

RESTRICTED

FINAL YEAR PROJECT REPORT

Deep Learning Application to Classify COVID-19 X-Rays

By

NUST SCHOLAR SYED MOHAMMAD WAQAS ALI

PAK-182211, 92(B) EC



ADVISOR

SQN LDR DR. AMMAR SALEEM

CO-ADVISOR

SQN LDR DR. NAYYER AAFAQ

COLLEGE OF AERONAUTICAL ENGINEERING

PAF ACADEMY ASGHAR KHAN, RISALPUR

(AUGUST 2022)

RESTRICTED

RESTRICTED

Deep Learning Application to Classify COVID-19 X-Rays

By

NUST Scholar Syed Mohammad Waqas Ali

PAK-182211, 92(B) EC



ADVISOR

SQN LDR DR. AMMAR SALEEM

CO-ADVISOR

SQN LDR DR. NAYYER AAFAQ

Report submitted in partial fulfillment of the requirements for the degree
of Bachelors of Engineering in Aerospace (BE Aerospace)

In

COLLEGE OF AERONAUTICAL ENGINEERING

PAF ACADEMY ASGHAR KHAN, RISALPUR

(AUGUST 2022)

i

RESTRICTED

RESTRICTED

Approval

It is certified that the contents and form of the project entitled “**Deep Learning Application to Classify COVID-19 X-Rays**” submitted by **NUST SCHOLAR SYED MOHAMMAD WAQAS ALI** have been found satisfactory for the requirement of the degree.

Advisor: **Sqn Ldr Dr. Ammar Saleem**

Signature: _____

Date: _____

Co-advisor : **Sqn Ldr Dr. Nayyer Aafaq**

Signature: _____

Date: _____

RESTRICTED

Dedication

I want to dedicate this report to the countless number of doctors who fought the COVID-19 pandemic, and lost their lives for the safety and welfare of humanity. The selfless warriors who gave their all while cutting themselves off from their families and loved ones.

RESTRICTED

Acknowledgment

I am thankful to ALLAH ALMIGHTY who enabled me to complete this project with the best of my efforts. I am sincerely thankful to my parents, for their encouragement, moral support and above all, prayers. I also want to thank my project advisor, Dr. Ammar Saleem who guided me in every step of the way, for his everlasting support, and for his constant supervision and help throughout the project phase. Furthermore, I am also thankful to my co-advisor, Dr. Nayyer Aafaq and Dr. Ammar for carefully going over the report drafts, presentations, and teaching me the academic and research skills that I would carry with me all my life.

I would also like to thank my coursemates who uplifted my spirits whenever I faced any difficulty.

RESTRICTED

Abstract

The first case of the COVID 19 Pandemic was confirmed in Wuhan, China on December 31st. Since then, it has drastically affected the course of this world. The healthcare systems were pushed to their limits since more and more people required screening. Reverse Transcription Polymerase chain reaction (RT-PCR) was the definitive test for the detection of the virus, but the need for better and faster results was increasing, and therefore medical advancement in COVID 19 Testing was required. Chest X-ray based disease classification has emerged as an alternative method to help diagnosis of COVID 19. In this paper, a model is trained that is able to classify between a COVID-19 infected and a Normal CXR. The family of ResNets, and its variants which are convolutional neural networks were trained using transfer learning on the dataset and the experimental results were compared. The maximum accuracy achieved was 88.43% on the ResNet-50 model.

Keywords—Convolutional neural networks, transfer learning, CXR, COVID-19, disease classification

Table of Contents

1	INTRODUCTION	1
1.1	Project Title	1
1.2	Project Description	1
1.3	Motivation	1
1.4	Project Scope	2
2	LITERATURE REVIEW	4
2.1	Background and Overview	4
2.2	Artificial Intelligence in Medicine	4
2.3	Existing COVID-19 Diagnostic Tools	6
2.3.1	RT-PCR	7
2.3.2	Lateral Flow Tests	7
2.4	Covolutional Neural Networks	8
2.4.1	Convolutional Layer	9
2.4.2	Pooling Layer	9
2.4.3	Dropout	10
2.4.4	Fully Connected Layer	10
2.5	Pre-trained Models	10
2.6	Previous Works	11
2.7	Available Web Applications Relating to Online Detection of COVID 19 through CXRs	15

RESTRICTED

3	Dataset, Environment, and Dependencies	16
3.1	Dataset	16
3.2	Environment	17
3.3	Dependencies	18
3.3.1	Tensorflow	18
3.3.2	Numpy	18
3.3.3	OpenCV	19
3.3.4	Flask	19
4	Methodology	21
4.1	Phase 1: Model Development	21
4.2	Phase 2: Development of Application, and Model Integration	21
5	Phase 1: Model Development	23
5.1	Preprocessing	23
5.2	Classification of Chest X-Rays	25
5.3	Training	27
5.4	Experimental Results	29
6	Phase 2: Development of Application, and Model Integration	34
7	CONCLUSIONS AND FUTURE WORK	36
A	Codes	37
A.1	Building and Training Model (ResNet-152) Code	37
A.2	Flask Application Code	45
A.2.1	app.py Code	45
A.2.2	index.html Code	46

List of Figures

2.1	An algorithm that learns the basic anatomy of a hand and can rebuild where a missing digit should be is shown in the image above. Hand X-rays are used as input, and the output is a trace of where missing hand parts should be. The hand outline is the model in this case, which may be generated and applied to other photos. This could help doctors to see where a limb should be reconstructed or where a prosthetic should be placed.	5
2.2	The left panel shows the image fed into an algorithm. The right panel shows a region of potentially dangerous cells, as identified by an algorithm, that a physician should look at more closely.	6
3.1	Sample images from each class in the COVID-19 Chest Radiography dataset	17
3.2	The libraries used in this project	20
5.1	The outputs of HE, and CLAHE are shown along with the original image. It can be seen how much the contrast has improved when compared with the original image	25
5.2	The image at each stage of pre-processing is displayed	26

RESTRICTED

5.3	Comparing the different models with ResNet-34. [1]	28
5.4	The operation of classification with all the blocks shown	30
5.5	The loss curve for the training of ResNet-50	31
5.6	The training curve for ResNet-50 architecture	32
5.7	Confusion Matrix of the test data with ResNet-50	32
5.8	Two examples chosen randomly after the classification of ResNet-50	33
6.1	A web-based Application GUI of COVID19Vision	34
6.2	Working of COVID19Vision	35

Chapter 1

INTRODUCTION

1.1 Project Title

The title of the project is “Deep Learning Application to classify COVID-19 X-Rays”.

1.2 Project Description

The goal of this research is to train a model that can accurately recognize COVID 19 in CXR and then apply that model in a mobile or online application. Given that CXRs are readily available, one of the key aims of this project is to create a web application or a mobile application that can be easily utilized by the general public to detect the existence of viruses. Because the model will be available online, it can also assist doctors in diagnosing the infection.

1.3 Motivation

COVID-19 is a virus that causes severe respiratory illnesses ranging from the ordinary cold to life-threatening infections such as Severe Acute Respi-

RESTRICTED

ratory Syndrome (SARS) and Middle East Respiratory Syndrome (MERS) (MERS).

The principal symptoms of COVID-19, according to WHO reports, are similar to those of the ordinary flu: fever, weariness, dry cough, shortness of breath, pains, and sore throat [2]. The similarity of COVID-19 symptoms to flu symptoms makes early diagnosis of the coronavirus problematic. The coronavirus, like other viruses and bacteria, has been discovered to cause pneumonia in some people, and the treatment for coronavirus-induced pneumonia differs from that for other types of pneumonia.

One of the quickest ways to diagnose patients is to use radiography and radiology images to detect the condition. Early research revealed distinct abnormalities in the chest radiographs of COVID-19-infected patients.

Advances in artificial intelligence have enabled the implementation of algorithms powered by convolutional neural networks to detect COVID 19 through Chest X-rays. These algorithms learn from the CXR images that are already available from different public sources and improve their accuracy throughout time. It saves a great amount of time for the doctors after which the patient with COVID 19 is confirmed through PCR testing, and then dealt accordingly.

1.4 Project Scope

In December 2019, the first case of novel coronavirus pneumonia was confirmed in Wuhan, China. In just a matter of months, the number of active cases increased exponentially. By September 2020, the virus has spread all over the world with hundreds of deaths each day.

Due to such an increased amount of COVID 19 patients, and limited amount of COVID tests, it was a dire need by health care professionals of

RESTRICTED

an alternate screening tests, which would be able to speed up the process of detecting positive patients of COVID 19.

Early studies identified abnormalities in chest X-ray images of COVID-19 infected patients that could be beneficial for disease diagnosis. Therefore, chest X-ray image-based disease classification has come out as an alternative to aid to the diagnosis of COVID 19.

However, the manual detection of the virus from the X ray is prone to human error. Hence, the deep learning algorithms has proved to enhance the diagnosis process by detecting the presence of COVID 19 from radiography images. [3] Therefore, the artificial intelligence-based system developed in this project will have an efficient COVID-19 detection, and will consist of a user-friendly application that has the capacity to become a rapid COVID-19 diagnosis method in the near future.

Chapter 2

LITERATURE REVIEW

2.1 Background and Overview

Before diving into the building of the model, first we need to understand the features that show COVID-19 in a CXR, and how can we exploit the power of AI to successfully diagnose different diseases such as COVID-19.

2.2 Artificial Intelligence in Medicine

Becoming a doctor is part of a process. A doctor trains himself throughout medical school, they observe and practice surgery repetitively and learn from mistakes to make themselves accurate at what they do. A similar process is seen when an AI algorithm is trained to do its job. In order to make an effective AI Algorithm, it is provided with a lot of data which is structured which means that each data point has a label that is recognizable to the algorithm. The Algorithm is fed with enough data points and data labels so that it is able to differentiate between different test labels. Hence, a test data, which the answers are already known is input to check the performance of the AI Algorithm. After which the according to the results, the performance can be improved by changing the algorithm accordingly and

train it again, just like a doctor [4].

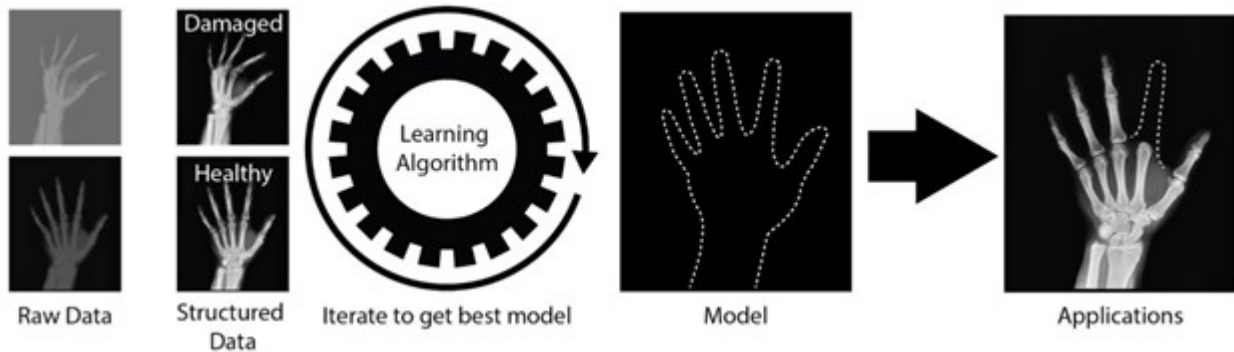


Figure 2.1: An algorithm that learns the basic anatomy of a hand and can rebuild where a missing digit should be is shown in the image above. Hand X-rays are used as input, and the output is a trace of where missing hand parts should be. The hand outline is the model in this case, which may be generated and applied to other photos. This could help doctors to see where a limb should be reconstructed or where a prosthetic should be placed.

The increasing amount of computation power paired with massive amounts of data being produced by the medical industry daily can be used to produce AI solutions that will cover an extensive part of medical industry problems relating to time consuming tests, and misdiagnosis.

For example, recently in 2018 researchers at Seoul National University Hospital and College of Medicine developed and validated a deep learning-based automatic detection algorithm (DLAD) for malignant pulmonary nodules (cancer cells) on chest radiographs and to compare its performance with physicians including thoracic radiologists [5]. The algorithm's performance was compared to multiple physician's detection abilities at the observer performance test on the same images and the algorithm outperformed 17 of 18 doctors, and it enhanced physicians' performances when used as a second reader.

One more example comes from Google AI Healthcare. In 2018, AI researchers at Google AI Healthcare created an algorithm called LYNA [6].

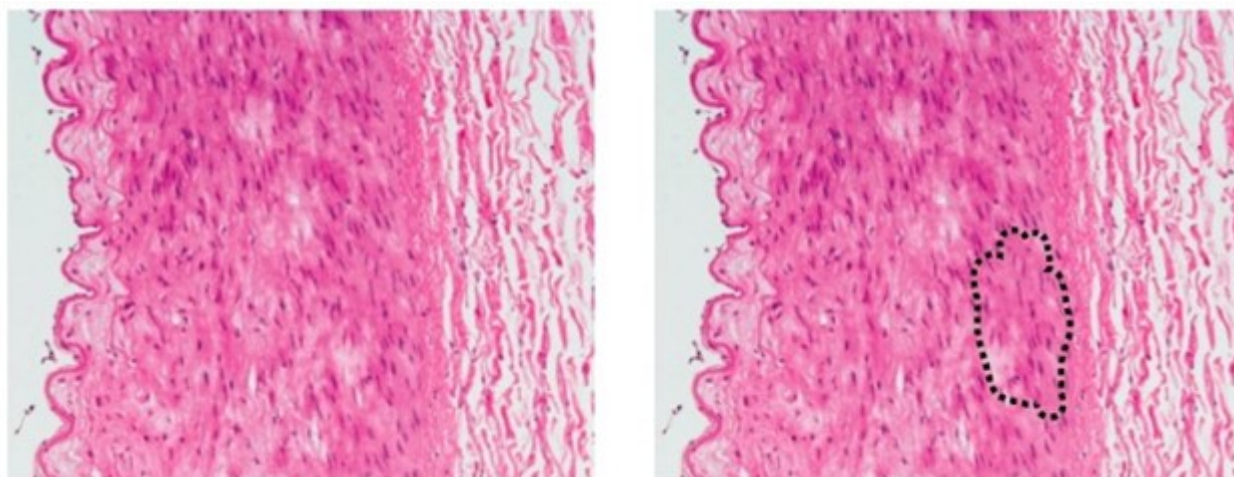


Figure 2.2: The left panel shows the image fed into an algorithm. The right panel shows a region of potentially dangerous cells, as identified by an algorithm, that a physician should look at more closely.

They used histology slides to or detect metastatic breast cancer in sentinel lymph node biopsies. Surprisingly, this algorithm could identify suspicious regions undistinguishable to the human eye in the biopsy samples given, being correct 99% of the time.

Both DLAD and LYNA are the prime examples of algorithms that are aiding the physicians in diagnosing by classifying healthy and diseased samples by showing them the features of the radiographic image that corresponds to the specific disease and exemplify the potential strengths of algorithms in medicine.

2.3 Existing COVID-19 Diagnostic Tools

Limited number of detection methods of COVID 19 were available, and unfortunately were not available to everyone. Some of the tools for detection of COVID 19 [7] are described below:

2.3.1 RT-PCR

Reverse-Transcription Polymerase Chain Reaction, also called RT-PCR test, is the most common COVID-19 test used across the globe. It is considered as the gold standard for viral testing, with almost all hospitals using it for COVID detection [8]. Despite being the universal standard, RT-PCR has many major drawbacks. It is an expensive test which requires a lot of resources and trained staff to carry out, which is also the reason why it is not available in many poor countries. The results take a lot of time, sometimes days to come back. It also puts the health workers in direct contact with the coronavirus. Hence, in some cases it can be quite inconvenient.

2.3.2 Lateral Flow Tests

LFTs are very similar to RT-PCR tests, requiring a nose swab sample. The only feature that differentiates them is that it indicates the with a colored line on the test sample if there COVID-19 present. This way, the results come in within 30-40 minutes without the need of sending the sample to the laboratory. The drawbacks of these tests are inaccuracies or low diagnostic sensitivity. The reason for this is that LFT requires 10,000 particles per mL which is high for early stage detection as compared to 300 particles per mL for RT-PCR [9]. A research conducted by scientists at University of Liverpool showed that LFTs were able to detect only 48.89% COVID-19 infected patients. Therefore, using LFTs for COVID tests are very inconvenient.

Tests such as RT-PCR required trained medical staff and expensive resources which was not feasible to accommodate the huge demand of tests due to the rising number of cases. A quicker and more reliable detection

RESTRICTED

Table 2.1: Comparing some common COVID 19 diagnostic tools.

RT-PCR	LFT
Gold standard for COVID-19 testing	A rapid testing solution. Needs to be confirmed by RT-PCR.
Samples sent to laboratory for results	Samples are not sent to the laboratory.
Results can take days.	Results are available in 30-40 minutes
High diagnostic sensitivity	Low diagnostic sensitivity
300 particles per mL for early stage detection	10000 particles per mL for early stage detection
Requires skilled staff to carry out procedures	Can be done patients themselves
Drawbacks: Expensive, requires resources and equipment	Drawbacks: Inaccurate

method was required to fill the huge inefficiency gap. Researchers and data scientists around the world started testing deep learning techniques to produce algorithms that can easily detect COVID 19 from images such as Chest X-rays of the patient.

2.4 Covolutional Neural Networks

In a CNN, the input to the algorithm is a tensor which has a particular shape given by:

$$\begin{aligned} &(\text{Number of inputs}) \times (\text{Input height}) \times \\ &(\text{Input width}) \times (\text{Input channels}) \end{aligned}$$

A CNN consists if layers, which are deccribed in the next sub-sections.

2.4.1 Convolutional Layer

When the input is passed through a convolutional layer, it creates an image processing kernel which contains a number of filters, after which the tensor input becomes a feature map (i.e. motifs). The feature map is also called an activation map. It has the following dimensions:

$$\begin{aligned} &(\text{Number of inputs}) \times (\text{Feature map height}) \times \\ &(\text{Feature map width}) \times (\text{Feature map channels}) \end{aligned}$$

The Convolutional Layer convolve (set of dot products) the input and pass the output to the next layer, and each convolutional layer processes the data only for its respective field. This layer determines the features in the patterns in the input image, and is very useful in analyzing multi-dimensional data (e.g. images) [?].

2.4.2 Pooling Layer

The Pooling layer reduces the spatial dimensions of output volume. It does so by reducing the number of feature maps, and network parameters. It combines the output of neuron clusters at one layer into a single neuron in the successive layer. Global Pooling is a pooling layer which acts on all the neurons of the feature map. The two types of pooling layers which are popularly used are max pooling layer, and average pooling layer. The task of an average pooling layer is to take the average value of each local cluster of neurons in the feature map, while a max pooling layer takes the maximum value. Furthermore, pooling layer helps improve the generalization of the model by reducing overfitting [10]. The output of the pooling layer is then a combination of features which are invariant to translational shifts and distortions [11].

2.4.3 Dropout

One of the main problems in neural networks is overfitting. It is because a fully connected layer occupies most of the parameters. Dropout is a method that reduces overfitting [12]. It introduces regularization within a network, which in turn improves generalization. It does so by avoiding training all nodes on all training data, therefore decreasing overfitting. This method also improves the training speed of the algorithm.

2.4.4 Fully Connected Layer

After passing the input through the convolutional and pooling layers, the final classification of the input is done through the fully connected layer. The neurons inside the fully connected layer have connections with all the activations in the previous layer. Hence, it outputs nonlinear transformed output via an activation function. As discussed above, it works on the features from all stages and produce a nonlinear set of classification features. The rectified linear unit (ReLU) was used in this step as it helps in overcoming the vanishing gradient problem [13].

2.5 Pre-trained Models

Due to time restrictions or computational restraints, it's not always possible to build a model from scratch which is why pre-trained Convolutional Neural Network (CNN) models there are pre-trained CNN models that can be used to improve our model. A common few models that are used for detection of COVID 19 in Radio-graphic images by scientists who have already worked on this have used the following deep learning algorithms:

- ResNet

- VGG
- Inception
- DenseNet
- SqueezeNet

2.6 Previous Works

The work on this idea has already been started around the globe. Many computer scientists with the aid of doctors have designed and tested models, which have had quite good accuracies for the detection of COVID 19.

- **Natheer et al.** used VGG-16 model to detect COVID 19 with an accuracy of 99% and an F1 Score of 99.1% [14]. They locally collected CXR images from 368 confirmed COVID-19 patients. Data from three freely available datasets was also utilized. The performance was graded in four different ways. For training and testing, the public dataset was used first. Second, data from both public and private sources was used to train and evaluate the models. Third, the model was trained using the public dataset, while the local data were solely used for testing. Finally, the combined data was used for training, while the local dataset was used for testing.

Table 2.2 shows the values of performance evaluations that resulted from the VGG-16 Model. Note that the model achieved its best accuracy of 99% when the fused data set was used for training, and the local data set was used for testing (Fourth Row of the Table 2.2).

The corresponding confusion matrices in Table 2.3 show how much of actual positive cases were predicted to be positive by the model,

RESTRICTED

Table 2.2: Performance evaluation metrics for the VGG-16 model

Dataset	Accuracy	Sensitivity	Specificity	F1-Score	Precision
Public Dataset	97.1%	98.2%	96.5%	95.9%	93.7%
Fused Dataset	98.7%	99.2%	98.4%	98.5%	97.9%
Public Dataset for training and Local Dataset for testing	97.2%	97.8%	96.5%	97.4%	97%
Fused Dataset for training and Local Dataset for testing	99%	99.5%	98.4%	99.1%	98.7%

and how much of the actual negative cases were predicted to be negative by the model. It also shows Type I and Type II errors. Type I error is made by the model when it predicts a case to be negative COVID 19, while its actually positive. Similarly, Type II error is the when the model predicts a case to be positive while its actually negative.

- **Horry et al.** used also used VGG-16 and VGG-19 [15] (contains 19 layers) models to detect COVID 19 with recall and precision rate both equal to 80% [16]. They obtained CXRs from publicly available COVID 19 Image Data Collection [17]. This collection of images is of various size and quality. Since the images were of various sizes and quality, it was first pre-processed before the model would be trained. They did 2 experiments. The first experiment was of Normal vs COVID 19 and Pneumonia. The data set used for this experiment contained 200 Normal CXRs vs 100 COVID 19 CXRs, and 100 Pneumonia CXRs.

The second experiment was of COVID 19 vs Pneumonia. The data set used for this experiment were 100 pneumonia CXRs vs 100 COVID 19 CXRs. The results are summarized below in Table 2.4.

- **Kubra et al.** [18] used SqueezeNet which is a CNN to classify between normal and abnormal CXRs and detect COVID 19 with an

RESTRICTED

Table 2.3: The confusion matrices resulting from the VGG 16 Model

(a) Public Dataset			
		Predicted Diagnosis	
		Positive	Negative
Actual	Positive	164	3
	Negative	11	306
(b) Fused Public and Local Datasets			
		Predicted Diagnosis	
		Positive	Negative
Actual	Positive	236	2
	Negative	5	312
(c) Public Dataset for Training and Local Dataset for Testing			
		Predicted Diagnosis	
		Positive	Negative
Actual	Positive	360	8
	Negative	11	306
(d) Fused Dataset for Training and Local Dataset for Testing			
		Predicted Diagnosis	
		Positive	Negative
Actual	Positive	366	2
	Negative	5	312

Table 2.4: The results from the experiment conducted by Horry et al. [16]

Experiment ID	VGG-16			VGG-19		
Performance Parameters	P	R	F1	P	R	F1
Experiment 1	82	80	80	83	80	80
Experiment 2	83	81	80	83	81	81

RESTRICTED

accuracy of 90.95%.

- **M. E. H. Chowdhury et al.** [19] used transfer learning to train several pre-trained models with and without data augmentation for detection of COVID 19 in CXRs. The highest accuracy achieved was with DenseNet-201 with an accuracy of 98.80%.
- **K. Biçakci and V. Tunalı** [20] used some very popular pre-trained DenseNet, Inception-v3, Inception-ResNet-v2, ResNet, VGG, and Xception models. They also ensembled three popular pre-trained architectures: DenseNet201, Inception-ResNet-v2, and Xception to make an ensemble model which achieved the highest F1 Score of 99%.
- **S. R. Abdani et al.** [21] proposed a lightweight 14-layer convolutional neural network with a modified spatial pooling module, SPP-COVID-Net to detect COVID 19 from CXRs. They deployed the model in mobile phones and tablets which used less than 4 Megabytes of memory. Their method managed to achieve a best mean accuracy of 0.946.
- **Amel et al.** [22] used fine-tuned ResNet-101, ResNet-50, and ResNet-34 pre-trained on ImageNet weights for COVID-19 detection on CXRs. The highest accuracy level was reached with ResNet-34 of 98.34%.
- **Neha et al.** [23] used a different way to detect COVID-19 from CXRs. They decomposed the Chest X-ray into seven modes which were inputs to a multi-scale convolutional neural network [24] into three classes: pneumonia, no-finding, and COVID-19 infected. With 5-fold cross validation scheme, the model obtained a maximum accuracy

of 96%, and 100% on dataset A, and dataset B respectively.

- Other than X-Rays, researchers around the world also used other imaging techniques as well. **A. Kaya et al.** used CT scans instead of CXRs to detect COVID-19 in a patient. They used data-augmentation [25] to increase the size of their dataset due to lack of clean datasets for training. The classification was done using pre-trained CNN networks. The highest accuracies obtained were for VGG-16 and Efficient-NetB3 of 96.5%, and 97.9% respectively.
- **Vipul et al.** [26] used CT scans for COVID-19 detection. They fine-tuned the architecture of pre-trained MobileNet-V2 and made an optimized version for it to be compatible with mobile and edge devices. They achieved a classification accuracy of 96.4%.

2.7 Available Web Applications Relating to On-line Detection of COVID 19 through CXRs

In his paper, Gunther et al. showed the developed web application (link: <https://toad.li/xray>) based on his model based on our model to directly enable users to upload chest X-ray images and detect the presence of COVID-19 within a few seconds [27].

They provide a cutting-edge artificial intelligence-based system for efficient COVID-19 detection, as well as a user-friendly application that has the potential to become a speedy COVID-19 diagnosis method in the near future.

Unfortunately, the link is no longer active.

Chapter 3

Dataset, Environment, and Dependencies

3.1 Dataset

To make a model that differentiates between COVID-19 CXR, and Normal CXR, a dataset must be compiled that will be used for the training of the model. The dataset that was used in this study is the COVID-19 Chest Radiography dataset which was compiled by researchers around Qatar, Bangladesh, Pakistan and Malaysia [28] [29]. The dataset is also the winner of the COVID-19 dataset award by Kaggle Community.

The dataset was last updated in March 2022. It contains a total of 21165 images which consists of 3616 COVID-19 CXRs, 10192 Normal CXRs, 6012 Lung Opacity (Lung infection) CXRs, and 1345 Viral pneumonia CXRs. The classification in this paper is only limited to COVID-19 and normal CXRs, therefore, the viral pneumonia and lung opacity CXRs were not used.

The images in the dataset are all labeled with a resolution of 299 x 299, with 3 channels (RGB). The dataset can be reached at the following link: <https://www.kaggle.com/datasets/tawsifurrahman/covid19->

RESTRICTED

radiography-database

Some sample of images are shown in Figure 3.1.

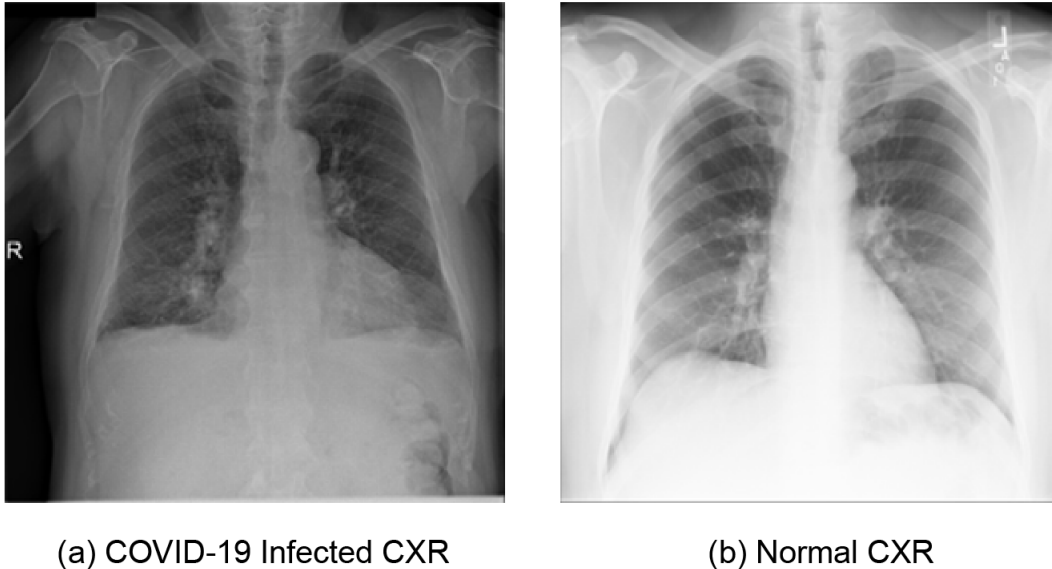


Figure 3.1: Sample images from each class in the COVID-19 Chest Radiography dataset

3.2 Environment

The Google Collaboratory, or just Google Colab environment which is a cloud-based platform was used for building and training the model.

Google Colab is a completely cloud-based Jupyter notebook environment that is free to use. The notebooks you create can be simultaneously modified by your team members, and most significantly, it doesn't require any setup. Many well-known machine learning libraries are supported by Colab like Tensorflow, Pytorch etc. and are simple to load in your notebook.

One of the many benefits of using Google Colab is that it assigns free

GPUs, and TPUs to the user, with High-RAM and disk space, which would be otherwise a very expensive setup.

Model was trained using Intel(R) Xeon(R) CPU @ 2.00 GHz, NVIDIA Tesla T4 70W.

3.3 Dependencies

A code library or package that is used by another piece of software is known as a software dependence. In simple words, these are the libraries that are required for our code to work. Some of the dependencies that were used are described in the next sub-sections below.

3.3.1 Tensorflow

Developed by Google, Tensorflow is a complete open-source framework for building machine learning applications is called TensorFlow. It is a symbolic math toolkit that carries out several operations targeted at deep neural network training and inference using dataflow and differentiable programming. It enables programmers to build machine learning applications utilising a range of instruments, frameworks, and community assets.

Google's TensorFlow is now the most well-known deep learning library in the world. All of Google's products include machine learning to enhance the search engine, translation, picture captioning, or recommendations.

3.3.2 Numpy

A general-purpose toolkit for handling arrays is called NumPy. It offers a multidimensional array object with outstanding speed as well as capabilities for interacting with these arrays. It is the cornerstone Python module

RESTRICTED

for scientific computing. The programme is open-source. It has a number of characteristics.

The equivalent of arrays in Python are lists, although they take a long time to execute. The goal of NumPy is to offer array objects that are up to 50 times quicker than conventional Python lists. In data research, where speed and resources are crucial, arrays are employed a lot. Additionally, it contains matrices, fourier transform, and functions for working in the area of linear algebra.

3.3.3 OpenCV

OpenCV is a sizable open-source library for image processing, machine learning, and computer vision. It currently plays a significant part in real-time operation, which is crucial in modern systems. Using it, one may analyse pictures and movies to find people, objects, and even human handwriting. Python is able to handle the OpenCV array structure for analysis when it is combined with other libraries, such as NumPy. We employ vector space and apply mathematical operations to these characteristics to identify visual patterns and their different features.

In this project, OpenCV was used for applying different functions on the images such as reshaping, converting the image to gray-scale, and also to enhance the contrast of the images using the methods of Histogram Equalization which are built-in in OpenCV.

3.3.4 Flask

A Python package called Flask serves as a web framework that makes it simple to create web apps.

RESTRICTED

Web application developers may design apps without having to be concerned with low-level aspects like protocol, thread management, and other issues thanks to a web framework, also known as a web application framework.

Since Flask is one of the most widely used web frameworks, it is current and cutting-edge. Its capabilities may be simply expanded. It can be scaled up for sophisticated applications.



Figure 3.2: The libraries used in this project

Chapter 4

Methodology

This project was divided into 2 phases. This section briefly explains the phases and how they work.

4.1 Phase 1: Model Development

The data set of COVID-19 affected, and normal X-rays which is available on Kaggle were used to train different models on Google Colab. The family of ResNets such as ResNet-50, ResNet-101, and ResNet-152 were trained, and the validation accuracies were compared. The model with the highest accuracy will be selected, and then will be used in further work.

4.2 Phase 2: Development of Application, and Model Integration

In this phase, the wire frame of the application was developed and the GUI of the application was made using the Flask framework according to the sketched wireframe.

This phase also involved the integration of the trained model with the application. The bugs, and glitches in the application were examined and

RESTRICTED

removed. The GUI requires being user-friendly, and not complex so any complexities involved in the application were fixed in this phase.

Chapter 5

Phase 1: Model Development

In order to take full advantage of the Chest X-rays, image pre-processing of images is done so that the images which are input to the CNN must be as clean as possible. There are several problems that unclean dataset can give rise to. Images may have different contrast or brightness based on the shape of the patient's body and the x-ray dose which he was provided. This gives a wide variation in the contrast and brightness of the images. To counter this effect, Histogram Equalization (HE) and contrast Limited Adaptive Histogram Equalization (CLAHE) techniques were used. HE and CLAHE are discussed in detail in the next subsection.

5.1 Preprocessing

First, the reshape process was applied to all the images. The original images were in a resolution of 299 x 299. ResNet50, which is one of the models used in the classification process uses a resolution of 224 X 224. Hence, a reshape algorithm was used to reshape all the images in the dataset.

After reshaping, histogram equalization was applied to all the images. Histogram equalization is a tool that is used to enhance the contrast of

images. A histogram of an image is a plot of number of pixels against intensity where the number of pixels at a specific intensity is plotted. In a grayscale image, the intensity varies from 0 to 255. Histogram equalization stretches the dynamic range of the image's histogram resulting in overall contrast improvement [30]. It means that the pixel levels are then spread evenly between 0 and 255.

To further improve the contrast around the different zones, CLAHE was applied. Contrast limited adaptive histogram equalization, or CLAHE has produced good results on medical images [31]. CLAHE is a variant of adaptive histogram equalization, which operates histogram equalization in small patches rather than on the whole image. It also restricts contrast amplification. The reason for this is because histogram equalization cannot be implemented in images with very large intensity variation [32]. Hence, CLAHE can be used to increase the contrast of different zones in X-rays such as consolidation zones, ribs, spine, collar bones, and heart [33].

Figure 5.1 shows the comparison between the three images, the original image, and the images processed after Histogram Equalization, and CLAHE.

The stages which the images were run through can be seen in Figure 5.2.

Image augmentation is a very useful technique especially for scarce datasets. It expands the training dataset artificially using rotation, shear, cropping, and a number of preprocessing functions. The problem with small datasets is that after training with less number of images the model is not able to generalize to the validation and test dataset. Thus, it decreases the model's accuracy, and they suffer from the problem of overfitting. To reduce overfitting, several methods were discussed in the paper by Wang

RESTRICTED

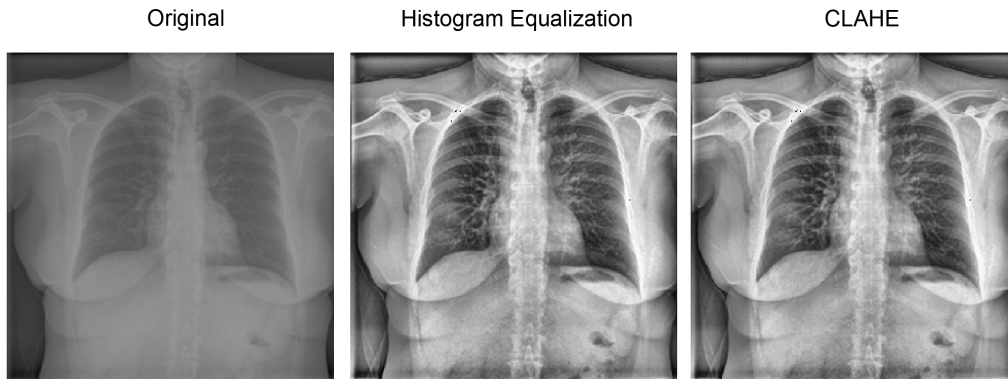


Figure 5.1: The outputs of HE, and CLAHE are shown along with the original image. It can be seen how much the contrast has improved when compared with the original image

et al. [34]. One of the methods is image augmentation. Using image augmentation, overfitting can be minimized, and the accuracy of the model can be increased [35].

5.2 Classification of Chest X-Rays

In this paper, the family of ResNet architectures were used to classify Chest X-ray images into COVID-19 and non-COVID-19. ResNet made its entry in 2015 and sent a wide wave across the computer vision market. Many researchers have looked into the secrets of this architecture, and refined it to make models which perform even better.

As networks were becoming more deeper, they were becoming harder to train. That is because the gradient was becoming smaller and smaller. This leads to the network saturating its performance and starts degrading quickly.

The basic idea of ResNet is to introduce a skip connection or a residual (RESidual) connection that skips one more layer to counter the problem of

RESTRICTED



Figure 5.2: The image at each stage of pre-processing is displayed

small gradients. They learn residual functions with reference to the layer, instead of learning the unreferenced layers. Therefore, the residual mapping is easier to optimize rather than the original unreferenced mapping [1].

Since the introduction of ResNet, many variants of the architecture have come out. Some models have different numbers of layers such as ResNet-50, ResNet-101, and ResNet-200.

In this study, we have trained different models on our dataset such as ResNet-50, ResNet-101, ResNet-152, and ResNet-200.

5.3 Training

To train these models, transfer learning was used. Transfer learning is a very popular approach in machine learning where pre-trained models such as ResNets as discussed above are used as a starting point for new different problems. Since a vast amount of resources and time is required to train those models, it is not feasible to repeat all the steps for a new problem. Therefore, the knowledge or weights from pre-trained models are transferred, and the models adapt to the target problem, which is detection of COVID-19 and non-COVID-19 in this case.

The input images were all in the RGB format which were converted to grayscale in the pre-processing of the images. The preprocessed dataset which was done in the previous section is then used as the input to the ResNet architecture. All the 13822 images were then split into train dataset, test dataset, and validation dataset.

20% of the 13822 images were randomly split into the test dataset. The remaining 80% were then randomly split into validation dataset and train dataset into a ratio of 0.15 and 0.85 respectively. Therefore, the total number of images in each category is shown in Table 5.1.

RESTRICTED

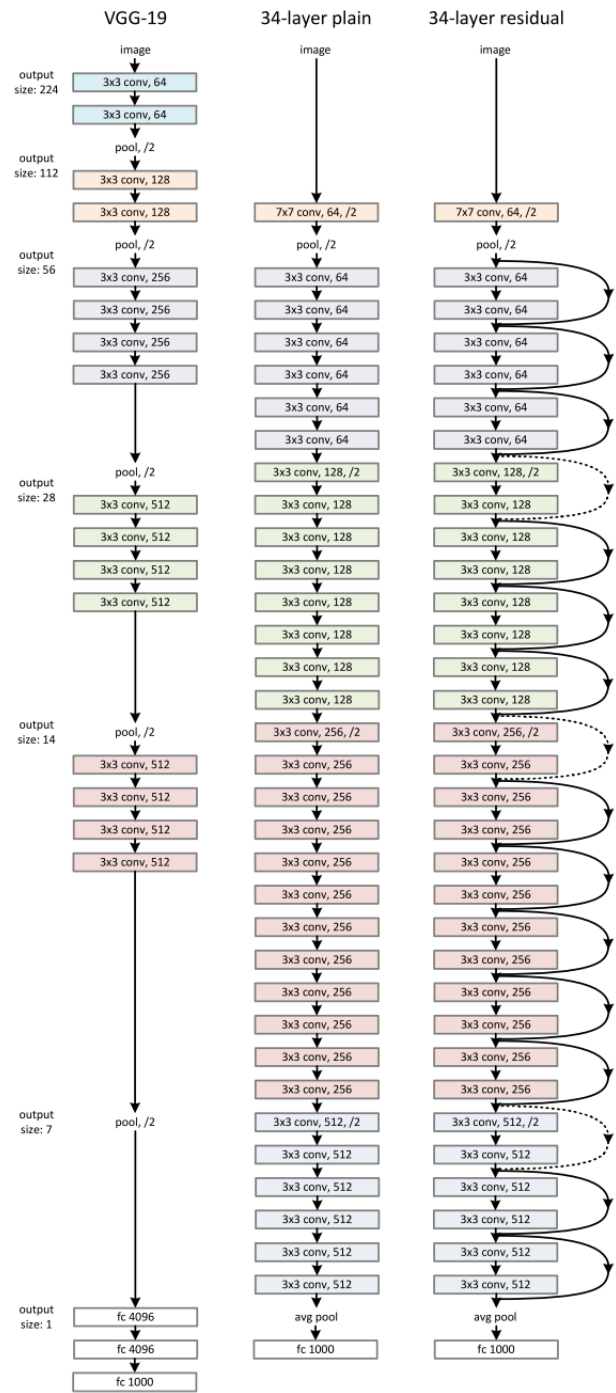


Figure 5.3: Comparing the different models with ResNet-34. [1]

Table 5.1: The number of COVID and normal images in each category for training the model

	Total Images	Images (COVID)	Images (Normal)
Train	9398	2463	6935
Test	2765	733	2032
Validation	1659	430	1229

The loss function used was Binary Crossentropy. The optimizer that was used to compile the model was the Adam optimizer. The default parameters, $\beta_1 = 0.9$, and $\beta_2 = 0.999$ were used. The learning rate was set to 0.001, and the numerical stability constant was set to the default value of $\epsilon = 0.0000001$. The method of training was transfer learning as discussed in the above sections with a batch size of 32. A summary of the operation is shown on Figure 5.4.

5.4 Experimental Results

The ResNet-50 model was then fitted on the train and validation dataset with 10 number of epochs. The experimental results after training are discussed in the next section.

As the result of ResNet-50 model's training was carried out, the classification of COVIS-19 infected CXRs, and Normal CXRs achieved a maximum validation accuracy of 88.43% with 10 number of epochs. The train gloss decreases as the number of epochs increases, and the validation loss decreases overall, with an increasing number of epochs as seen on Figure 5.5.

In a similar fashion, the training curve is shown on Figure 9. It can be

RESTRICTED

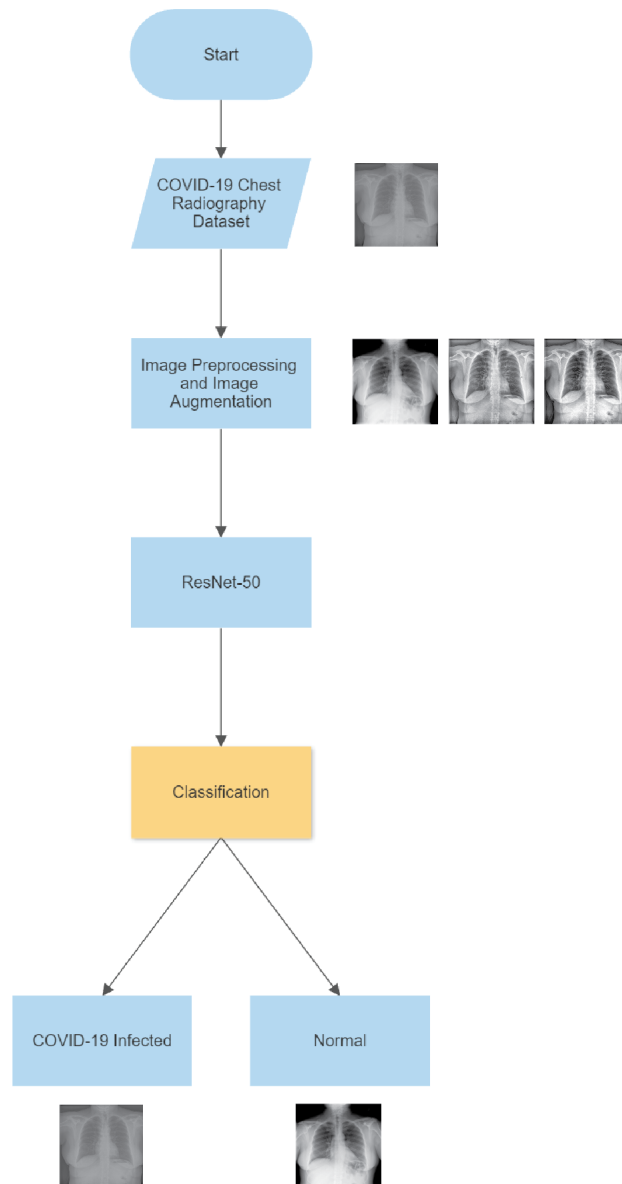


Figure 5.4: The operation of classification with all the blocks shown

RESTRICTED

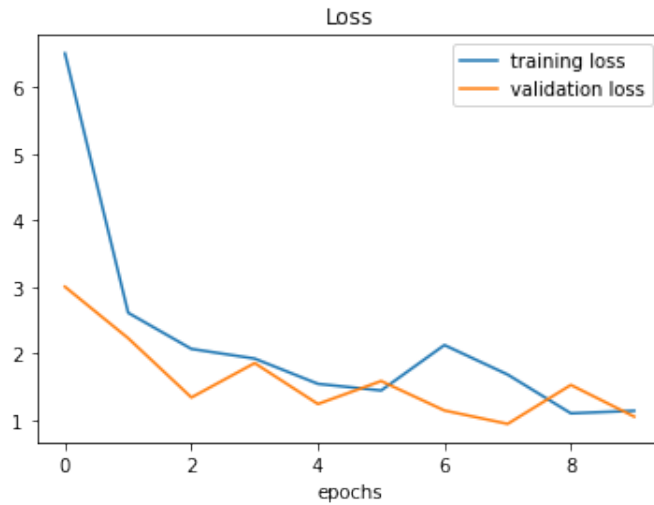


Figure 5.5: The loss curve for the training of ResNet-50

seen that there is an overall increase in the training curve. The reason for this fluctuation in the curves is the non-uniform number of samples in the batch size. Due to less number of COVID-19 CXRs, and very large number of normal CXRs, the samples in each batch size are not the same. This creates uneven batches because of which there is so much fluctuation in the curves.

Similarly, other models were also trained and evaluated. The validation accuracy of all the architectures trained in this study are shown in Table 5.2.

Table 5.2: The summary of validation accuracies of all the architectures used

Model	Validation Accuracy
ResNet-50	88.43%
ResNet-101	78.30%
ResNet-152	85.47%
ResNet-200	75.83%

Among the tested models, ResNet-50, and ResNet-152 performed well.

RESTRICTED

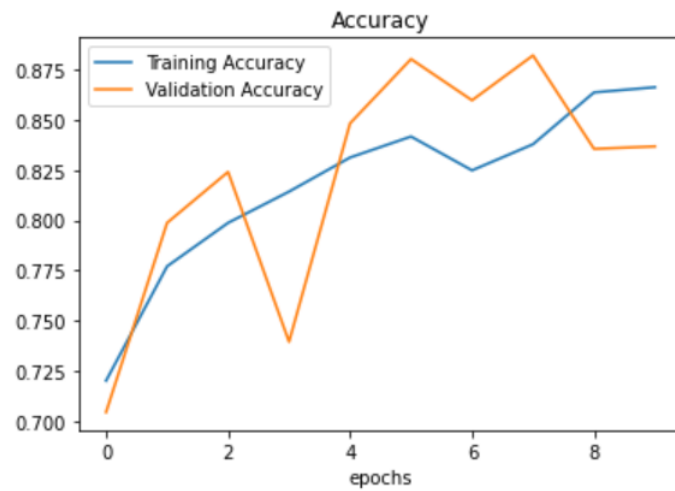


Figure 5.6: The training curve for ResNet-50 architecture

The highest accuracy achieved among these models was by ResNet-50 of 88.43%. The testing of this model was done using the test set that contained a total of 2765 images (733 COVID-19, 2032 Normal). The confusion matrix after the testing of ResNet-50 is shown in Figure 5.7.

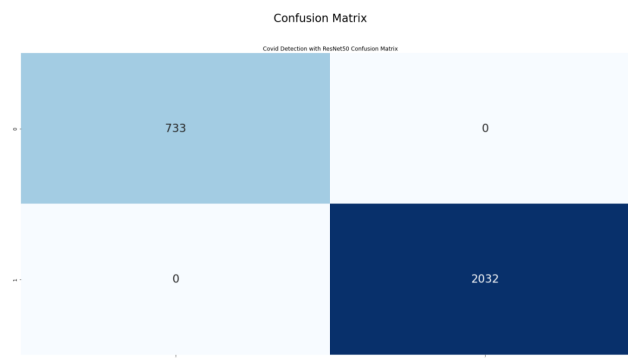


Figure 5.7: onfusion Matrix of the test data with ResNet-50

The evaluation metrics of the ResNet-50 is shown in Table 5.3.

In Figure 5.8, the left CXR was predicted to be a Normal CXR (Actual Class: Normal) by the model with a probability of 98.51%. The right CXR

RESTRICTED

Table 5.3: Evaluation metrics of ResNet-50

Evaluation Metrics	Value
Precision	0.85
Recall	0.85
F1-Score	0.85

was predicted to COVID-19 infected (Actual Class: COVID) by the model with a probability of 97.2%.

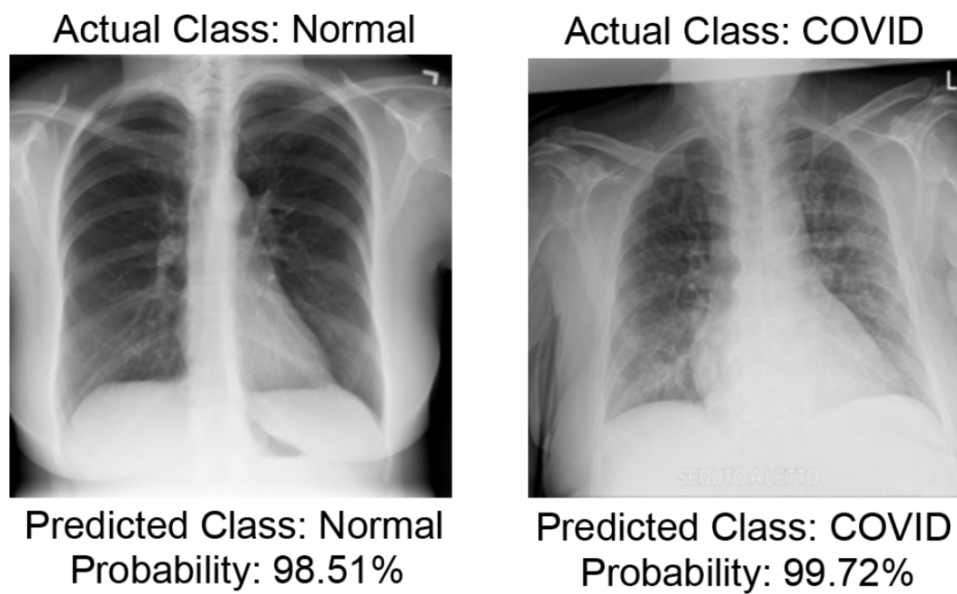


Figure 5.8: Two examples chosen randomly after the classification of ResNet-50

Chapter 6

Phase 2: Development of Application, and Model Integration

The application was developed using the Flask web framework. The application, which was named COVID19Vision has a simple GUI where a person can upload a picture after which the model processes the image indicating whether a person is COVID-19 infected, or Normal with a probability associated with it. The GUI is shown in Figure 6.1.

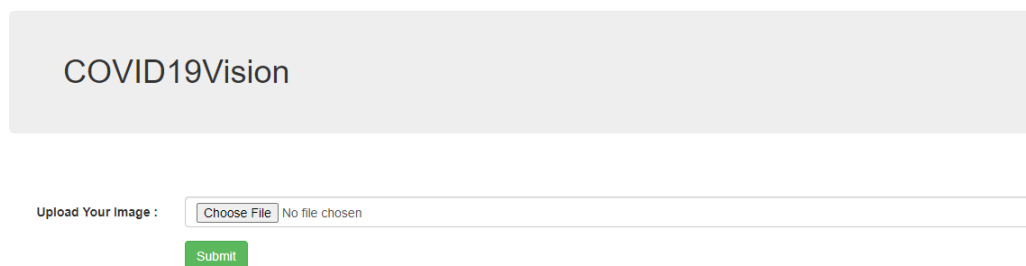



Figure 6.1: A web-based Application GUI of COVID19Vision

An demonstration of the working application can be seen where a person uploads a CXR, and the COVID19Vision indicates the result as shown in Figure 6.2.

RESTRICTED

COVID19Vision

Upload Your Image : No file chosen



The predicted class is : *COVID-19* with the probability of *0.997585*

Figure 6.2: Working of COVID19Vision

Chapter 7

CONCLUSIONS AND FUTURE WORK

The COVID-19 pandemic was a calamity that united people. This pandemic should be seen as an opportunity to advance technical solutions that can handle the torrents of cases efficiently. This pandemic introduced the power of deep learning to the medical industry which showed that the two combined can make a team that can handle the magnitude of a pandemic. In this study, using a dataset of Chest X-rays, we trained the family of ResNets so that they are able to classify between an COVID-19 infected CXR, and a normal CXR. A high accuracy of 88.43% was achieved by ResNet-50.

The high accuracy results open the door to developing mobile and easy-to-use apps that increase diagnosis accuracy, reduce burden for over-worked health providers, and improve healthcare access in understaffed or underequipped locations. This area, as well as the creation and evaluation of multiclass classification models, will be the focus of future research.

Appendix A

Codes

A.1 Building and Training Model (ResNet-152) Code

```
#Installing Dependencies and Loading Data

from google.colab import drive
drive.mount('/content/drive')

cd /content/drive/MyDrive/ColabData

import os
import numpy as np
import random
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
import keras
# Deep learning libraries
from keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf

# Explore Data Analysis

path_normal =
    "/content/drive/MyDrive/ColabData/COVID-19_Radiography_
Dataset/Dataset/Normal/images"
path_covid =
    "/content/drive/MyDrive/ColabData/COVID-19_Radiography_
```

RESTRICTED

```
Dataset/Dataset/COVID/images"

len_normal = len(os.listdir(path_normal))
len_covid = len(os.listdir(path_covid))

print(len_normal)
print(len_covid)

classes = ('Normal', 'COVID-19')
y_pos = np.arange(len(classes))
print(y_pos)
performance = [len_normal, len_covid]

plt.barh(y_pos, performance, align='center', alpha=0.5)
plt.yticks(y_pos, classes)
plt.title('Chest X-ray images')

plt.show()

**Overview Chest X-ray images**

labels = ['Normal', 'COVID-19']
img1 = path_normal+"/Normal-1.png"
img2 = path_covid+"/COVID-1994.png"

imgs = [img1, img2]

fig, ax = plt.subplots(1, 2, figsize=(15, 15))
ax = ax.ravel()
plt.tight_layout()

for i in range(0,2):

    ax[i].imshow(plt.imread(imgs[i]), cmap='gray')
    ax[i].set_title(labels[i])

**Load data**

from keras import preprocessing
from skimage.transform import resize

def resize_image(imgpath):
    img = tf.keras.preprocessing.image.load_img(imgpath)
```


RESTRICTED

```
img = tf.keras.preprocessing.image.img_to_array(img)
img = resize(img, (224,224), anti_aliasing=True)
return img

import os
x_ = list()
y = list()

for i in os.listdir(path_normal):
    try:
        imgpath = path_normal+"/"+i
        img = resize_image(imgpath)
        x_.append(img)
        y.append(0)
    except:
        None

for i in os.listdir(path_covid):
    try:
        imgpath = path_covid+"/"+i
        img = resize_image(imgpath)
        x_.append(img)
        y.append(1)
    except:
        None

x_ = np.array(x_)

print(x_.shape)

**Model Classes**

# Split test, train, validation data

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x_,y,test_size =
    0.2,random_state = 20)

x_train,x_val,y_train,y_val =
    train_test_split(x_train,y_train,test_size = 0.15,random_state
    = 40)
```

RESTRICTED

```
**Classes: Normal, COVID-19, Lung Opacity, Viral pneumonia**

from keras.utils.np_utils import to_categorical

y = to_categorical(y,num_classes = 2)

classNames = ["Normal","COVID_19"]
print(classNames)

print(y)

unique, counts = np.unique(y_train, return_counts=True)
uniqueVal, countsVal = np.unique(y_val, return_counts=True)
dict(zip(unique, counts))

dict(zip(uniqueVal, countsVal))

#Building the Resnet 152 Model

IMAGE_SHAPE= (224,224)
BATCH_SIZE= 32

# Create TensorBoard callback (funtionized beacuse we need to
    create a new one for each model)

import datetime

def create_tensorboard_callback(dir_name, experiment_name):
    log_dir = dir_name + "/" + experiment_name + "/" +
        datetime.datetime.now().strftime("Y&m&d-&H&M&S")
    tensorboard_callback =
        tf.keras.callbacks.TensorBoard(log_dir=log_dir)
    print(f"Saving TensorBoard logfiles to {log_dir}")
    return tensorboard_callback

#Let's compare the following two models
resnet_152_url =
    "https://tfhub.dev/sayakpaul/bit\_r152x2\_224\_feature\_extraction/1"

# import dependencies
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow.keras import layers
```

RESTRICTED

```
# lets make a create_model() function to create a model from URL
def create_model(model_url, num_classes=2):
    """
    Takes a Tensorflow Hub URL and creates a Keras Sequential Model
    with it.

    Args:
        model_url (str): A tensorflow hub feature extractor URL.
        num_classes (int): Number of output neurons in the output
            layer, should be equal to number of target classes, default
            2.

    Returns:
        An uncompiled Keras sequential model with model_url as feature
        extractor layer and Dense output layer with num_classes
        output neurons.
    """
    #Download the pre-trained model and save it as Keras layer
    feature_extractor_layer = hub.KerasLayer(model_url,
                                              trainable=False, #Freeze the
                                              already learned patterns
                                              name="feature_extractor_layer",
                                              input_shape=IMAGE_SHAPE+(3,))

    #Create our own model
    model = tf.keras.Sequential([feature_extractor_layer,
                                  layers.Dense(1, activation="sigmoid",
                                                  name="output_layer")
                                  ])

    return model

#create resnet model
resnet_152_model = create_model(resnet_152_url)

#Compile our resnet model
resnet_152_model.compile(loss="binary_crossentropy",
                        optimizer=tf.keras.optimizers.Adam(),
                        metrics=["accuracy"])

resnet_152_model.summary()

y_train, y_val = np.array(y_train), np.array(y_val)
```

RESTRICTED

```
print(y_train)
print(y_val)

# Lets fit the model to our data
history = resnet_152_model.fit(x_train, y_train, epochs=5,
    validation_data=(x_val, y_val), batch_size = 32,
    callbacks=[create_tensorboard_callback(dir_name="tensorflow_hub",
        experiment_name="Resnet152")])

# Test and Evaluate Model

**Predict with test data**

y_test=np.array(y_test)
print(y_test)

# Get predictions
mypredict = model.predict(x_test)

pred_labels= list()

for label in mypredict:
    rounded_labels=np.round_(label)
    rounded_labels=rounded_labels.astype(int)
    pred_labels.append(rounded_labels[0])
pred_labels=np.array(pred_labels)
print(pred_labels)

evaluate = model.evaluate(x_test, y_test)

print("Accuracy: {:.2f}%".format(evaluate[1] * 100))
print("Loss: {}".format(evaluate[0]))

# Confusion matrix
```

A confusion matrix **is** a summary of prediction results on a classification problem. The number of correct **and** incorrect predictions are summarized with count values **and** broken down by each **class**. This **is** the key to the confusion matrix. The confusion matrix shows the ways **in** which your classification model **is** confused when it makes predictions. It gives us insight **not** only into the errors being made by a classifier but

RESTRICTED

more importantly the types of errors that are being made.

```
# Example of a confusion matrix in Python
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

cm = confusion_matrix(y_test, pred_labels)

plt.figure(figsize=(24,12))

plt.suptitle("Confusion Matrix",fontsize=24)
plt.title("Covid Detection with ResNet50 Confusion Matrix")
sns.heatmap(cm,annot=True,cmap="Blues",fmt="d",cbar=False,
            annot_kws={"size": 24})

plt.show()

print ('Accuracy Score :',accuracy_score(y_test, pred_labels ))
print ('Report : ')
print (classification_report(y_test, pred_labels ))

#Graph of Loss Functions etc.

import matplotlib.pyplot as plt

#Plot the validation and training curves

def plot_loss_curves(history):
    """
    Returns saperate loss curves for training and validation metrics.

    Args:
        history: Tensorflow history object.

    Returns:
        Plots of training/validation loss and accuracy metrics.
    """
    loss = history.history["loss"]
    val_loss = history.history["val_loss"]
```

RESTRICTED

```
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]

epochs = range(len(history.history["loss"]))

#Plot loss
plt.plot(epochs, loss, label="training loss")
plt.plot(epochs, val_loss, label = "validation loss")
plt.title("Loss")
plt.xlabel("epochs")
plt.legend()

#Plot Accuracy
plt.figure()
plt.plot(epochs, accuracy, label = "Training Accuracy")
plt.plot(epochs, val_accuracy, label = "Validation Accuracy")
plt.title("Accuracy")
plt.xlabel("epochs")
plt.legend()

plot_loss_curves(history)

#Prediction of a Single Image

from numpy.core.fromnumeric import reshape
img_path='/content/Normal-4.png'
img = tf.keras.preprocessing.image.load_img(img_path)
img = tf.keras.preprocessing.image.img_to_array(img)
img = resize(img, (100,100), anti_aliasing=True)
img= img.reshape(1,100,100,3)
predict=model.predict(img)
prediction=np.round_(predict).astype(int)
predicted_class=classes[prediction[0][0]]
if prediction[0][0]==0:
    probability=1-predict[0][0]
else:
    probability=predict[0][0]
print("The predicted class is", predicted_class, "with the
      probability of", probability)

model.save('resnet_model.h5')
```

A.2 Flask Application Code

A.2.1 app.py Code

```
import tensorflow_hub as hub
from flask import Flask, render_template, request
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import load_model
import numpy as np
import tensorflow as tf
from skimage.transform import resize
from numpy.core.fromnumeric import reshape

app = Flask(__name__)

classes = ('Normal', 'COVID-19')

resnet_model= tf.keras.models.load_model('resnet_model.h5',
    custom_objects={'KerasLayer':hub.KerasLayer})

def predict_label(img_path):
    img = tf.keras.preprocessing.image.load_img(img_path)
    img = tf.keras.preprocessing.image.img_to_array(img)
    img = resize(img, (100, 100))
    img = img.reshape(1, 100, 100, 3)
    predict = resnet_model.predict(img)
    prediction = np.round_(predict).astype(int)
    predicted_class = classes[prediction[0][0]]
    if prediction[0][0] == 0:
        probability = 1 - predict[0][0]
    else:
        probability = predict[0][0]
    return predicted_class, probability

#routes
@app.route("/", methods=['GET','POST'])
def kuch_bhi():
    return render_template("index.html")

@app.route("/about")
def about_page():
```

RESTRICTED

```
str = "COVID 19 Image Classification. From Party_Boy69. All
      right not reserved right now, but someday eventually, bitch"
return str

@app.route("/submit", methods=['GET', 'POST'])
def get_hours():
    if request.method == 'POST':
        img = request.files['my_image']
        img_path = "static/" + img.filename
        img.save(img_path)

        pred_class , prob = predict_label(img_path)

    return render_template("index.html", prediction = pred_class ,
                          probability = prob, img_path = img_path)

if __name__ == '__main__':
    app.debug=True
    app.run(host='0.0.0.0', port=3000)
```

A.2.2 index.html Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Image Classification</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width,
    initial-scale=1">
  <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/
bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/
3.5.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/
3.4.1/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
```


RESTRICTED

```
<h1 class="jumbotron bg-primary">COVID19Vision</h1>
<br><br>
<form class="form-horizontal" action="/submit" method="post"
  enctype="multipart/form-data">

  <div class="form-group">
    <label class="control-label col-sm-2" for="pwd">Upload Your
      Image :</label>
    <div class="col-sm-10">
      <input type="file" class="form-control" placeholder="Hours
        Studied" name="my_image" id="pwd">
    </div>
  </div>

  <div class="form-group">
    <div class="col-sm-offset-2 col-sm-10">
      <button type="submit" class="btn
        btn-success">Submit</button>
    </div>
  </div>
</form>

  {% if prediction %}

  <h2> The predicted class is : <i> {{prediction}} </i> with the
    probability of <i> {{probability}} </i></h2>

  {% endif %}

</div>

</body>
</html>
```

Bibliography

- [1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [2] W. Zhang, "Imaging changes of severe covid-19 pneumonia in advanced stage," *Intensive care medicine*, vol. 46, no. 5, pp. 841–843, 2020.
- [3] G. C. Bacellar, M. Chandrappa, R. Kulkarni, and S. Dey, "Covid-19 chest x-ray image classification using deep learning," Cold Spring Harbor Laboratory Press, 2021.
- [4] D. Greenfield and S. Wilson, "Artificial intelligence in medicine: Applications, implications and limitations," *Science in the News*, 2019.
- [5] J. G. Nam, S. Park, E. J. Hwang, J. H. Lee, K.-N. Jin, K. Y. Lim, T. H. Vu, J. H. Sohn, S. Hwang, J. M. Goo, *et al.*, "Development and validation of deep learning–based automatic detection algorithm for malignant pulmonary nodules on chest radiographs," *Radiology*, vol. 290, no. 1, pp. 218–228, 2019.
- [6] Y. Liu, T. Kohlberger, M. Norouzi, G. E. Dahl, J. L. Smith, A. Mohtashamian, N. Olson, L. H. Peng, J. D. Hipp, and M. C. Stumpe, "Artificial intelligence–based breast cancer nodal metastasis detection:

RESTRICTED

- Insights into the black box for pathologists,” *Archives of pathology & laboratory medicine*, vol. 143, no. 7, pp. 859–868, 2019.
- [7] M. Sreepadmanabh, A. K. Sahu, and A. Chande, “Covid-19: Advances in diagnostic tools, treatment strategies, and vaccine development,” *Journal of biosciences*, vol. 45, no. 1, pp. 1–20, 2020.
- [8] M. Yüce, E. Filiztekin, and K. G. Özkaya, “Covid-19 diagnosis—a review of current methods,” *Biosensors and Bioelectronics*, vol. 172, p. 112752, 2021.
- [9] P. England, “Preliminary report from the joint phe porton down university of oxford sars-cov-2 test development and validation cell. rapid evaluation of lateral flow viral antigen detection devices (lfds) for mass community testing,” 2020.
- [10] D. Scherer, A. Müller, and S. Behnke, “Evaluation of pooling operations in convolutional architectures for object recognition,” in *International conference on artificial neural networks*, pp. 92–101, Springer, 2010.
- [11] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in *2007 IEEE conference on computer vision and pattern recognition*, pp. 1–8, IEEE, 2007.
- [12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

RESTRICTED

- [13] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.
- [14] N. Khasawneh, M. Fraiwan, L. Fraiwan, B. Khassawneh, and A. Ib-nian, "Detection of covid-19 from chest x-ray images using deep convolutional neural networks," *Sensors*, vol. 21, no. 17, p. 5940, 2021.
- [15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [16] M. J. Horry, S. Chakraborty, M. Paul, A. Ulhaq, B. Pradhan, M. Saha, and N. Shukla, "X-ray image based covid-19 detection using pre-trained deep learning models," 2020.
- [17] J. P. Cohen, P. Morrison, L. Dao, K. Roth, T. Q. Duong, M. Ghassemi, *et al.*, "Covid-19 image data collection: Prospective predictions are the future," *arXiv preprint arXiv*, 2006.
- [18] K. N. Akpınar, S. Genc, and S. Karagöl, "Chest x-ray abnormality detection based on squeezenet," in *2020 international conference on electrical, communication, and computer engineering (ICECCE)*, pp. 1–5, IEEE, 2020.
- [19] M. E. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al Emadi, *et al.*, "Can ai help in screening viral and covid-19 pneumonia?," *IEEE Access*, vol. 8, pp. 132665–132676, 2020.

- [20] K. Biçakci and V. Tunali, "Transfer learning approach to covid-19 prediction from chest x-ray images," in *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1–5, IEEE, 2021.
- [21] S. R. Abdani, M. A. Zulkifley, and N. H. Zulkifley, "A lightweight deep learning model for covid-19 detection," in *2020 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*, pp. 1–5, IEEE, 2020.
- [22] A. Ksibi, M. Zakariah, M. Ayadi, H. Elmannai, P. K. Shukla, H. Awal, and M. Hamdi, "Improved analysis of covid-19 influenced pneumonia from the chest x-rays using fine-tuned residual networks," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.
- [23] N. Muralidharan, S. Gupta, M. R. Prusty, and R. K. Tripathy, "Detection of covid19 from x-ray images using multiscale deep convolutional neural network," *Applied Soft Computing*, vol. 119, p. 108610, 2022.
- [24] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European conference on computer vision*, pp. 354–370, Springer, 2016.
- [25] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.
- [26] V. K. Singh and M. H. Kolekar, "Deep learning empowered covid-19 diagnosis using chest ct scan images for collaborative edge-cloud computing platform," *Multimedia Tools and Applications*, vol. 81, no. 1, pp. 3–30, 2022.
- [27] G. C. Bacellar, M. Chandrappa, R. Kulkarni, and S. Dey, "Covid-19 chest x-ray image classification using deep learning," *medRxiv*, 2021.

- [28] M. E. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al Emadi, *et al.*, “Can ai help in screening viral and covid-19 pneumonia?,” *IEEE Access*, vol. 8, pp. 132665–132676, 2020.
- [29] T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. A. Kashem, M. T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan, *et al.*, “Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images,” *Computers in biology and medicine*, vol. 132, p. 104319, 2021.
- [30] S.-D. Chen and A. R. Ramli, “Contrast enhancement using recursive mean-separate histogram equalization for scalable brightness preservation,” *IEEE Transactions on consumer Electronics*, vol. 49, no. 4, pp. 1301–1309, 2003.
- [31] A. M. Reza, “Realization of the contrast limited adaptive histogram equalization (clahe) for real-time image enhancement,” *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 38, no. 1, pp. 35–44, 2004.
- [32] A. Mishra, “Contrast limited adaptive histogram equalization (clahe) approach for enhancement of the microstructures of friction stir welded joints,” *arXiv preprint arXiv:2109.00886*, 2021.
- [33] K. N. Akpinar, S. Genc, and S. Karagol, “Chest x-ray abnormality detection based on squeeze-net,” in *2020 international conference on electrical, communication, and computer engineering (ICECCE)*, pp. 1–5, IEEE, 2020.

RESTRICTED

- [34] B. Wang and D. Klabjan, "Regularization for unsupervised deep neural nets," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [35] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.