DTranspose(1024, kernel_size=4, strides=2,
32     ↪ padding="same"),
33   layers.LeakyReLU(alpha=0.2),
34   layers.Conv2DTranspose(2048, kernel_size=4, strides=2,
35     ↪ padding="same"),
36   layers.LeakyReLU(alpha=0.2),
37   layers.Conv2D(3, kernel_size=4, padding="same", activation="tanh"),
38 ],
39 name="generator",
40 )
41 generator.summary()
42

```

The following hyper parameters were used to train the DCGAN to produce images from the normal class of the dataset

Parameter	Value
Latent Dimension	256
Generator Optimizer	RMSProp
Discriminator Optimizer	RMSProp
Generator Learning Rate	$1 \times 10^{-4}$
Discriminator Learning Rate	$1 \times 10^{-4}$
Generator Momentum	0
Discriminator Momentum	0
Steps per Epoch	47
Batch Size	2
Number of Epochs	100

Table 3.18: DCGAN for Producing Synthetic Normal Class Data for X-ray COVID19 dataset

The final model achieved a loss of 0.6890 for the discriminator and a loss of 0.7623 for the generator.

The augmentation of the pneumonia class had similar results as the normal class. Given the success of the DCGAN architecture for the normal class it made sense to reuse this

architecture when designing the DCGAN for the pneumonia class, the model also shares the same hyper-parameters. The final model had a discriminator loss of 0.6895 and a generator loss of 0.7757 which is roughly around the same as the DCGAN for the normal class.

### 3.7 GANs for Chest X-ray COVID-19

Due to this model having 11 classes and limited data for each class a decision was made to exclude this dataset from the research. This decision was made to conserve computational units and due to the limited data for each class. There are also a number of classes in this dataset which contain very little data and are grossly underrepresented, the data for these classes is extremely limited and it would not be worthwhile training a GAN using this data.

## 3.8 Extensive COVID-19 X-Ray / CT

### 3.8.1 VAEs

The Extensive VAEs were trained on using both the X-ray COVID positive class and the Non-COVID CT class. The output seemed to be fairly similar and it appears that the models suffered from mode collapse, but these images may have shown some benefit to the models. The decision to conserve computational limits was made and the models were not augmented with data from the VAE models. We have included images of the VAE models latent space below, as shown from the grids the resemblance between the images is striking although there may have been subtle differences non-descernible to the untrained eye.

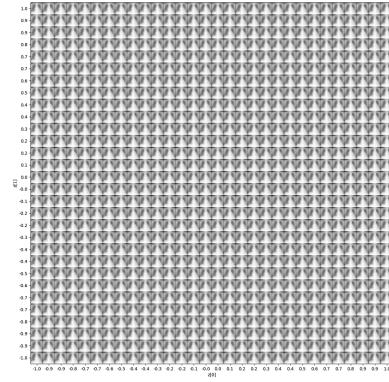


Figure 3.43: Possible Mode Collapse COVID X-ray Extensive Dataset

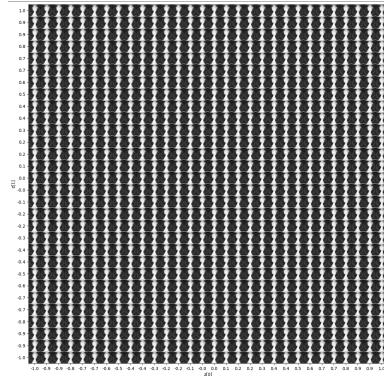


Figure 3.44: Possible Mode Collapse Non-Covid CT Extensive Dataset

### 3.8.2 DCGANs

#### X-ray DCGANs

The extensive COVID-19 X-ray models were broken up into two classes COVID and Non-COVID. The first model trained was trained to produce synthetic COVID images and ran for a total of 200 epochs and used the following architecture

```

1 # based off example:
2   ↳ https://keras.io/examples/generative/dcgan\_overriding\_train\_step/
3 discriminator = keras.Sequential(
4   [
5     keras.Input(shape=(128, 128, 3)),
6     layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
7     layers.LeakyReLU(alpha=0.5),
8     layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
9     layers.LeakyReLU(alpha=0.5),
10    layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
11    layers.LeakyReLU(alpha=0.5),
12    layers.Flatten(),
13    layers.Dropout(0.4),
14    layers.Dense(1, activation="sigmoid"),
15  ],
16  name="discriminator",
17 )
18 discriminator.summary()
19
20 # Create the generator.
21 generator = keras.Sequential(A

```

```

21   [
22     keras.Input(shape=(latent_dim,)),
23     layers.Dense(8 * 8 * 128),
24     layers.Reshape((8, 8, 128)),
25     layers.Conv2DTranspose(128, kernel_size=4, strides=2,
26       ↳ padding="same"),
27     layers.LeakyReLU(alpha=0.3),
28     layers.Conv2DTranspose(256, kernel_size=4, strides=2,
29       ↳ padding="same"),
30     layers.LeakyReLU(alpha=0.3),
31     layers.Conv2DTranspose(512, kernel_size=4, strides=2,
32       ↳ padding="same"),
33     layers.LeakyReLU(alpha=0.3),
34     layers.Conv2DTranspose(1024, kernel_size=4, strides=2,
35       ↳ padding="same"),
36     layers.LeakyReLU(alpha=0.3),
37     layers.Conv2D(3, kernel_size=4, padding="same", activation="tanh"),
38   ],
39   name="generator",
40 )
41 generator.summary()

```

along with the following hyper parameters

Parameter	Value
Latent Dimension	256
Generator Optimizer	RMSProp
Discriminator Optimizer	RMSProp
Generator Learning Rate	$1 \times 10^{-4}$
Discriminator Learning Rate	$1 \times 10^{-4}$
Generator Momentum	0
Discriminator Momentum	0
Steps per Epoch	253
Batch Size	16
Number of Epochs	200

Table 3.19: DCGAN for Producing Synthetic X-ray COVID Class Data for Extensive COVID 19 Dataset

Once training finished the model attained a discriminator loss of 0.6901 and a generator

loss of 0.7557.

The next DCGAN architecture produced the Non-COVID class, this is the majority class but we wanted to compare and contrast the results from both DCGANs. The model uses the following architecture

```

1 # based off example:
2   ↪ https://keras.io/examples/generative/dcgan\_overriding\_train\_step/
3 discriminator = keras.Sequential(
4   [
5     keras.Input(shape=(128, 128, 3)),
6     layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
7     layers.LeakyReLU(alpha=0.5),
8     layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
9     layers.LeakyReLU(alpha=0.5),
10    layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
11    layers.LeakyReLU(alpha=0.5),
12    layers.Flatten(),
13    layers.Dropout(0.4),
14    layers.Dense(1, activation="sigmoid"),
15  ],
16  name="discriminator",
17 )
18 # Create the generator.
19 generator = keras.Sequential(
20   [
21     keras.Input(shape=(latent_dim,)),
22     layers.Dense(16 * 16 * 128),
23     layers.Reshape((16, 16, 128)),
24     layers.Conv2DTranspose(256, kernel_size=4, strides=2,
25       ↪ padding="same"),
26     layers.LeakyReLU(alpha=0.2),
27     layers.Conv2DTranspose(512, kernel_size=4, strides=2,
28       ↪ padding="same"),
29     layers.LeakyReLU(alpha=0.2),
30     layers.Conv2DTranspose(1024, kernel_size=4, strides=2,
31       ↪ padding="same"),
32     layers.LeakyReLU(alpha=0.2),
33     layers.Conv2D(3, kernel_size=4, padding="same", activation="tanh"),
34   ],

```

```

31     name="generator",
32 )

```

The model runs for 100 epochs and uses a learning rate of  $10^{-5}$  for the generator and discriminator instead of  $10^{-4}$ . The batch size was also increased to 64 to help the model train faster. The model achieved a final loss of 0.6912 for the discriminator and 0.7575 for the generator.

## CT DCGANs

For the DCGAN for CT Covid the batch size was chosen as 64 to increase the training speed. The following architecture was used for this DCGAN

```

1 # based off example:
→ https://keras.io/examples/generative/dcgan_overriding_train_step/
2 discriminator = keras.Sequential(
3 [
4     keras.Input(shape=(128, 128, 3)),
5     layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
6     layers.LeakyReLU(alpha=0.5),
7     layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
8     layers.LeakyReLU(alpha=0.5),
9     layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
10    layers.LeakyReLU(alpha=0.5),
11    layers.Flatten(),
12    layers.Dropout(0.4),
13    layers.Dense(1, activation="sigmoid"),
14 ],
15 name="discriminator",
16 )
17 discriminator.summary()
18
19 # Create the generator.
20 generator = keras.Sequential(
21 [
22     keras.Input(shape=(latent_dim,)),
23     layers.Dense(8 * 8 * 128),
24     layers.Reshape((8, 8, 128)),
25     layers.Conv2DTranspose(128, kernel_size=4, strides=2,
26     → padding="same"),
27     layers.LeakyReLU(alpha=0.2),

```

```

27     layers.Conv2DTranspose(256, kernel_size=4, strides=2,
28         ↪ padding="same"),
29     layers.LeakyReLU(alpha=0.2),
30     layers.Conv2DTranspose(512, kernel_size=4, strides=2,
31         ↪ padding="same"),
32     layers.LeakyReLU(alpha=0.2),
33     layers.Conv2DTranspose(1024, kernel_size=4, strides=2,
34         ↪ padding="same"),
35     layers.LeakyReLU(alpha=0.2),
36     layers.Conv2D(3, kernel_size=4, padding="same", activation="tanh"),
37 ],
38 name="generator",
39 )
40 generator.summary()

```

and used the following hyper parameters

Parameter	Value
Latent Dimension	256
Generator Optimizer	RMSProp
Discriminator Optimizer	RMSProp
Generator Learning Rate	$1 \times 10^{-4}$
Discriminator Learning Rate	$1 \times 10^{-4}$
Generator Momentum	0
Discriminator Momentum	0
Steps per Epoch	85
Batch Size	64
Number of Epochs	100

Table 3.20: DCGAN for Producing Synthetic CT COVID Class Data for Extensive COVID 19 Dataset

The model achieved a final loss score of 0.6443 for the discriminator and 0.9409 for the generator.

The next model for the non-covid class uses the following architecture

```

1 # based off example:
2   ↪ https://keras.io/examples/generative/dcgan\_overriding\_train\_step/
3 discriminator = keras.Sequential(
4   [

```

```

4     keras.Input(shape=(128, 128, 3)),
5     layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
6     layers.LeakyReLU(alpha=0.5),
7     layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
8     layers.LeakyReLU(alpha=0.5),
9     layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
10    layers.LeakyReLU(alpha=0.5),
11    layers.Flatten(),
12    layers.Dropout(0.2),
13    layers.Dense(1, activation="sigmoid"),
14  ],
15  name="discriminator",
16 )
17 discriminator.summary()
18
19 # Create the generator.
20 generator = keras.Sequential(
21 [
22     keras.Input(shape=(latent_dim,)),
23     layers.Dense(4 * 4 * 128),
24     layers.Reshape((4, 4, 128)),
25     layers.Conv2DTranspose(64, kernel_size=4, strides=2, padding="same"),
26     layers.LeakyReLU(alpha=0.2),
27     layers.Conv2DTranspose(128, kernel_size=4, strides=2,
28                           padding="same"),
29     layers.LeakyReLU(alpha=0.2),
30     layers.Conv2DTranspose(256, kernel_size=4, strides=2,
31                           padding="same"),
32     layers.LeakyReLU(alpha=0.2),
33     layers.Conv2DTranspose(512, kernel_size=4, strides=2,
34                           padding="same"),
35     layers.LeakyReLU(alpha=0.2),
36     layers.Conv2DTranspose(1024, kernel_size=4, strides=2,
37                           padding="same"),
38     layers.LeakyReLU(alpha=0.2),
39     layers.Conv2D(3, kernel_size=4, padding="same", activation="tanh"),
40 ],
41 name="generator",

```

---

```

38    )
39 generator.summary()

```

and the following hyper parameters

Parameter	Value
Latent Dimension	256
Generator Optimizer	RMSProp
Discriminator Optimizer	RMSProp
Generator Learning Rate	$1 \times 10^{-4}$
Discriminator Learning Rate	$1 \times 10^{-4}$
Generator Momentum	0
Discriminator Momentum	0
Steps per Epoch	42
Batch Size	64
Number of Epochs	100

Table 3.21: DCGAN for Producing Synthetic CT Non COVID Class Data for Extensive COVID 19 Dataset

The model achieved a final loss score of 0.6522 for the discriminator and 1.0048 for the generator.

### 3.9 Conclusion

In conclusion most of the CNN models showed decent results when trained on the original datasets, although there are a few exceptions where the dataset used to train the models was small in size or imbalanced. The GANs had achieved satisfactory results and produced images which were similar to those in the dataset which can be seen in the next chapter of this thesis. Many of the models developed took a lot of tweaking, in order to get satisfactory results. A lot of hyperparameter tuning was involved and the training was a time-consuming and laborious process as each model needed to be rerun when the hyperparameters were updated.

With that said a number of models may benefit from more tweaking but for the purpose of this study the results achieved were satisfactory, given the purpose of this study is to gauge if synthetic data could improve an existing model's accuracy. It is also worth remembering the models trained in this section are trained on limited data and / or imbalanced datasets which could skew the results.

# Chapter 4

## Results

### 4.1 Evaluation of Performance of The Augmented CNN Models

In this section we will evaluate the augmented CNN models and compare and contrast with the original models. The section is broken into subsections for each datasets CNN models.

#### 4.1.1 Radiography CNN Models

The non-augmented radiography dataset was imbalanced so a total of 6,570 images were generated for the COVID X-rays and masks and a total of 8,840 pneumonia X-ray and mask images were generated to bring both classes in relative balance with the Normal class. The augmented dataset will be off by around 5-10 images for augmented classes due to computational issues when trying to generate all files at once but the results still show a big improvement in some models. The augmented dataset contains 60,933, while the original dataset contained only 30,306.

##### Radiography Baseline Model

The radiography baseline model ran for 10 epochs with 1333 steps per epoch and attained a final training set accuracy of 0.9314 and a final training set loss of 0.1663 along with a final validation set accuracy of 0.9200 and a validation set loss of 0.2200. The model achieved a test set accuracy of 0.9222 and a test set loss of 0.2068 when the dataset was augmented. In comparison to the original model which has a training set accuracy of 0.8611 and a training loss of 0.3446 and a validation set accuracy of 0.8306 and a validation set loss of 0.4120 and a test set accuracy of 0.8434 and a test set loss of 0.3801 when trained for the same amount of epochs on the non-augmented set. The model trained on the augmented set shows a clear improvement in terms of accuracy and of loss. The training accuracy increased by 0.0703 and

the training loss decreased by 0.1683 along with the validation accuracy increasing by 0.0894 and the loss decreased by 0.192. The test set accuracy also increased by 0.0788 and the test set loss was lowered by 0.1732 when training on the augmented dataset. In this case the augmented model shows a clear improvement over the original baseline model. The training and validation accuracy along with the training and validation loss of this model are shown below.

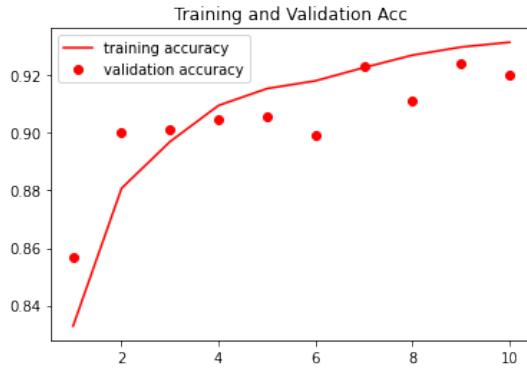


Figure 4.1: Radiography Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

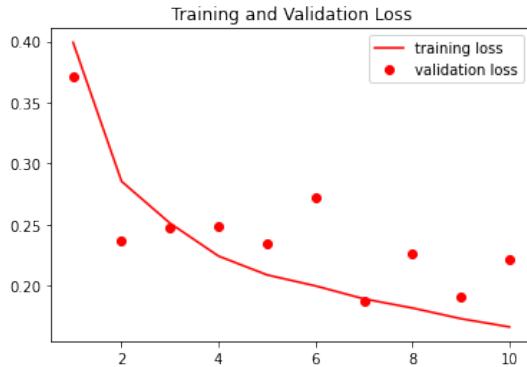


Figure 4.2: Radiography Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Radiography Xception Model

The augmented Radiography Xception model attained a final training set accuracy of 0.9699 and a final training set loss of 0.0776 alongside a final validation accuracy of 0.9435 and a final validation loss of 0.1365 after 10 epochs. The model also achieved a final test set accuracy of 0.9464 and a test set loss of 0.1369. In comparison the original model attained a training set accuracy of 0.9543 and a training set loss of 0.1162 alongside a validation set accuracy of 0.8964 and a validation set loss of 0.3867. The original model also had a test set accuracy of

0.9002 and a test set loss of 0.3833. The augmented model shows an increase of 0.0156 for the training accuracy and a decrease of 0.0386 for the training loss. The validation set accuracy had an increase of 0.0471 and the validation set loss had an decrease of 0.2502. The model also performed better on the test set with accuracy increasing by 0.0462 and loss decreasing by 0.2464. Again this model outperformed the original non-augmented model. Figure 4.3 shows the training and validation accuracy and figure 4.4 shows the training and validation loss.

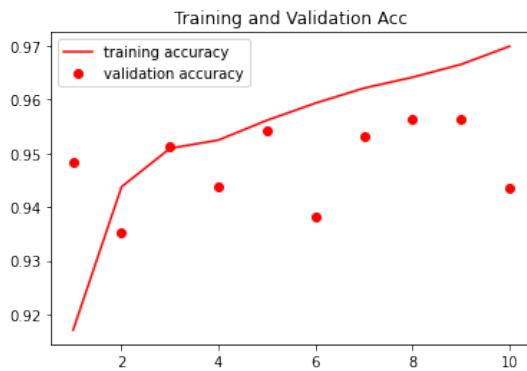


Figure 4.3: Radiography Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

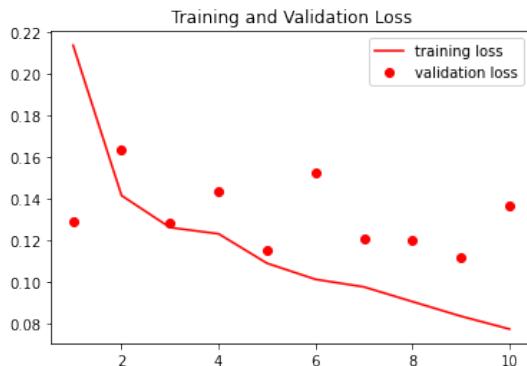


Figure 4.4: Radiography Augmented Xception Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Radiography ResNet50V2 Model

The augmented radiography ResNet50V2 Model attained a final training set accuracy of 0.9593 and a training set loss of 0.1034 alongside a validation set accuracy of 0.8638 and a validation set loss of 0.3838. The augmented model also has a test set accuracy of 0.8623 and a test set loss of 0.3814. To contrast this with the original model, the original model attained a final training set accuracy of 0.9172 and a training set loss of 0.2024 along with a validation

set accuracy of 0.8695 and a validation set loss of 0.3944. The original model also had a test set accuracy of 0.8809 and a test set loss of 0.3942. The augmented model performs better on the training set with an accuracy increase of 0.0421 and a loss decrease of 0.099 but performed worse on the validation set in terms of accuracy with an accuracy decrease of 0.0057 and a loss decrease of 0.0106. The model also performed slightly worse on the test set in terms of accuracy with a decrease of 0.0186 although it did have a decrease in loss which decreased by 0.0129. Figure 4.5 shows the training and validation accuracy and figure 4.6 shows the training and validation loss.

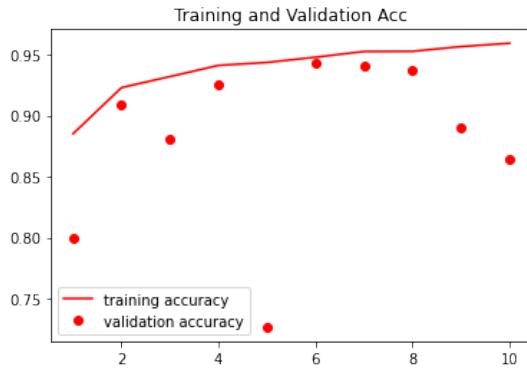


Figure 4.5: Radiography Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

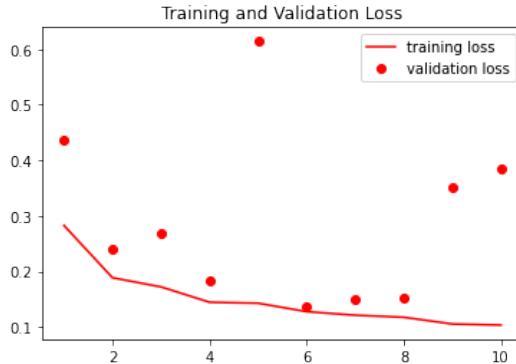


Figure 4.6: Radiography Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Radiography EfficientNetV2S Model

The augmented EfficientNetV2S Model completed training with a final training set accuracy of 0.9640, with a training set loss of 0.0910 alongside a validation set accuracy of 0.9553 and a validation loss of 0.1167. The model also had a test set accuracy of 0.9604 and a test set loss of 0.1056. In contrast the original model attained a final training set accuracy 0.9383 of

and a training set loss of 0.1519 alongside a validation accuracy of 0.8844 and a validation loss of 0.3362. The original model also achieved a test set accuracy of 0.8811 and a test set loss of 0.3217. This shows that the augmented model's training set accuracy increased by 0.0257 and its training set loss decreased by 0.0609 and the validation set accuracy increased by 0.0709 and its validation loss decreased by 0.2195. The test set also improved in terms of loss and accuracy with an accuracy increase of 0.0794 and a loss decrease of 0.2161. The augmented model has shown clear improvements when compared with the base model however, as shown from the images below early stopping could have helped improve the accuracy for the validation set. Figure 4.7 shows the training and validation accuracy and figure 4.8 shows the training and validation loss.

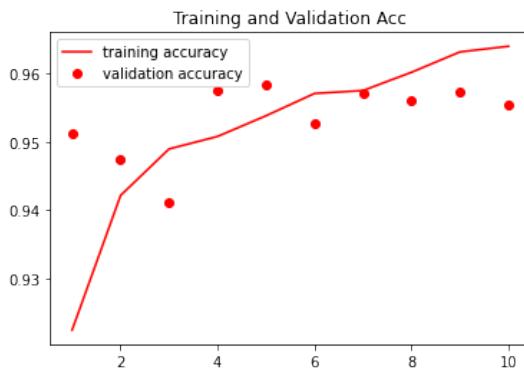


Figure 4.7: Radiography Augmented EfficientNetV2S Model DCGAN Accuracy The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 4.8: Radiography Augmented EfficientNetV2S Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Evaluation of Radiography Models

Most of the radiography models showed a clear improvement across all sets when training on the augmented datasets. The models which performed worse only suffered from a slight

decrease in performance in terms of accuracy but had a decrease in loss which means the model may not have predicted all the cases correctly but was more confident in doing so. The increase in confidence of all the radiography models could avoid situations where patients are unnecessarily quarantined. The results from this dataset are significant in that they show a clear improvement across all models.

#### 4.1.2 Extensive CNN Models

The Extensive CT dataset was augmented by 2,700 images and the augmented dataset has a total of 10,754 images, the original Extensive CT dataset had 8,054 images. The original Extensive X-ray dataset had a total of 9,537 images which was increased to 10,537 when augmented, in total an additional 1,180 images were added to the augmented dataset.

##### **Extensive CNN CT Baseline Model**

The Extensive CNN CT baseline model attained a final training set accuracy of 0.9199 and a final training set loss of 0.1839 along with a final validation set accuracy of 0.5333 and a validation set loss of 3.4314 when the dataset was augmented. The model also had a test set loss of 3.2788 and a test set accuracy of 0.5373. In comparison to the original model which had a training set accuracy of 0.9023 and a training set loss of 0.2286 and a validation set accuracy of 0.8150 and a validation set loss of 0.3837 when trained for the same amount of epochs on a non-augmented set. The original model also had a test set accuracy of 0.8319 and a test set loss of 0.3574. Overall the augmented model performed better on the training set but worse on both the validation and test sets. The training set accuracy increased by 0.0176 and the training set loss decreased by 0.0447 along with the validation set accuracy decreasing by 0.2817 and the loss increasing by 3.0477. The test set accuracy also declined by 0.2946 and the loss increased by 2.9206. The training and validation set accuracy along with the training and validation set loss of this model are shown below. The model appears to have a significant decrease in accuracy on the 10th epoch early stopping may have improved the overall accuracy. Figure 4.9 shows the training and validation accuracy and figure 4.10 shows the training and validation loss.

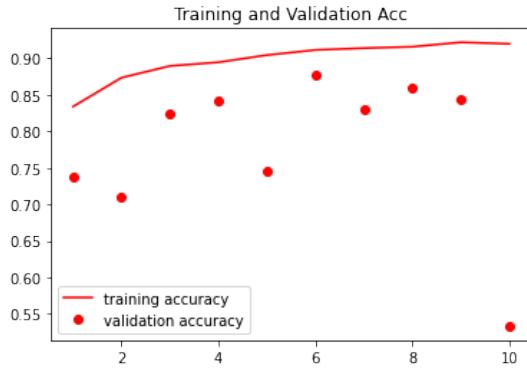


Figure 4.9: Extensive CT Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

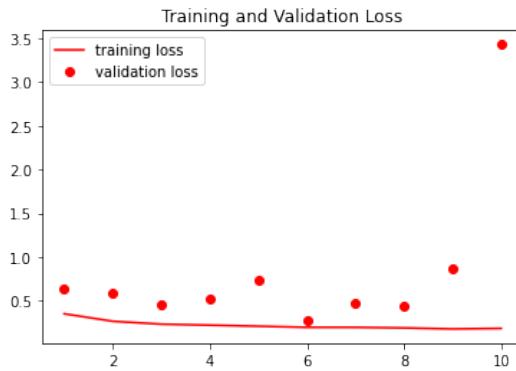


Figure 4.10: Extensive CT Augmented Baseline Model DCGAN - The X-Axis shows the epoch number and the Y-axis shows the loss

### Extensive CNN CT Xception Model

The Extensive CNN CT Xception model attained a final training set accuracy of 0.9920 and a final training set loss of 0.0278 along with a final validation set accuracy of 0.9704 and a validation set loss of 0.0777 when the dataset was augmented. The model also has a test set accuracy of 0.9688 and a test set loss of 0.1054. In comparison to the original model which had a training set accuracy of 0.9904 and a training set loss of 0.0238 and a validation set accuracy of 0.9534 and a validation set loss of 0.2250 when trained for the same amount of epochs on a non-augmented set. The original model also had a test set accuracy of 0.9538 and a test set loss of 0.1772. The augmented model performed better on all sets. The training set accuracy increased by 0.0016 and the training set loss decreased by 0.1494. The validation set accuracy increased by 0.017 and the loss decreased by 0.1472. The test set accuracy also increased by 0.015 and the test set loss decreased by 0.0717. Early stopping would have improved accuracy on the validation set as shown in the figures below which may in turn have improved the test

set accuracy. Figure 4.11 shows the training and validation accuracy and figure 4.12 shows the training and validation loss.

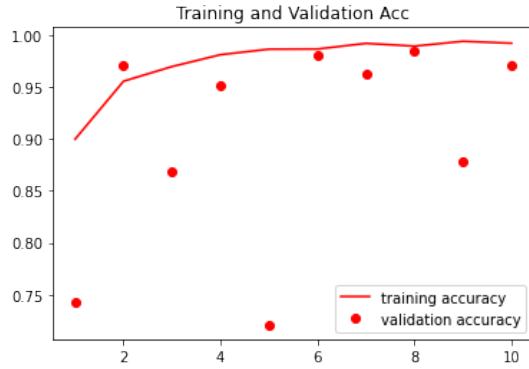


Figure 4.11: Extensive CT Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

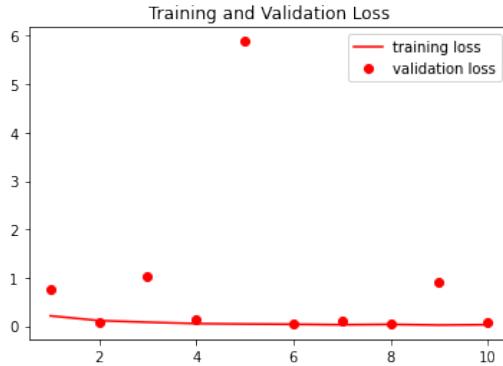


Figure 4.12: Extensive CT Augmented Xception Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Extensive CNN CT ResNet50V2 Model

The Extensive CNN CT ResNet50V2 model attained a final training set accuracy of 0.9485 and a final training set loss of 0.1255 along with a final validation set accuracy of 0.6091 and a validation set loss of 1.1563 when the dataset was augmented. The model also has a test set accuracy of 0.6040 and a test set loss of 1.1806. In comparison to the original model which had a training set accuracy of 0.9604 and a training set loss of 0.1140 and a validation set accuracy of 0.9350 and a validation set loss of 0.2078 when trained for the same amount of epochs on a non-augmented set. The original model also had a test set accuracy of 0.9262 and a test set loss of 0.1993. The augmented model performed worse on all sets. The training set accuracy decreased by 0.0119 and the training set loss increased by 0.0115. The validation set accuracy decreased by 0.3259 and the validation set loss increased by 0.9485. The test

accuracy also decreased by 0.3222 and the test loss increased by 0.9807. Early stopping may have improved accuracy on the training and validation sets as shown in the figures below which may in turn have improved the test set accuracy. Figure 4.13 shows the training and validation accuracy and figure 4.14 shows the training and validation loss.

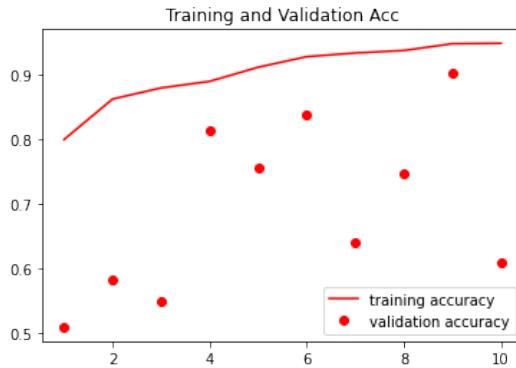


Figure 4.13: Extensive CT Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

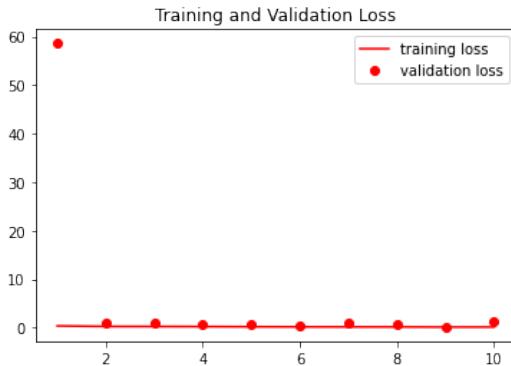


Figure 4.14: Extensive CT Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Extensive CNN CT EfficientNetV2S Model

The Extensive CNN CT EfficientNetV2S model attained a final training set accuracy of 0.9957 and a final training set loss of 0.0150 along with a final validation accuracy of 0.9815 and a validation set loss of 0.0634 when the dataset was augmented. The augmented model also has a test accuracy of 0.9790 and a test loss of 0.0874. In comparison to the original model which had a training set accuracy of 0.9901 and a training set loss of 0.0291 and a validation set accuracy of 0.9375 and a validation set loss of 0.2638 when trained for the same amount of epochs on a non-augmented set. The original model also has a test set accuracy of 0.9437 and a test set loss of 0.2353. The model performed better on all sets when training on the

augmented set. The training accuracy increased by 0.0056 and the training loss decreased by 0.0141. The validation accuracy increased by 0.044 and the loss decreased by 0.2004. The test set accuracy increased by 0.0352 and the test set loss decreased by 0.1479. This is an improvement over the original model on all sets which is a significant result. Figure 4.15 shows the training and validation accuracy and figure 4.16 shows the training and validation loss.

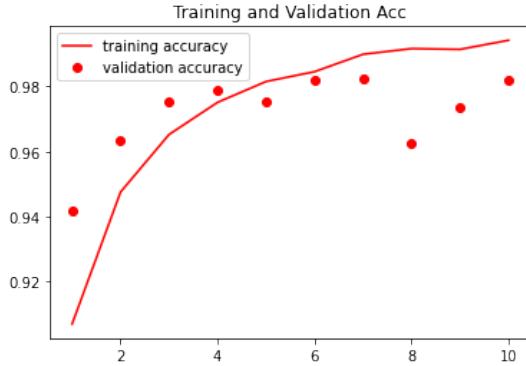


Figure 4.15: Extensive CT Augmented EfficientNetV2S Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 4.16: Extensive CT Augmented EfficientNetV2S Model DCGAN - The X-Axis shows the epoch number and the Y-axis shows the loss

### Evaluation of Extensive CNN CT Models

The extensive CT models had mixed results when training on the augmented set. The baseline model performed worse as did the ResNet50V2 model although the Xception and EfficientNetV2S models showed improvements across all sets. Early stopping may have improved the accuracy for the Baseline and ResNet50V2 models as shown in the figures showing the validation and training accuracy and losses, it appears that the models may have been overtrained when training for the same number of epochs as the original given there is more data in the augmented sets. The baseline model shows clear signs of overtraining as there is a large loss

in accuracy and a large increase in loss on the 10th epoch and the ResNet50V2 model shows a large drop in accuracy on the 10th epoch as well.

### Extensive CNN X-ray Baseline Model

The Extensive CNN X-ray baseline model attained a final training set accuracy of 0.9257 and a final training set loss of 0.1911 along with a final validation set accuracy of 0.7741 and a validation set loss of 1.1363 when the dataset was augmented. The augmented model also had a test set accuracy of 0.7784 and a test set loss of 1.0640. In comparison to the original model which had a training set accuracy of 0.9278 and a training set loss of 0.1948 in addition to a validation set accuracy of 0.8587 and a validation set loss of 0.3482 when trained for the same amount of epochs on a non-augmented set. The original model also had a test set accuracy of 0.8698 and a test set loss of 0.3203. The model performed worse on all sets when trained on the augmented set. The training set accuracy decreased by 0.0021 and the training set loss decreased by 0.0037. The validation set accuracy decrease by 0.0846 and the validation set loss increased by 0.7881. The test set accuracy decreased by 0.0914 and the test set loss increased by 0.7437.

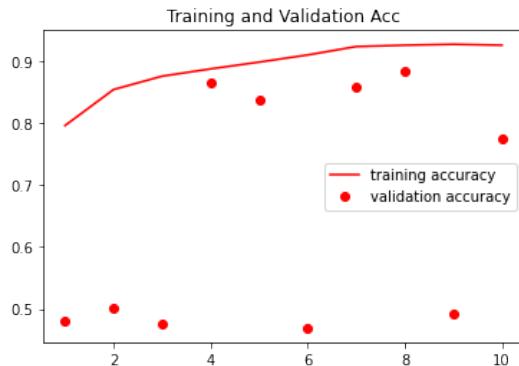


Figure 4.17: Extensive X-ray Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

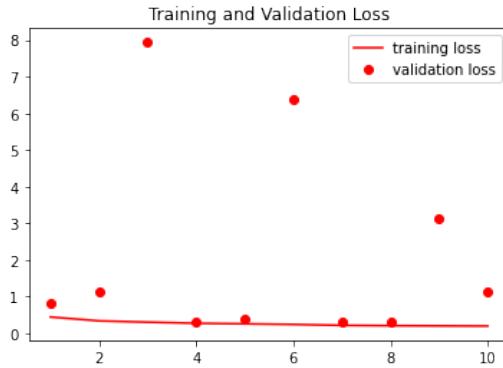


Figure 4.18: Extensive X-ray Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Extensive CNN X-ray Xception Model

The Extensive CNN X-ray Xception model attained a final training set accuracy of 0.9864 and a final training set loss of 0.0304 along with a final validation set accuracy of 0.9581 and a validation set loss of 0.2502 when the dataset was augmented. The augmented model also has a test set accuracy of 0.9659 and a test set loss of 0.2254. In comparison to the original model which had a training set accuracy of 0.9876 and a training set loss of 0.0277 in addition to a validation set accuracy of 0.9416 and a validation set loss of 0.2814 when trained for the same amount of epochs on a non-augmented set. The original model also has a test set accuracy of 0.9578 and a test set loss of 0.1948. The model performed slightly worse on the training set but better on both the validation set and on the test set when training on the augmented set. The training set accuracy decreased by 0.0012 and the training set loss increased by 0.0027. The validation set accuracy increased by 0.0165 and the validation set loss decreased by 0.0312. The test set accuracy increased by 0.0081 and the test set loss increased by 0.0306. The augmented and original model are fairly close in terms of accuracy and loss with the augmented model performing worse on the training set by a small margin but the model performed better on the validation and test sets in terms of accuracy by a small margin although the loss for the test set is slightly higher in the augmented model. Figure 4.19 shows the training and validation accuracy and figure 4.20 shows the training and validation loss.

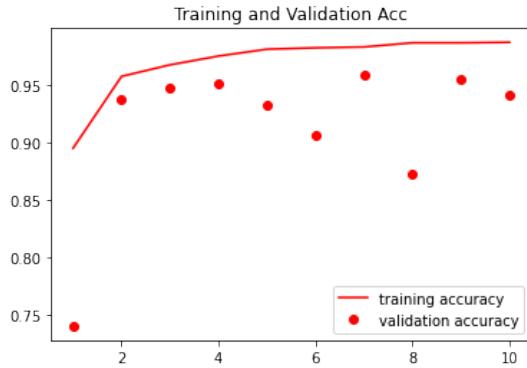


Figure 4.19: Extensive X-ray Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 4.20: Extensive X-ray Augmented Xception Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Extensive CNN X-ray ResNet50V2 Model

The Extensive CNN X-ray ResNet50V2 model attained a final training set accuracy of 0.9687 and a final training set loss of 0.0852 along with a final validation set accuracy of 0.8275 and a validation set loss of 0.4574 when the dataset was augmented. The augmented model also achieved an accuracy of 0.8210 for the test set along with a test set loss of 0.4680. In comparison to the original model which has a training set accuracy of 0.9649 and a training set loss of 0.0907 in addition to a validation set accuracy of 0.9139 and a validation set loss of 0.9139 when trained for the same amount of epochs on a non-augmented set. The original model also had a test set accuracy of 0.9224 and a test set loss of 0.2379. The model performed worse on all sets when training on the augmented set. The training set accuracy decreased by 0.0038 and the training set loss decreased by 0.0055. The validation set accuracy decreased by 0.0864 and the validation set loss increased by 0.1732. Test set accuracy decreased by 0.1014 and the test set loss increased by 0.2301.

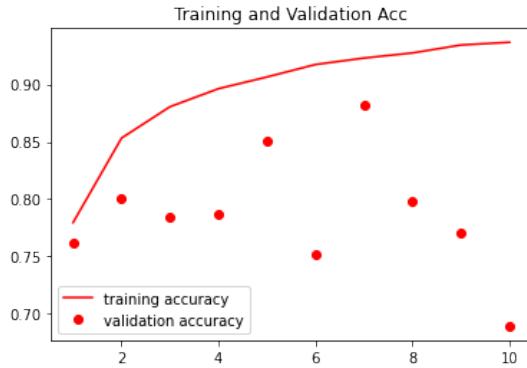


Figure 4.21: Extensive X-ray Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 4.22: Extensive X-ray Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Extensive CNN X-ray EfficientNetV2S Model

The Extensive CNN X-ray EfficientNetV2S model attained a final training set accuracy of 0.9864 and a final training set loss of 0.0366 along with a final validation set accuracy of 0.9695 and a validation set loss of 0.1686 when the dataset was augmented. The model also has a test set accuracy of 0.9697 and a test set loss of 0.1579. In comparison to the original model which had a training set accuracy of 0.9840 and a training set loss of 0.0396 in addition to a validation set accuracy of 0.9702 and a validation set loss of 0.1603 when trained for the same amount of epochs on a non-augmented set. The original model also has a test set accuracy of 0.9766 and a test set loss of 0.1513. The model performed slightly worse on all sets except the training set when training on the augmented set. The training set accuracy increased by 0.0024 and the training set loss decreased by 0.003. The validation set accuracy decreased by 0.0007 and the validation set loss increased by 0.0083. The test set accuracy decreased by 0.0069 and the test set loss increased by 0.0066. It appears from analyzing these models that

the quality of the X-ray images don't match the quality of the CT images as all the X-ray models performed worse than the originals. Figure 4.23 shows the training and validation accuracy and figure 4.24 shows the training and validation loss.

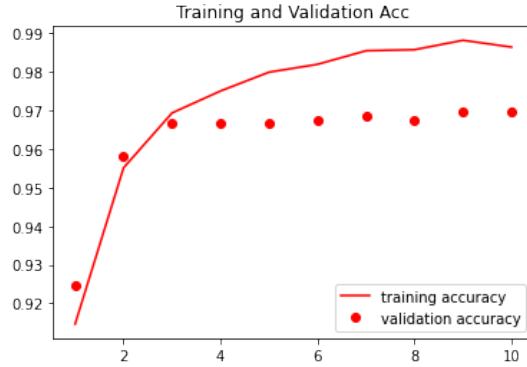


Figure 4.23: Extensive X-ray Augmented EfficientNetV2S Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

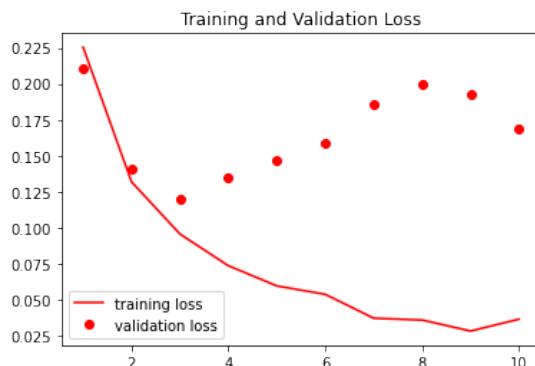


Figure 4.24: Extensive X-ray Augmented EfficientNetV2S Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Evaluation of Extensive CNN X-ray Models

Most of the Extensive X-ray CNN models performed worse in terms of accuracy and loss bar the Xception model which had slight improvements to both accuracy and loss. The reason for the poor performance may be due to the variety in the X-ray images for this dataset as some X-rays are taken from the side and others are taken when the patient is facing forward. The poor performance could also have been caused by the quality of the augmented data. When analyzing the output of the GAN for the X-ray model it appears that a lot of the images are grainy and lacking in quality. The CT models on the other hand showed a clear improvement as the data generated from that set may suffer from grain and artifacts but the synthetic images share a lot of features with the original images.

#### 4.1.3 X-ray COVID-19 dataset CNN Models

This dataset was augmented with 2000 artificially generated images, 1000 belonging to the pneumonia class and the other 1000 belonging to the normal class. The augmented images were used only in training the models. Given the limited size of the dataset if we were to include augmented images in the validation set it would greatly skew the results as we would be gauging the model's ability to predict synthetic images as opposed to real images. The limited size of the validation set for these models means that their performance is open to scrutiny. The transfer learning models for this model include an additional layer of 1024 units with a ReLU activation function and another layer of 1 unit with a Sigmoid activation function. The total size of the augmented set amounts to 2,148 images, the original dataset contained only 188 images.

#### X-ray COVID-19 Baseline Model

The X-ray COVID-19 Baseline model attained a final training set accuracy of 0.9926 and a final training set loss of 0.0207 along with a final validation set accuracy of 0.5000 and a validation set loss of 5.2546 when the dataset was augmented. The model also has a test set accuracy of 0.4375 and a test set loss of 6.2093. In comparison to the original model which had a training set accuracy of 0.9122 and a training set loss of 0.2651 in addition to a validation set accuracy of 0.5000 and a validation set loss of 0.7918 when trained for the same amount of epochs on a non-augmented set. The model performed worse on the validation and test sets in terms of accuracy when training on the augmented set but the augmented model had a higher accuracy and lower loss for the training set. The training set accuracy increased by 0.0804 and the training set loss decreased by 0.2444. The validation set accuracy stayed the same at 0.5000 and the validation set loss increased by 4.4628. The test set accuracy decreased by 0.0313 and the test set loss increased by 5.392. Early stopping may have helped this model as the loss is a lot lower at epoch 8 and 9 for the validation set and has a higher validation accuracy with similar training accuracy. Figure 4.25 shows the training and validation accuracy and figure 4.26 shows the training and validation loss.

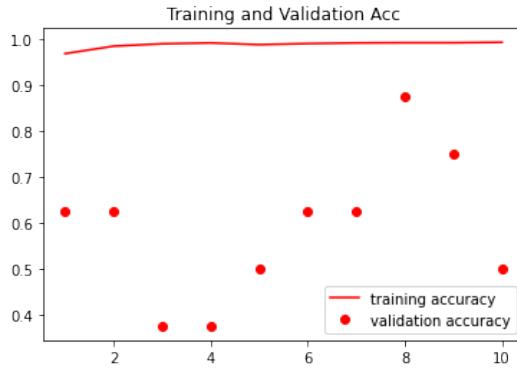


Figure 4.25: X-ray COVID-19 Augmented Baseline Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy



Figure 4.26: X-ray COVID-19 Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### X-ray COVID-19 Xception Model

The X-ray COVID-19 Xception model attained a final training set accuracy of 1.0000 and a final training set loss of 0.000047 along with a final validation set accuracy of 0.8750 and a validation set loss of 0.1786 when the dataset was augmented. The model also achieved a test set accuracy of 0.8438 and a test set loss of 0.5912. In comparison to the original model which had a training set accuracy of 0.9662 and a training set loss of 0.0951 in addition to a validation set accuracy of 0.5000 and a validation set loss of 5.4697 when trained for the same amount of epochs on a non-augmented set. The original model also has a test set accuracy of 0.3750 and a test set loss of 7.5530 . The model performed better on all sets in terms of accuracy when training on the augmented set. The training set accuracy increased by 0.0338 and the training set loss decreased by 0.0950. The validation accuracy increased by 0.375 and the loss decreased by 5.2911. The test set accuracy increased by 0.4688 and test set loss decreased by 6.9618. Early stopping may help this model increase it's test set accuracy as the

training loss is much lower at epochs 6 and 7 as is shown in the figures below. Figure 4.27 shows the training and validation accuracy and figure 4.28 shows the training and validation loss.

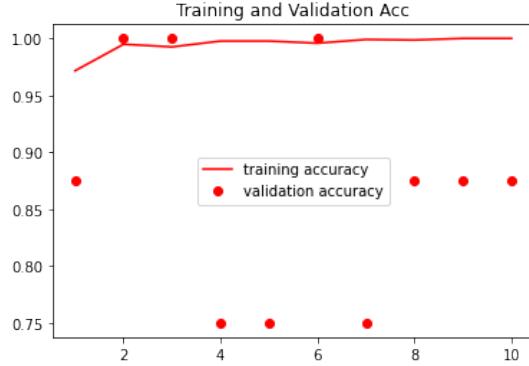


Figure 4.27: X-ray COVID-19 Augmented Xception Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

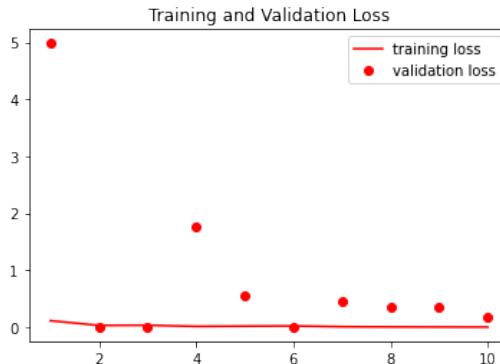


Figure 4.28: X-ray COVID-19 Augmented Baseline Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### X-ray COVID-19 ResNet50V2 Model

The X-ray COVID-19 ResNet50V2 model attained a final training set accuracy of 0.9846 and a final training set loss of 0.0565 along with a final validation set accuracy of 0.6250 and a validation loss of 1.4650 when the dataset was augmented. The model also attained a test set accuracy of 0.6250 along with a test set loss of 2.0608. In comparison to the original model which had a training set accuracy of 0.9257 and a training loss of 0.2622 in addition to a validation set accuracy of 0.3750 and a validation set loss of 29.2193 when trained for the same amount of epochs on a non-augmented set. The original model also had a test set accuracy of 0.4688 and test set loss of 23.9440. The model performed better on all sets in terms of both accuracy and loss. The training set accuracy increased by 0.0589 and the training set

loss decreased by 0.2057. The validation set accuracy increased by 0.25 and the validation set loss decreased by 27.7543. The test set accuracy decreased by 0.15625 and the test set loss decreased by 21.88. The model shows a significant improvement when compared to the non augmented model. As seen with previous models early stopping would have benefited this model as the training and validation accuracy is higher in earlier epochs. Figure 4.29 shows the training and validation accuracy and figure 4.30 shows the training and validation loss.

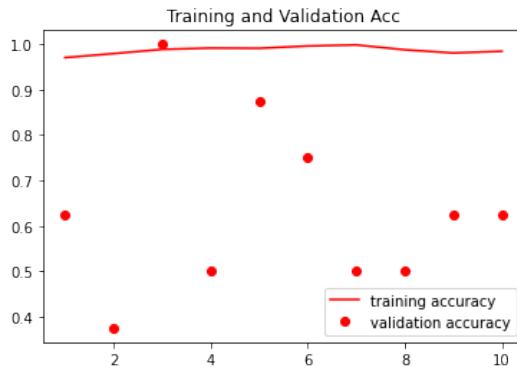


Figure 4.29: X-ray COVID-19 Augmented ResNet50V2 Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

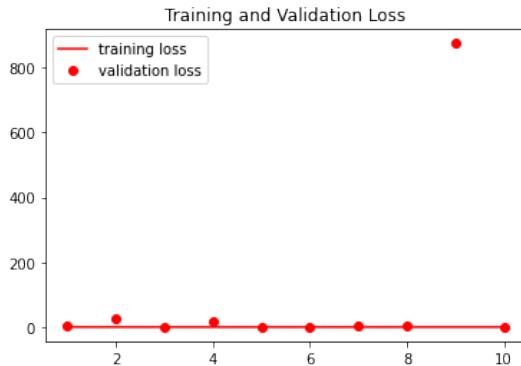


Figure 4.30: X-ray COVID-19 Augmented ResNet50V2 Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### X-ray COVID-19 EfficientNetV2S Model

The X-ray COVID-19 EfficientNetV2S model attained a final training set accuracy of 0.9981 and a final training set loss of 0.0091 along with a final validation set accuracy of 1.0000 and a validation set loss of 0.0106 when the dataset was augmented. The model attained a test set accuracy of 0.9375 along with a test set loss of 0.1713. In comparison to the original model which had a training set accuracy of 1.0000 and a training set loss of 0.0063 in addition to a validation set accuracy of 1.0000 and a validation set loss of 0.0011 when trained for the same

amount of epochs on a non-augmented set. The non augmented model also had a test set accuracy of 0.9375 and a test set loss of 0.2317. The model performed better on the test set but slightly worse on the training set and validation set. The training set accuracy decreased by 0.0019 and the training set loss increased by 0.0028. The validation set accuracy stayed the same and the validation set loss increased by 0.0095. The test set accuracy stayed the same and the test set loss decreased by 0.0604. Early stopping would have helped increase the training and validation scores as shown in the figures below. Figure 4.31 shows the training and validation accuracy and figure 4.32 shows the training and validation loss.

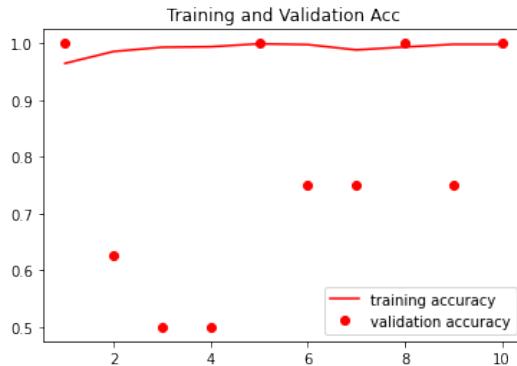


Figure 4.31: X-ray COVID-19 Augmented EfficientNetV2S Model DCGAN Accuracy - The X-Axis shows the epoch number and the Y-axis shows the accuracy

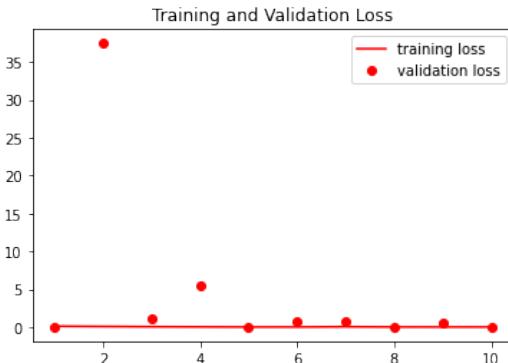


Figure 4.32: X-ray COVID-19 Augmented EfficientNetV2S Model DCGAN Loss - The X-Axis shows the epoch number and the Y-axis shows the loss

### Evaluation of CNN models for X-ray COVID-19 Dataset

Although some of the models do show improvement, the ability to evaluate the models is limited. The original dataset is broken into images for train and test and the test set only contains 40 images. It would be unwise to augment the test set as then we would have been comparing synthetic images with synthetic images, and given that the amount of images

synthetically generated is far larger than the real data in the dataset, it would have greatly skewed the results. Some of the models do show an improvement when compared with the original and those that perform worse than the original do so by a small margin. The more data in the dataset the more features there are to detect so a small increase in loss and accuracy for these models is to be expected, especially given the small size of the dataset. It was surprising to see that a few models here did outperform the original as the dataset used to train the GANs was limited to 188 images, although this could be due to the test data in the dataset also being used to train the GANs. The model's performance may possibly have been improved by tweaking some of the hyper-parameters or adjusting the size of the added layer of 1024 units as the number of units added may have been too large.

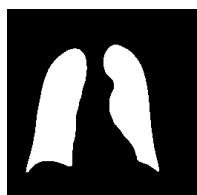
## 4.2 Evaluation of GAN Models

In this section we will analyze the output for each GAN on the augmented classes of each dataset. Some of the classes mentioned were not used to generate synthetic images for the models, as they were the majority class, but we thought they were worth discussing regardless.

### 4.2.1 Radiography GAN Models

#### Radiography DCGAN for COVID-19 Class Augmentation

The radiography DCGAN for synthetically generated COVID-19 samples had mixed results. Some of the images generated by the DCGAN came out looking very similar to the masks of patients lungs which were in the database. We have included a side by side comparison in figure 4.33a and 4.33b below



(a) Real COVID-19 Radiography Mask Example



(b) Generated COVID-19 Radiography Mask Example DCGAN

As shown in the above figures 4.33a and 4.33b, the synthetically generated COVID-19 mask looks very similar to the example taken from the dataset. However not every single generated image came out as well as those that we have shown for demonstration purposes. From reviewing the generated images it appears that a number of images have some issues. A common issue faced was images which were generated with artefacts and some images which were not up to standard with the images in the dataset.



(a) Synthetically generated COVID 19 mask with Artefacts(DCGAN)



(b) Malformed Image of synthetically generated COVID 19 Mask(DCGAN)

From training a number of models there appears to be a need for pruning out bad images generated by the GAN and determining which images resemble X-Rays and Masks and which are "garbage" images which don't resemble data in our dataset. This will require a lot of manual effort in determining which generated images are worth including in the augmented dataset and which are worth throwing away.

Similarly the augmentation of the X-ray images produced good results although, much like the synthetically generated masks, there were a number with artefacts and some images that were malformed. We have included two images below to compare the synthetically generated example 4.35b with a real example 4.35a below



(a) Real COVID-19 Radiography X-ray Example



(b) Generated COVID-19 Radiography X-ray Example DCGAN

As shown in the above figures 4.35a and 4.35b the two images look very similar to each other and there are clear similarities contained in the images. Malformed images are also present in the generated data and two such examples have been shown in the figures below.



(a) Malformed COVID-19 Radiography X-ray Example Number 1 Radiography DCGAN



(b) Malformed COVID-19 Radiography X-ray Example Number 2 Radiography DCGAN

### Radiography DCGAN for Pneumonia Class Augmentation

Much like in the previous section there was some success when augmenting the Pneumonia class despite it being a little less than half the size of the covid class (for reference the covid

class contains 3,616 images in both the mask and X-ray folders where as the Pneumonia class contains 1,345 images in both the mask and X-ray folders). Most of the masks generated resembled those in the dataset an example of a synthetically generated pneumonia mask 4.37a alongside a real example mask 4.37b is visible in the figures below



(a) Generated Pneumonia Radiography Mask Example DCGAN

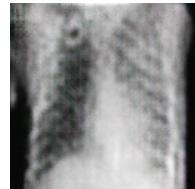


(b) Real Pneumonia Radiography Mask Example

The X-ray DCGAN also produced convincing results which are shown below in figures 4.38a which shows a real X-ray example from the dataset and 4.38b.



(a) Real Pneumonia Radiography X-ray Example



(b) Generated Pneumonia Radiography X-ray Example DCGAN

As shown in the above images 4.38a and 4.38b there appears to be a number of similarities between the two images but the synthetically generated X-ray does appear to lack the quality of the original.

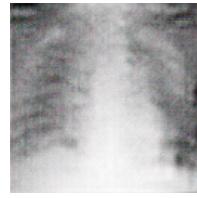
In conclusion the malformed images may have a detrimental effect on the training and it would require a lot of manual effort and computational power to both prune the augmented dataset and retrain the GANs. The examples above show that the DCGAN is a powerful method of generated synthetic data although it is not without its faults.

#### 4.2.2 Extensive X-Ray GAN Models

The X-ray DCGAN models achieved some success when synthetically generating both the X-rays for COVID and X-rays of non-COVID patients. The following images show a real example and a synthetic example side by side for comparison.



(a) Real COVID X-ray Example Extensive



(b) Synthetically Generated COVID X-ray Example Extensive DCGAN

As shown in the images above there are some similarities between the two X-rays although the synthetically generated example looks blurry and appears to be of low quality when compared with the real example. Unlike the Radiography dataset where images appeared to share characteristics there seems to be a lot more variance in this dataset as some X-rays are taken from the side where as others are taken straight forward. This is the reason that the synthetically generated images don't appear to match the quality of the synthetically generated radiography dataset images.

When using the model to produce non-COVID X-rays there seemed to be slightly better results as is shown in the figures below 4.40a which is the real example from our dataset and in 4.40b which is the synthetic example generated by the DCGAN.



(a) Real Non COVID X-ray Example Extensive



(b) Synthetically Generated Non COVID X-ray Example Extensive DCGAN

The reason for the synthetically generated non-COVID examples having more quality is that there seems to be less variance in X-ray images in the non-COVID class and also the non-COVID class is the majority in this dataset.

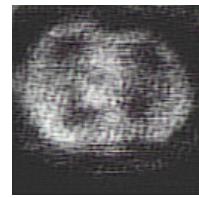
There were also the issues of malformed images and artefacts when training these DCGANs like the previous examples shown in figures 4.34b and 4.34a

#### 4.2.3 Extensive CT GAN Models

The CT GAN models had some issues reproducing the CT images this was due to their being a lot of variety in the CT dataset. Included in the figures below is an example of a real COVID CT scan along with the synthetically generated COVID CT scan.



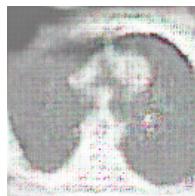
(a) Real COVID CT Example Extensive



(b) Synthetically Generated COVID CT Example Extensive DCGAN

As shown in the figures above the synthetic example 4.41b appears to be grainy and lacking in quality despite sharing some similar features with the real example 4.41a. The poor quality may be caused by the variety of images in the dataset, as there are many differing features between the images in the dataset.

As with the COVID CT scans, the non-COVID CT scans had similar results which can be seen below



(a) Synthetically Generated non COVID CT Example Extensive DCGAN



(b) Real non COVID CT Example Extensive

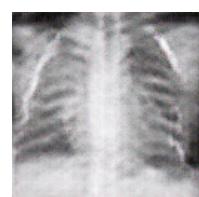
The above figures show the lack of quality in the synthetically generated image 4.42a when compared with the example taken from the dataset 4.42b. From trying a number of GAN architectures the same issue was seen with lack of quality augmented images being produced. However the synthetic images do share some features with the real example.

#### 4.2.4 X-ray COVID-19 dataset GAN Models

Surprisingly given the very small dataset the X-ray COVID-19 DCGANs produced relatively convincing images for both classes. Below are two X-ray examples for the normal class one image is generated by the DCGAN and the other is taken from the dataset



(a) Real Normal X-ray X-ray Dataset COVID-19

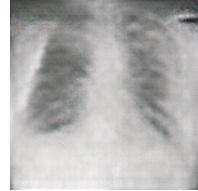


(b) Synthetically Generated Normal X-ray X-ray Dataset COVID-19 DCGAN

As shown in the figures above the synthetically generated example 4.43b lacks the quality of the original 4.43a but appears to have similar features. The next figures show an example of a synthetically generated pneumonia X-ray along with a sample taken from the dataset



(a) Real Pneumonia X-ray X-ray Dataset  
COVID-19



(b) Synthetically Generated Pneumonia  
X-ray X-ray Dataset COVID-19 DCGAN

The figure shown above 4.44b suffers from the same issue the synthetically generated images of the Normal class suffered from, in that the synthetic image appears grainy but does share some similar features with the original images 4.44a. Given the limited size of the dataset used to train the GANs there was no need to experiment further as this seems to be the best quality a DCGAN can produce.

#### 4.2.5 Summary of Results

In this section I will give a brief summary of the results from the research and their significance. I will detail what has been achieved upon completion of the implementation of the AI models.

#### Analysis of Results and Their Significance

Overall a number of augmented models showed improvements in terms of both accuracy and loss. This means that the augmented models performed better than the original models. The increase in performance of the augmented models may provide advantages in this area and allow for doctors to more easily diagnose patients more accurately, which would in turn allow doctors to have more free-time to spend on other patients. The results from this study could possibly be used in a number of fields and different areas other than the automated diagnosis of COVID-19 but more research will need to be done to gauge synthetically generated data in said areas and fields.

In comparison to the existing COVID automated diagnosis model viewed in the literature review which had accuracy of 99.4% none of the augmented or original models were able to surpass this performance. The model which came closest was the augmented Extensive CNN CT EfficientNetV2S model which had a test set accuracy of 0.9697 or approximately 97%. When evaluating the performance it is important to remember that the size of the dataset plays a significant role, the literature review model was trained on a dataset with of 4,575 X-ray images, 1,525 of which were X-rays of COVID positive patients. In comparison the Extensive COVID-19 CT dataset contains 2,628 Non-COVID CT scans and 5,427 COVID

CT scans, the augmented dataset contained an additional 2,700 Non-COVID images which in total gives the dataset a size of 10,755 images which is an increase of 4,655 images. Given that the model is exposed to more data it could be argued that this model would be more beneficial in a clinical setting given it is exposed to more features in the data which could be an improvement in terms of the model's ability to generalize. The augmented radiography models also showed some decent performance compared to the model mentioned in the literature review with the top model getting a test set accuracy of 0.9604 or 96% when trained on a far larger dataset of 60,933 images. Some of the non-augmented models were also able to get close to the COVID-19 diagnostic model mentioned in the literature review while training on slightly larger datasets the Extensive COVID-19 X-ray model which had a test set accuracy of 0.9766 or approximately 98% is a good example of this.

The overall results of this study found that synthetic data was useful for some model architectures and improved a number of models but a number of models trained on the synthetic data performed poorly compared to models trained on real data. It appears more research is needed in this area as computational limitations and lack of data pruning may have skewed the results negatively. The variation of data also proved to be detrimental in some datasets which can be seen from the Extensive X-ray, the poor performance could be due to the variety of data present as some of the X-rays are taken from the side and others are taken from with the patient facing the camera.

### 4.3 Conclusion

In conclusion of this section we can see from the figures produced by the GANs that the models were able to reproduce convincing images which look close to their target domain. We can also see that some images such as figures 4.34a and 4.34b have aberrations and artifacts which distort the image and could confuse the model when training on the synthetic data. The only solution to this issue would be to prune each image manually which would take a lot of time and effort. Due to time constraints we were unable to prune and curate each image. The models could possibly be improved by removing malformed images or images with artifacts. There is also the issue of image resolution which may have caused some models to perform worse than they normally would have if the images were standardized. Due to computational limitations it was very difficult and time-consuming to train GANs to produce high resolution images. Trying to produce images with high resolutions used a lot of computational power and often caused Colab to crash during training. However, the augmented CNN models do show improvement on a number of datasets such as the Radiography and Extensive CT datasets. Although the accuracy may be lower and the loss may be higher for some models the added data may be beneficial when applied in a clinical setting, as the augmented dataset adds more variety of images and more data to for the model to learn from to detect subtle features

and characteristics. It is important to remember the size of the datasets when gauging these models as it would be easier to achieve a much higher accuracy and lower loss on smaller datasets than it would on larger sets. The overall results of this research do show that there is much improvement in terms of both loss and accuracy as is detailed in the previous section “Analysis of Results and Their Significance“.

# **Chapter 5**

## **Conclusions**

### **5.1 Limitations**

In this section I will outline limitations faced when conducting this research and where possible include ways in which they may be mitigated when conducting future research into this problem domain.

#### **5.1.1 Computational Resources Offered by Google Colab Pro**

Due to limitations with Google Colab Pro I wasn't able to surpass certain limits when training the Convolutional Neural Networks and Generative Adversarial Networks. This means that the number of units per layer of each model could not surpass a certain limit as the runtime would run out of memory and processing power. The model's performance may be improved in future experiments when more computational power is available.

Due to this limitation models with approximately 10 to 20 million unit parameters maxed out the resources available depending on a number of factors such as the hyper parameters of the model. The lack of computational resources also affected the GANs as I was not able to use high resolutions for the images and settled for a smaller resolution when training them on the images, as higher resolutions are more computationally expensive.

Larger models could be trained when the option was available to opt for premium GPU on Google Colab but Google has only allocated a certain amount of compute units per month to pro users, meaning that the access to premium GPUs were limited. This affected the training time and size of models I was able to create.

This limitation also meant that I was unable to train the GANs to produce higher resolution images. The images produced by the GANs mentioned in this thesis have a resolution of  $128 \times 128$ , in future research the CNNs may possibly be improved by using a resolution consistent with that of the dataset.

### 5.1.2 Run time Limits in Google Colab Pro

Due to run time limits I was also frequently met with disconnects when training larger models, this meant that during the process of training the model the run time would disconnect and I would be forced to run the model again. This is due to Google conserving computational resources and limiting the amount of time a model can train while being idle. I was able to mitigate this somewhat by following advice from a stack overflow post and including the following code:

```
1 import IPython
2 js_code = '''
3 function ClickConnect(){
4 console.log("Working");
5 document.querySelector("colab-toolbar-button#connect").click()
6 }
7 setInterval(ClickConnect,60000)
8 '''
9 IPython.display.Javascript(js_code)
```

The above code was used to click the connect button after a certain amount of time to ensure the runtime was not disconnected. There was however an limit to the amount of time this code could be run without the notebook disconnecting which was estimated to be approximately 24 hours.

There was also an issue with Google taking away the use of a TPU and GPU backend, without a TPU or GPU to train the models they could take days to train. Due to this limitation the improvement of transfer learning models was greatly hindered as they require a lot of computational power to train.

### 5.1.3 Lack of Data

During the course of this study I was met with a desire for more data to use to train the GANs and CNNs, I found that the data in the classes which needed augmenting was not nearly enough to train a Generative Adversarial Model to produce perfect X-Rays nor to train a CNN to increase it's generalization ability. This greatly hindered progress when training the GANs as mode collapse frequently occurred and tended to produce black square images which looked just enough like X-Rays to fool the discriminator. If more data were available it may have mitigated a lot of the problems which occurred during the training of the GANs and possibly would have led to more realistic X-Rays being produced and a more various selection of X-Rays.

#### **5.1.4 Bias within Original and Augmented Datasets**

Existing bias within the datasets used in this study could exist as the manual labelling of data is an expensive and laborious process and in many cases labelling is done by small teams or a singular individual. It is therefore necessary to take the results of this study with a grain of salt. The GAN model's trained could also exhibit biased results as there may be over-represented features in some of the synthetic images or it may have been trained on a biased dataset.

#### **5.1.5 Time**

Time was a major limitation during the writing of this thesis as Convolutional Neural Networks and Generative Adversarial Models can take a very long time to train and develop. Due to the time-consuming trial and error effort of adjusting the hyper parameters of models and rerunning the models to compare results of previous implementations I was spending a lot of my time waiting for models to train so that I could analyze the results. This became especially cumbersome as mode collapse occurred many times when training the GANs. The issue of time was also exacerbated by the computational limits of Google Colab which only allows a certain amount of memory and computational power to be allocated to the user.

#### **5.1.6 Financial Limitations**

The training of very large models was limited due to financial limitations, as of today's date Google charges 11.38 euro for 100 compute units and 51.97 euro for 500 compute units. This meant I was only able to train the GAN models for a certain number of epochs which I limited to 100 epochs to conserve compute units.

#### **Assessment of Data Quality**

The models could have possibly benefited from the assessment of synthetic data quality. The augmented sets could be reviewed by doctors to determine if the synthetic images share the same characteristics as the original as some images which may seem high quality to the untrained eye may be lacking in features which would contribute to the diagnosis of the patient.

### **5.2 Future Research**

This section will discuss future research into this problem domain and information which may be valuable to those wishing to explore and expand the use of GANs in the recreation of X-Ray / CT images.

### **5.2.1 Suggestions for Future Research**

#### **Advancements in The Field of Artificial Intelligence**

At the time this thesis was written, there has been much research and many advancements taking place in regards to Generative Adversarial Networks, Convolutional Neural Networks, synthetic data generation, and in the overall field of Artificial Intelligence. I advise researchers who wish to expand on this problem domain and this research to research new methodologies and advances in this field as technology moves at such a rapid pace and undoubtedly the implementation of the networks contained within this thesis will become archaic and under perform in comparison to the latest and greatest implementations of such networks.

The use of synthetic data appears to contain great promise for making data more ubiquitous and to encourage many people to enter the field of Machine Learning and Artificial Intelligence due to the abundance of data throughout various fields. Not only could the generation of synthetic data encourage new people to enter the fields of Machine Learning and Artificial Intelligence, but it would also yield more robust models of CNNs and machine learning models in general which will perhaps be able to generalize better than our current models and assist experts in a variety of fields.

#### **Conducting Experiments with More Data**

With more data around COVID-19 becoming public it may be possible at a future date to conduct these experiments with more data. More data would have greatly improved the training and performance of both the Convolutional Neural Networks and Generative Adversarial Networks. Advancements in medical imaging technology may also have a positive effect upon future research as would the use of standardised and high quality datasets.

I would therefore advise those looking to expand upon this research to seek out more datasets which will hopefully be more readily available in the future.

## **5.3 Conclusion of Work**

### **5.3.1 Issues Faced and How They Should be Mitigated in Future Research**

In this section I will discuss issues which were faced when completing this research and suggest ways in which they could be mitigated.

#### **Slow Training of Models Due to Lack of Computational Resources**

This issue could be mitigated by investing in faster hardware, due to the lack of an NVIDIA GPU the models were trained using Google Colab which can be slow (especially when using the free tier). To mitigate this issue I strongly suggest future researchers invest in a powerful

NVIDIA GPU as NVIDIA has invested a lot of money into AI research and unlike AMD, NVIDIA has compatibility with most ML / AI frameworks.

### **Poor Quality Synthetic Data**

This issue could be mitigated by having a dedicated team to analyze the quality of each synthetic image and prune those which were of poor quality. Due to time-constraints we were unable to review each and every image which may have caused some models to perform worse than they normally would have when trained on pruned synthetic images.

#### **5.3.2 Final Words**

During the course of this thesis I learned a lot about the field of Artificial Intelligence and best practices. I also learned a number of new concepts and techniques when training and evaluating models. The research was complex and tough at times but it was extremely rewarding. The knowledge I have gained from completing this thesis is priceless and it has sparked an interest in the field of Artificial Intelligence which I hope will last a lifetime.

I hope that through the research conducted in this thesis other researchers can explore the use of synthetic data across a number of fields and areas. I also hope that this thesis provides insight to said researchers on how to go about and conduct their studies.

In closing although tough this thesis has been an incredible journey from start to finish. I would like to end this thesis with a quote from a scientist I admire which I find to be quite fitting

“We can only see a short distance ahead but we can see plenty there that needs to be done.” - Alan Turing

# Bibliography

- [1] A. Binkheder *et al.*, “COVID-19 Data Explorer,” [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8701465/>.
- [2] S. Maior *et al.*, “Convolutional Neural Network Model Based on Radiological Images to Support COVID-19 Diagnosis: Evaluating Database Biases,” [Online]. Available: [https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0247839#sec025/](https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0247839#sec025).
- [3] A. Goodfellow *et al.*, “Generative Adversarial Nets,” 2014. [Online]. Available: <https://arxiv.org/pdf/1406.2661.pdf>.
- [4] P. Zhu *et al.*, “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks,” 2020. [Online]. Available: <https://arxiv.org/pdf/1703.10593.pdf>.
- [5] A. Karras Laine, “A style-based generator architecture for generative adversarial networks,” 2019. [Online]. Available: <https://arxiv.org/pdf/1812.04948.pdf>.
- [6] K. Oord Kalchbrenner, “Pixel recurrent neural networks,” 2016. [Online]. Available: <https://arxiv.org/pdf/1601.06759.pdf>.
- [7] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” [Online]. Available: <https://arxiv.org/pdf/1511.08458.pdf>.
- [8] M. Hashemi, “Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation,” [Online]. Available: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0263-7>.
- [9] J. Delua, “Supervised vs. Unsupervised Learning: What’s the Difference?,” [Online]. Available: <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>.
- [10] Multiple, “Tensorflow,” [Online]. Available: <https://www.tensorflow.org/>.
- [11] Multiple, “Tensorflow Case Studies,” [Online]. Available: <https://www.tensorflow.org/about/case-studies/>.
- [12] Multiple, “Keras,” [Online]. Available: <https://keras.io/>.
- [13] Multiple, “Keras Examples,” [Online]. Available: <https://keras.io/examples/>.

- [14] Multiple, “WHO Coronavirus (COVID-19) Dashboard,” [Online]. Available: <https://covid19.who.int/>.
- [15] Multiple, “WHO Coronavirus (COVID-19) Dashboard Ireland,” [Online]. Available: <https://covid19.who.int/region/euro/country/ie/>.
- [16] Multiple, “COVID-19 Deaths and Cases Statistics,” [Online]. Available: <https://www.cso.ie/en/releasesandpublications/ep/p-covid19/covid-19informationhub/health/covid-19deathsandcasesstatistics/>.
- [17] Our World In Data(multiple), “COVID-19 Data Explorer,” [Online]. Available: <https://ourworldindata.org/explorers/coronavirus-data-explorer?tab=map&facet=none&Metric=Confirmed+cases&Interval=Cumulative&Relative+to+Population=true&Color+by+test+positivity=false&country=USA~ITA~CAN~DEU~GBR~FRA~JPN>.
- [18] a. o. Ahmed Wang, “Automated Detection of COVID-19 Through Convolutional Neural Network Using Chest X-ray Images,” [Online]. Available: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0262052/>.
- [19] C. E. Belman-López, “Detection of COVID-19 and Other Pneumonia Cases using Convolutional Neural Networks and X-ray Images,” [Online]. Available: [http://www.scielo.org.co/scielo.php?script=sci\\_arttext&pid=S0120-56092022000100108/](http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-56092022000100108/).
- [20] J. P. Cohen, P. Morrison, and L. Dao, “COVID-19 image data collection,” *arXiv*, 2020. [Online]. Available: <https://github.com/ieee8023/covid-chestxray-dataset>.
- [21] C. Rahman *et al.*, “COVID-19 Radiography Database,” [Online]. Available: <https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database>.
- [22] Asraf, “COVID19 with Pneumonia and Normal Chest Xray(PA) Dataset,” [Online]. Available: <https://www.kaggle.com/datasets/amanullahasraf/covid19-pneumonia-normal-chest-xray-pa-dataset>.
- [23] F. E. A. E.-S. Walid El-Shafai, “Extensive COVID-19 X-Ray and CT Chest Images Dataset,” [Online]. Available: <https://data.mendeley.com/datasets/8h65ywd2jr/3>.
- [24] Multiple, “CDC Museum COVID-19 Timeline,” [Online]. Available: <https://www.cdc.gov/museum/timeline/covid19.html#Late-2019>.
- [25] A. Maxmen, “Wuhan market was epicentre of pandemic’s start, studies suggest,” [Online]. Available: <https://www.nature.com/articles/d41586-022-00584-8>.
- [26] S. M. Amy Maxmen, “The COVID lab-leak hypothesis: what scientists do and don’t know,” [Online]. Available: <https://www.nature.com/articles/d41586-021-01529-3>.

- [27] E. Gamillo, “Covid-19 Surpasses 1918 Flu to Become Deadliest Pandemic in American History,” [Online]. Available: <https://www.smithsonianmag.com/smart-news/the-covid-19-pandemic-is-considered-the-deadliest-in-american-history-as-death-toll-surpasses-1918-estimates-180978748/>.
- [28] A. Mckeever, “COVID-19 surpasses 1918 flu as Deadliest Pandemic in U.S. History,” [Online]. Available: <https://www.nationalgeographic.com/history/article/covid-19-is-now-the-deadliest-pandemic-in-us-history/>.
- [29] R. Roberts and Jarić, “Dating First Cases of COVID-19,” [Online]. Available: [https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1009620/](https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1009620).
- [30] W. D. Heaven, “Hundreds of AI tools have been built to catch covid. None of them helped.,” [Online]. Available: <https://www.technologyreview.com/2021/07/30/1030329/machine-learning-ai-failed-covid-hospital-diagnosis-pandemic/>.
- [31] A. Shrivastava, “Underfitting Vs Just right Vs Overfitting in Machine learning,” [Online]. Available: <https://www.kaggle.com/getting-started/166897>.
- [32] B. Mahmoudi *et al.*, “A Deep Learning-Based Diagnosis System for COVID-19 Detection and Pneumonia Screening Using CT Imaging,” [Online]. Available: <https://www.mdpi.com/2076-3417/12/10/4825/>.
- [33] I. Islam and Asraf, “A Combined Deep CNN-LSTM Network for the Detection of Novel Coronavirus (COVID-19) using X-ray Images,” [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352914820305621>.
- [34] S. Hochreiter, “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, pp. 107–116, [Online]. Available: <https://doi.org/10.1142/S0218488598000094>.
- [35] G. Chen, “A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation,” pp. 1–10, [Online]. Available: <http://arxiv.org/abs/1610.02583>.
- [36] Tanaka and Aranha, “Data Augmentation Using GANs,” *Proceedings of Machine Learning Research*, pp. 1–16, 2019. [Online]. Available: <https://arxiv.org/pdf/1904.09135.pdf/>.
- [37] B. Chalwa *et al.*, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research* 16, pp. 321–357, 2002. [Online]. Available: <https://arxiv.org/pdf/1106.1813.pdf/>.
- [38] G. He Bai, “ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning,” *IEEE World Congress on Computational Intelligence*, pp. 1322–1328, 2008. [Online]. Available: [https://www.researchgate.net/publication/224330873\\_ADASYN\\_Adaptive\\_Synthetic\\_Sampling\\_Approach\\_for\\_Imbalanced\\_Learning](https://www.researchgate.net/publication/224330873_ADASYN_Adaptive_Synthetic_Sampling_Approach_for_Imbalanced_Learning).

- [39] Wang and Xiao, “Lychee Surface Defect Detection Based on Deep Convolutional Neural Networks with GAN-Based Data Augmentation,” *Agronomy*, vol. 11, 2021. [Online]. Available: <https://www.mdpi.com/2073-4395/11/8/1500>.
- [40] C. Jiang and Wang, “TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up,” *Advances in Neural Information Processing Systems 34*, 2021. [Online]. Available: <https://arxiv.org/pdf/2102.07074.pdf>.
- [41] Girshick *et al.*, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. [Online]. Available: <https://ieeexplore.ieee.org/document/6909475>.
- [42] Multiple, “Hardware and Software Support Reference Guide,” [Online]. Available: [https://docs.amd.com/en-US/bundle/Hardware\\_and\\_Software\\_Reference\\_Guide/page/Hardware\\_and\\_Software\\_Support.html](https://docs.amd.com/en-US/bundle/Hardware_and_Software_Reference_Guide/page/Hardware_and_Software_Support.html).
- [43] Multiple, “Image classification from scratch,” [Online]. Available: [https://keras.io/examples/vision/image\\_classification\\_from\\_scratch/](https://keras.io/examples/vision/image_classification_from_scratch/).
- [44] Z. Ramachandran and Le, “Searching for Activation Functions,” [Online]. Available: <https://arxiv.org/pdf/1710.05941.pdf/>.
- [45] N. Patwardhan, M. Ingalhalikar, and R. Walambe, “Aria: Utilizing richard’s curve for controlling the non-monotonicity of the activation function in deep neural nets,” Jul. 2018.
- [46] Multiple, “Keras Applications,” [Online]. Available: <https://keras.io/api/applications/>.
- [47] Multiple, “ImageNet,” [Online]. Available: <https://www.image-net.org/index.php/>.
- [48] F. Chollet, “DCGAN to Generate Face Images,” [Online]. Available: [https://keras.io/examples/generative/dcgan\\_overriding\\_train\\_step//](https://keras.io/examples/generative/dcgan_overriding_train_step//).
- [49] I. Salián, “NVIDIA Research Achieves AI Training Breakthrough Using Limited Datasets,” 2020. [Online]. Available: <https://blogs.nvidia.com/blog/2020/12/07/neurips-research-limited-data-gan/>.