GPU ENABLED CNN BASED COVID-19 HRCT IMAGES CLASSIFIER

Kavin Raj A C	**(17Z220)**
Pranav Shankar Ramalingam	**(17Z230)**
Preethi S V	**(17Z231)**
Samyuktha G	**(17Z238)**
Varsha Devi K	**(17Z256)**

15Z720 PROJECT WORK I

Dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

November 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous institution)

COIMBATORE – 641 004

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

GPU ENABLED CNN BASED COVID-19 HRCT IMAGES CLASSIFIER

Bonafide record of work done by

Kavin Raj A C	**(17Z220)**
Pranav Shankar Ramalingam	**(17Z230)**
Preethi S V	**(17Z231)**
Samyuktha G	**(17Z238)**
Varsha Devi K	**(17Z256)**

Dissertation submitted in partial fulfilment of the requirements for the degree of

BACHELOR OF ENGINEERING

Branch: COMPUTER SCIENCE AND ENGINEERING

of Anna University

November 2020

....

Dr. Karpagam G R **Dr. Sudha Sadasivam G**

Faculty guide **Head of the Department**

Certified that the candidate was examined in the viva-voce examination held on …….

....

(Internal Examiner) (External Examiner)

CERTIFICATE

Certified that this report titled "**GPU enabled CNN based COVID-19 HRCT Images Classifier**", for PROJECT WORK I (15Z720) is a bonafide work of **Kavin Raj A C (17Z220), Pranav Shankar Ramalingam (17Z230), Preethi S V (17Z231), Samyuktha G (17Z238), Varsha Devi K(17Z256)** who have carried out the work under my supervision for the partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering. Certified further that to the best of my knowledge and belief, the work reported herein does not form part of any other thesis or dissertation based on which a degree or an award was conferred on an earlier occasion.

Place: Coimbatore

Date:

Dr. Karpagam G R

Professor

Department of Computer Science and Engineering,

PSG College of Technology,

Coimbatore - 641004

COUNTERSIGNED

HEAD

Department of Computer Science and Engineering

PSG College of Technology, Coimbatore - 641004

CONTENTS

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K. Prakasan,** Principal, PSG College of Technology for his kind patronage.

We wish to extend sincere thanks to **Dr. Sudha Sadasivam G**. Professor and Head of the Department, Computer Science and Engineering for her moral support and valuable advices.

We wish to sincerely thank **Dr. Vinoth Kumar B**, Associate Professor, Department of Information Technology for his technical support and valuable advice.

We extend our gratitude to **Dr. Maheshwaran Viyannan**, PSG IMSR for giving us this opportunity and valuable advice.

We sincerely thank our project guide, **Dr. Karpagam G R**, Professor and Programme Coordinator, Department of Computer Science and Engineering for having given us an opportunity to pursue our academic interests and also for her constant support throughout the course of the project. We are grateful to her inspiring guidance which motivated us, right from the inception of the project.

We are thankful to all our friends, faculty members and non-teaching staff of the Computer Science and Engineering Department for extending their co-operation towards the completion of this project work.

ABSTRACT

The ongoing COVID-19 pandemic has caused a bottleneck in the healthcare system around the world. The virus is highly contagious as it spreads through air droplets and person to person contact. The standard method of diagnosis for such viruses is the RT PCR (Reverse Transcription-Polymerase Chain Reaction) method. However, there are numerous issues in using RT-PCR as it is prone to producing a large number of false negative and false-positive results because of its low sensitivity and specificity. Also, there are a limited number of RT-PCR testing kits available. In this scenario, medical imaging such as X-ray and Computed Tomography (CT) of the lung of the patients can be used as an alternative tool to make the diagnosis, as the disease primarily targets the epithelial cells of the lung. As CT scanners are widely available, they are considered to be functional and practical diagnostic tools. Artificial intelligence (AI) technologies can be used to strengthen the power of imaging tools and help medical specialists in analyzing hundreds of CT images quickly. Deep learning provides state-of-the-art performance for detection, segmentation, classification, and prediction. Hence, deep learning offers a convenient tool for diagnosing and predicting the severity of lung infection in CT images for COVID-19 patients. We aim to use a 3D U-Net architecture for segmenting the lung from the CT scans and 3D CNN architecture for classification and scoring of the severity of the CT scans.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The COVID-19 infection is an ongoing pandemic caused by Severe Acute Respiratory Syndrome Coronavirus-2 (SARS COVID-2), creating a seismic shift in the economic and socio-political circles across the globe. It has been declared by the world health organization (WHO) as a public health emergency. The disease is highly contagious and spreads from person to person.

The diagnosis is primarily made by RT PCR of the throat swab. However, there are numerous issues in using RT PCR as it is suffered by many false-negative results and limited availability of testing kits. In this scenario, a CT scan of the lung of the patients can be used as an alternative tool to make the diagnosis, as the disease primarily targets the lung of the patients. The widespread availability of CT scanners puts them in an advantageous position as a diagnostic tool.

The primary concerns raised by experts in the utility of HRCT of the lung in the diagnosis of COVID-19 infection are 1. Compromise in maintaining social distancing while performing the scan as the CT scan technicians have to manually position the patient on the scanner. 2. Given the pandemic nature of the disease, the number of scans performed is huge with hundreds of images to be analyzed for each patient. The limited expertise/manpower available may be overwhelmed by the sheer volume of scans, which leads to delay in diagnosis, where valuable time may be lost in making crucial treatment decisions.

Artificial intelligence (AI) technologies can strengthen the power of imaging tools and help medical specialists analyze hundreds of CT images quickly. Deep learning provides state-of-the-art performance for detection, segmentation, classification, and prediction. Hence, deep learning offers a convenient tool for diagnosing and predicting lung infection severity in CT images for COVID-19 patients.

1.1 CONVOLUTIONAL NEURAL NETWORKS AND U-NET

A Convolutional Neural Network consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a ReLU layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution.

Convolutional Neural Networks (CNN) prove to be the best while working with images, and especially 3D CNN models work better than 2D CNN models for biomedical images such as CT scans.

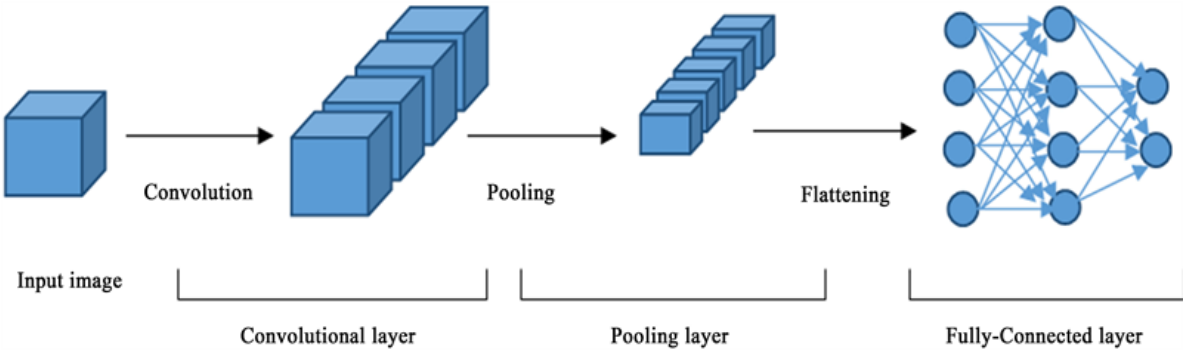

Fig. 1.1 Basic 3D CNN Architecture

U-Net is a convolutional neural network that was developed for biomedical image segmentation. The network consists of a contracting path and an expansive path, which gives it the u-shaped architecture.

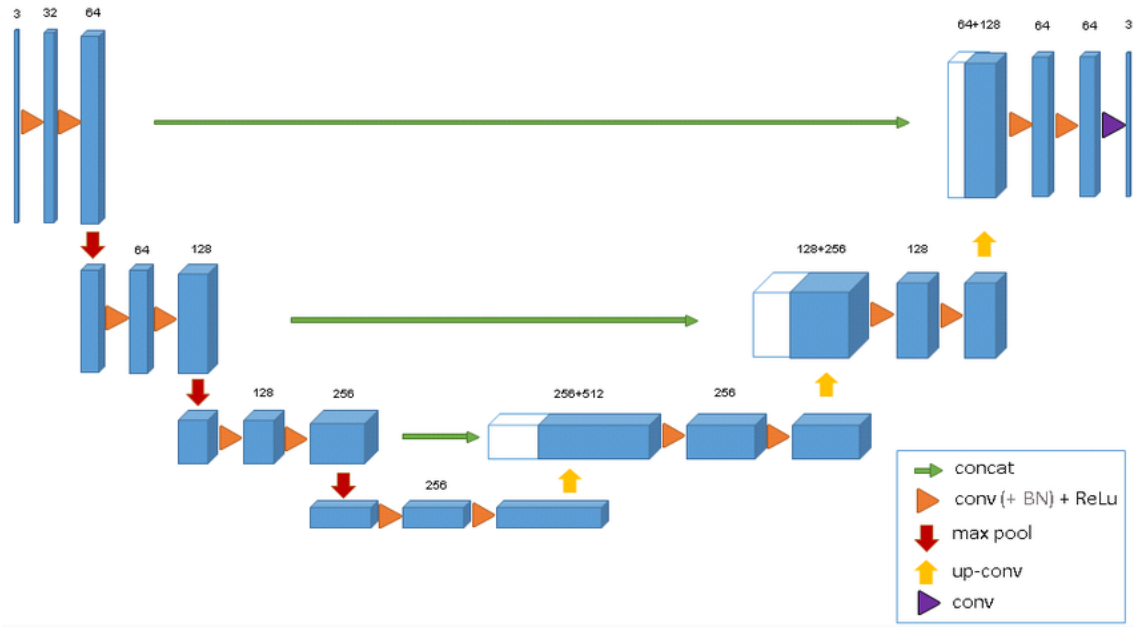

Fig. 1.2 Simple 3D U-Net Architecture

The contracting path is a typical convolutional network that consists of repeated application of convolutions, each followed by a rectified linear unit (ReLU) and a max pooling operation.

During the contraction, the spatial information is reduced while feature information is increased. The expansive pathway combines the feature and spatial information through a sequence of up-convolutions and concatenations with high-resolution features from the contracting path.

The idea is to use a 3D U-Net architecture for the purpose of segmenting the lung from the CT scans and 3D CNN model for classification and scoring of the severity of the CT scans.

1.2 PROBLEM STATEMENT

A GPU enabled CNN based COVID-19 HRCT Images Classifier that can detect COVID-19 infection from the given CT scan images of COVID and non-COVID patients and determine the severity of lung infection from those images.

This is done mainly using the four steps:

1. Preprocessing the data: Refers to all the transformations on the acquired data before it is fed to the deep learning algorithm. This step involves anonymizing patient details, verifying data distribution, and sampling the required images for each patient.

2. Segmentation: The preprocessed lung images will undergo segmentation to partition the lung scan into different segments. This is done to simplify the representation of an image to something more meaningful and easier to analyze.

3. Classification: The segmented images are taken for the process of classification to categorize all the pixels in a digital image into one of two classes.

4. **Scoring:** The images classified as COVID positive are assigned a severity score based on the amount of lung infected.

1.3 DATASET

The data set for this project is provided by PSG Institute of Medical Sciences & Research, Coimbatore. This data set contains a combination of Computed Tomography (CT) scan images of patients who are affected by COVID-19 and patients who are not affected by COVID-19.

A total of 210 patients CT scan images were given in which 125 patients are affected by COVID-19, and 85 patients are not affected by COVID-19.

Each of these 210 patients' CT scans contained nearly 400 to 800 slices of information.

The distribution of COVID-19 and non COVID-19 patients in the dataset is sufficient to train and test the model.

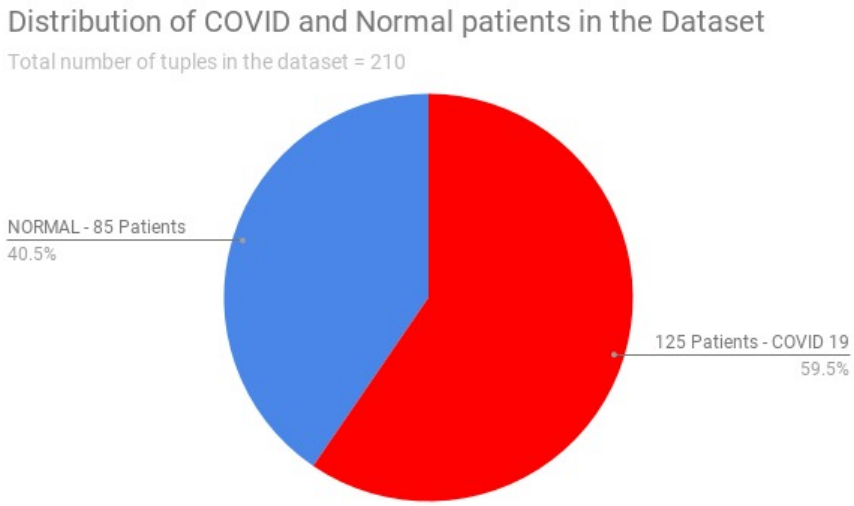

Fig.1.3. Distribution of COVID and Normal Patients in the Dataset

The severity of the infection in the lungs is measured by severity score, and this severity score is used to determine whether the patient is affected by COVID-19 or not.

The figure below represents the distribution of the patients' data according to the severity score.

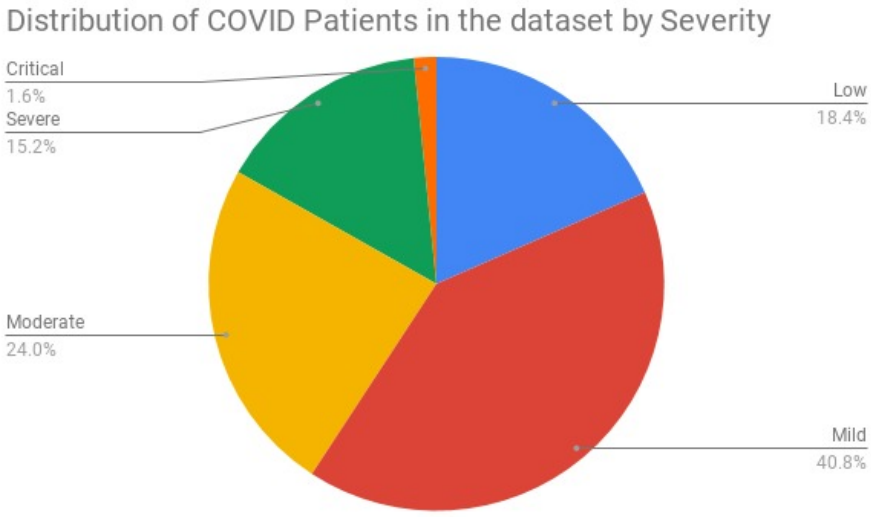

Fig.1.4. Distribution of COVID Patients in the Dataset by Severity

The CTSI score determines the severity of the infection and the score is directly proportional to the severity of the infection.

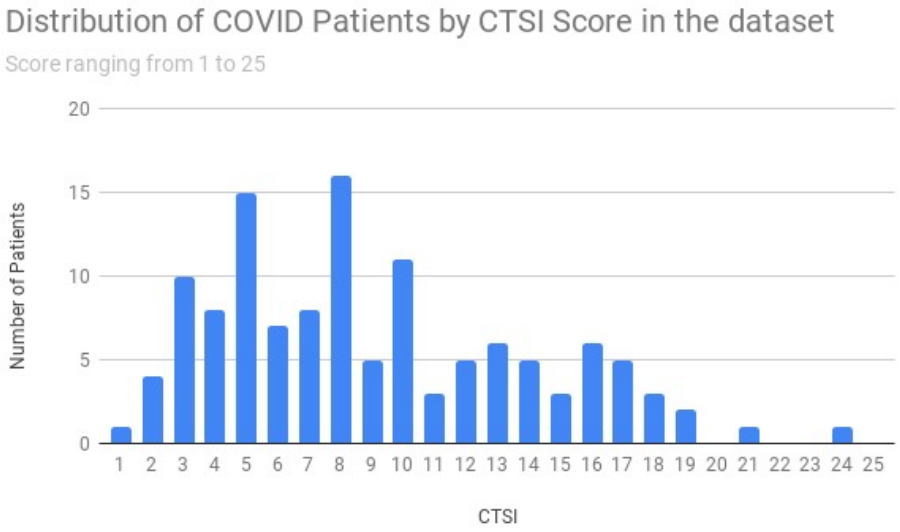

Fig.1.5. Distribution of COVID patients by CTSI Score in the dataset

The given dataset is split for training the model and for testing the model. 80 percent of the data is used for training and 20 percent data is used for testing.

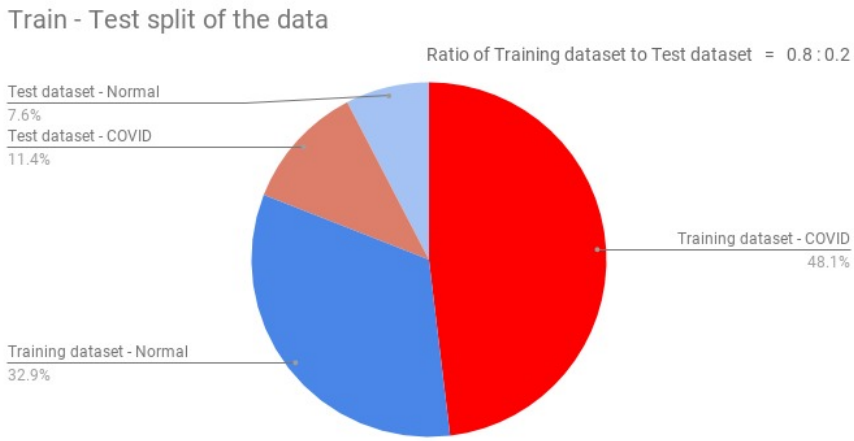

Fig.1.6. Dataset is split into training and test dataset

1.4 SCOPE AND APPLICATIONS OF THE PROJECT

The GPU enabled CNN based COVID-19 HRCT Images Classifier considers only DICOM images as input and other image formats are not supported. This model can only identify the infected lungs and cannot identify the cause of those infections. This model gives the severity scoring of lung infections.

Some applications are as follows

Testing for COVID-19 virus - As the swab tests used for testing Corona virus is not available all over the world and it also provides false negative and false positive results, this model can be used as an alternative by using the CT Scan images as input to the results which are more accurate and fast.

Testing for infections in the Lungs - This model can be used to find the infections in lungs with the help of the CT Scans. This model is marginally faster than the manual process of analyzing each and every CT Scan image by a doctor which is tedious work.

1.5 OBJECTIVES OF THE PROJECT

The objectives of the project are:

- Design and develop GPU enabled Convolutional Neural Network (CNN) based deep learning models to analyse Computed Tomography (CT) scans
- Use the developed 3D models to predict whether a patient is affected by COVID-19 or not using their CT scan and classify the severity of the infection.

1.6 ORGANIZATION OF THE REPORT

The report is organised as various chapters, each of which explains various parts of the project.

Chapter 1 explains the problem statement, motivation and general objectives of the project.

Chapter 2 deals with study of various similar papers that aim to perform lung segmentation and COVID-19 classification.

Chapter 3 describes the functional and non-functional requirements, hardware and software specifications required for the project.

Chapter 4 describes the various steps in the design and flow of the project.

Chapter 5 deals with the implementation of the project step by step. It describes the algorithm and the model used for classification and scoring.

Chapter 6 shows the results of segmentation, classification and scoring models and their accuracy in both training and validation dataset.

Chapter 7 is the conclusion which gives an overview of the various steps of the project.

CHAPTER 2

LITERATURE REVIEW

This chapter deals with the study of various research papers that aim to describe several approaches involved in lung segmentation, COVID-19 classification and comparison of famous architectures for COVID-19 classification.

2.1 TIME COURSE OF LUNG CHANGES ON CHEST CT DURING RECOVERY FROM COVID-19 PNEUMONIA

F. Pan, et al. described an approach which uses Computed Tomography Scans (CT) to assess the severity of lung involvement in coronavirus disease and its purpose is to determine the changes in the CT findings associated with COVID-19 from initial diagnosis until the recovery of the patient.

This retrospective review includes patients with real-time polymerase chain reaction and confirmed COVID-19 who presented between January 2020 and February 2020. This does not include patients with severe respiratory problems and oxygen requirements during the disease course. Repeated chest CT was performed approximately at an interval of 4 days.

Each of the five lung lobes were visually scored on a scale of 0 - 5, with 0 indicating no involvement and 5 indicating more than 75 percentage involvement..

In patients recovering from COVID-19, 4 stages of lung CT were identified where

- Stage 1
    - seen between day 0 to day 4 of the infection
    - shows ground glass opacities
- Stage 2
    - seen between day 5 to day 8
    - shows increased crazy paving pattern
- Stage 3
    - seen between day 9 to day 13
    - comes up with a consolidation
- Stage 4
    - seen after 14 days
    - has a gradual resolution of consolidation in most of the patients.

2.2 DEEP LEARNING WORKFLOW IN RADIOLOGY

Montagnon et al. introduced a checklist to know about the steps that are basically involved in deep learning projects especially in Radiology.

The article mainly illustrates the workflow of liver lesion detection and about its segmentation, classification, monitoring, and prediction of recurrence of tumor and the survival of the patient.

It also discusses about the challenges involved which includes ethical considerations and cohorting, collection of data, anonymization, and about the availability of expert annotations.

Key points:

- Deep learning helps with advanced performance for detection, segmentation, classification as well as prediction.
- The most time-consuming steps include data collection and curation.
- There are availabilities for many open-source deep learning frameworks including permissive licenses.
- Cloud computing basically influences third-party hardware, storage as well as technical resources.

This review thus provides a practical guide for radiologists interested in making projects involving deep learning in abdominal radiology. Specifically, the objectives of this article are to

1. Provide an overview of clinical use cases.
2. Describe the composition of a multidisciplinary team.
3. Summarizes current approaches to patient, data, model, and hardware selection.

2.3 PERFORMANCE AND SCALABILITY OF GPU-BASED CONVOLUTIONAL NEURAL NETWORKS

In this paper, D. Strigl et al describe about a framework for accelerating, training and classification model of CNN by utilizing the GPU and how it is implemented. The Multilayer Perceptron Neural Networks are used to derive CNN. These CNNs are optimised to solve 2D pattern recognition problems. The 2D pattern recognition problems include Optical Character Recognition, Face Recognition and Face Detection. These models define the basic parts of the Convolution Neural Network.

These models are used to test the performance and scalability and check whether there is any improvement which can be achieved by using GPU instead of CNN. And from the testing, the GPU is far more faster than CPU. Then, with respect to neural network size, GPU is much better than CPU.

The GPU implementation is better because the computations are parallel and it performs a large amount of floating point operations in lesser time.

Key Points:

- Performance and scalability tests were performed measurements on two different networks - *SimardNet* and *LeNet5*
- Training and classification using GPU implementation shows that training model takes 2 to 24 times lesser time in GPU than on the CPU.
- On the basis of scalability, GPU implementation scales much better than CPU implementation depending on the network topology.

2.4 A FULLY AUTOMATED DEEP LEARNING-BASED NETWORK FOR DETECTING COVID-19 FROM A NEW AND LARGE LUNG CT SCAN DATASET.

Mohammad R, Seyed M et al. introduced a fully-automated system to accurately identify COVID-19 from patients' CT scans. They added a new dataset that contained more than 48000 CT scan images from around 280 normal persons and more than 15000 images from about 95 patients with COVID-19 disease. First, they dropped CT images in which lungs are not sufficiently noticeable through an image processing algorithm. It helped to reduce the processing time and false detections.

Next, they increased the classification accuracy of convolutional networks using the ResNet50V2 network and a modified feature pyramid network alongside the newly designed architecture for classifying the selected CT images as having COVID-19 or not. Next, the system determines the condition of the patient using a selected threshold.

Their model achieved 98.49% accuracy on nearly 8000 test images in the first image classification stage. At the patient identification phase, the system accurately identified almost 95% of patients with high speed. To evaluate the model's classification correctness, they investigated the classified images with the Grad-CAM algorithm to show the area of infections in images.

Key Points:

Patient’s information is accessible via the DICOM files (16-bit grayscale), converting them to TIFF format, which holds the same 16-bit grayscale data does not conclude the patients’ private information.

2 stages:

- Stage 1:
    - Image processing to discard CT scans in which the infection is not visible or not useful depending on the count of dark pixels with a chosen threshold.
- Stage 2:
    - Feature pyramid network with deep convolutional neural network *ResNet50V2* to perform classification.

2.5 AUTOMATED ASSESSMENT OF CO-RADS AND CHEST CT SEVERITY SCORES IN PATIENTS WITH SUSPECTED COVID-19 USING ARTIFICIAL INTELLIGENCE.

This paper proposes a new attention model to develop and validate an AI system to score the severity of COVID-19 on lung CT images using CO-RADS and CT severity scoring

systems. CORADS-AI consists of three deep learning algorithms that segment the five pulmonary lobes, assign a CO-RADS score for the possibility of COVID-19 and assign a CT severity score for the degree of parenchymal involvement per lobe.

The CT scans were scored fully automatically using three successively applied deep learning algorithms.

These algorithms performed the following actions:

1. Pulmonary lobe segmentation, and labelling.
2. Lesion segmentation and CT severity score prediction.
3. CO-RADS score prediction.

This study also included patients who had received an unenriched lung CT scans due to clinical suspicion of COVID-19 at two different medical centers. The system was then trained, validated, and tested with data from one of the medical centers. Data from the second medical center was considered as an external test dataset.

Diagnostic performance and acceptance with scores assigned by eight independent observers were measured using receiver operating characteristic (ROC) analysis, linearly-weighted kappa and classification accuracy.

Important Results:

- CORADS-AI predicted scores from lung CT exams that were within one CO-RADS category in 81% of the patients and within one CT severity score point per lobe in 94% of the patients when compared to readings by eight independent human observers.
- CORADS-AI identified patients with COVID-19 from lung CT exams with an AUC of 0.95 in an internal cohort and with an AUC of 0.88 in an external cohort.

2.6 BENCHMARKING DEEP LEARNING MODELS AND AUTOMATED MODEL DESIGN

In this paper, Xin He et al. initially built a dataset called as Clean-CC-CCII which consisted of segmented CT scan images without noise.

This dataset consisted of more than 3 lakh slices of more than 2600 patients.

The dataset consisted of 3 classes

1. Novel Coronavirus Pneumonia (NCP)
2. Common Pneumonia (CP)

3. Normal

They aim to benchmark the performance of various famous 3D and 2D architectures of CNNs on the same classification problem. The comparison between 2D and 3D CNNs showed that 3D models learn better and produce much better results than 2D CNNs.

After hyper-tuning efforts, they conclude that the 3D CNN models DenseNet3D121 (the highest accuracy - 88.63%) and ResNet3D34 (best AUC - 0.959) performed the best. They also show that mixed-up data augmentation technique can improve the model's performance by 1-3% in terms of accuracy.

Finally, they propose a lightweight model named MNas3DNet41 which is a fully automated deep learning model for COVID-19 classification and achieved 87.14% accuracy with 87.25% F1-score and 0.957 AUC.

Key Points:

1. AutoML techniques were used to design DL models.
2. Compared two types of state-of-the-art (SOTA) DL models.
    a. 3D convolutional neural networks (CNNs), including DenseNet3D121, R2Plus1D, MC3 18, ResNeXt3D101, Pre- Act ResNet, and ResNet3D series.
    b. 2D CNNs including DenseNet121, DenseNet201, ResNet50, ResNet101 and ResNeXt101.
3. 3D CNN models perform much better than 2D CNN models.
4. The model performance does not scale very well with the model depth.
5. Increasing the number of slices does not necessarily improve model performance.

CHAPTER 3

REQUIREMENTS

After thorough analysis, the necessary requirements have been identified and are classified into functional and non-functional requirements. These have been explained in detail below.

3.1 FUNCTIONAL REQUIREMENTS

A functional requirement can be defined as a feature/function that must definitely be included in the system that is being designed. The functional requirements that thus system requires are given below:

1. System should be able to perform segmentation of the lung
2. System should be able to identify the presence of COVID-19 within the segmented portion of the lung
3. System should be able to assign a severity score based on the presence of COVID-19 in the lung.

3.2 NON-FUNCTIONAL REQUIREMENTS

Non- functional requirements are essentially based on performance, efficiency, accuracy, ease of use.

3.2.1 SOFTWARE QUALITY ATTRIBUTES

Availability : It will be available on all Hospital approved computers.

Correctness : The system should provide an accurate prediction of the presence of COVID-19 and assign the most relevant severity score.

Usability : The system should be able to handle all needs of the medical professional regarding detecting COVID-19 presence in the lung and score assigning.

Performance : System should perform efficiently.

Accuracy : The response time should be quick and reliable.

3.3 HARDWARE SPECIFICATIONS

RAM : 8 GB RAM or above

Processor : Intel i5 and above

Memory : 4 GB and above

Speed : 1.2 GHz and above

3.4 SOFTWARE SPECIFICATIONS

Operating System : Windows, MacOS, Linux

Platform used : Google Colab

Language used : Python

Libraries used : For models - Tensorflow, Keras

Other Python Libraries required are numpy, pydicom, matplotlib, scipy, h5py, os, SimpleITK, datetime, pytz.

CHAPTER 4

SYSTEM DESIGN

A total of 125 COVID-19 positive and 85 COVID-19 negative patients' images have been collected from the hospital, all these images are High-Resolution DICOM images (512x512 resolution) and are compiled as the dataset.

This chapter describes the various steps in the design and flow of the project.

4.1 FLOW DESIGN OF THE MODEL

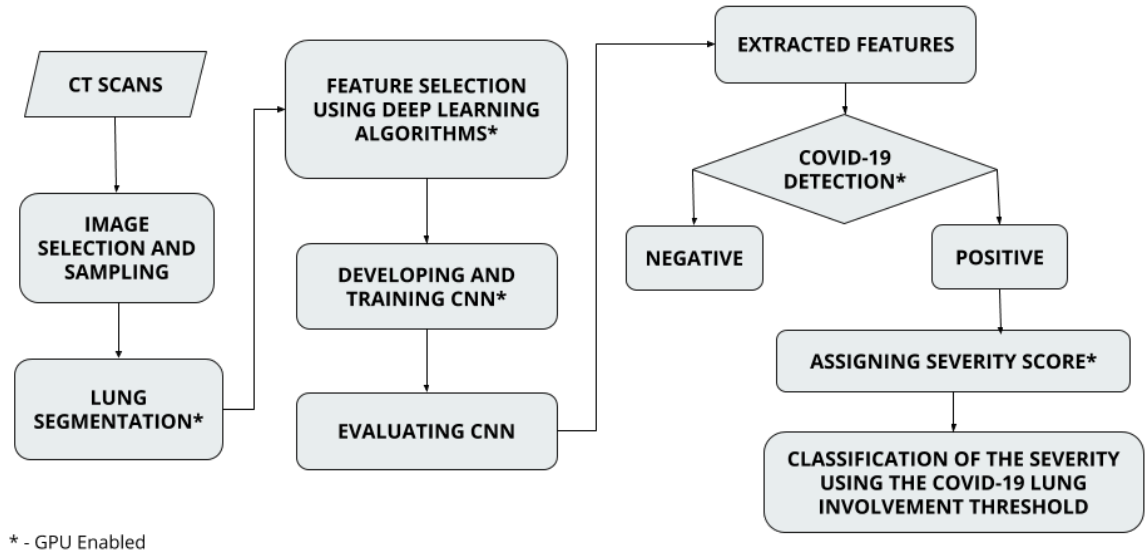

Fig.4.1. System Design - Flow Chart

Not all images in the dataset are valid images. When a patient inhales the lungs are open and those images are valid, but when the patient exhales, the lungs collapse making these images invalid.

Also, the images that are produced as the patient enters the CT scan and exits the CT scan are also invalid as these images don't contain the entire lung.

The lung portion is isolated from each image; this process is done by using a pre-trained model which segments the left and right lung from the background. The mask produced is mapped on to the original image. This process is speeded up by using GPU.

After selecting a particular model, the deep learning algorithms extract features by itself without any manual intervention.

The CNN model chosen is trained using the required outputs (scores for scoring model and labels for classification model) for the required number of epochs.

The chosen models are evaluated according to their accuracy or some other metrics and the best model is chosen.

Features extracted by deep learning algorithms are used to predict whether the patient is affected by COVID-19 or not.

Model outputs severity score 0 to 100 which is then used to classify the severity of the disease as Low, Mild, Moderate, Severe or Critical.

CHAPTER 5

SYSTEM IMPLEMENTATION

This chapter deals with the system implementation of the COVID-19 HRCT Image classification and scoring model. It describes the various steps in the implementation followed by the algorithm and the model used for classification and scoring.

5.1 PLATFORM FOR BACK-END

Colaboratory, or "Colab" allows developers to write and execute Python in the web browser without any configuration. Google Colab provides free GPU and 12 GB of RAM for running and executing Python files, which is extremely useful for training models. Python 3 language is used to run the code for the project.

5.2 LOADING DATA

The dataset consists of 210 patient samples (COVID-19 positive: 170, COVID-19 negative: 40) along with associated CT severity score, and the COVID-19 test result is fetched from Google drive.

5.3 DATA ANONYMIZATION

The received radiology images were in DICOM format with 512 * 512 pixels resolution. DICOM files of the patients contained patient information like Patient ID, etc. all of which were removed after anonymization using a professional DICOM editor called Sante DICOM Editor.

5.4 DATA VERIFICATION

We analyzed samples from the dataset and its data distribution. The given data contained more patients with less severity (score < 11) compared to those with more severity (score > 14) of the disease. Data verification was performed to avoid anomalies after anonymization using the CT scan reference number present in the DICOM headers.

5.5 SAMPLING

Each patient scan containing 350 to 800 slices is downsized to 64 slices. Each scan is resized across height, width, and depth to 64 * 128 * 128 shape to prevent overfitting and fit the model into the GPU provided by the Google Colab environment.

CT scans store raw voxel intensity in Hounsfield units (HU). To process the data, the following steps are carried out:

- We fix the depth as 64, height and width as 128.
- We resize the volume according to the new width, height, and depth.

5.6 SEGMENTATION

Sampled and resized images are segmented by applying a mask from U-Net R231 (Pre-trained 3D Model). Segmentation separates the left and right lung from the background. The segmented images are stored in the drive.

The U-net(R231) [1] model proves to perform very well because of the wide range of variability in its training dataset. The mask thus created using U-Net model is applied on the original CT scan images and the segmented dataset is created and saved.

5.7 BUILD TRAIN AND VALIDATION DATASET

The scans from respective class directories are read and assigned labels. The downsampled scans having a shape of 64 * 128 * 128 is split into train and validation subsets.

The dataset is randomly split into 170 samples in the training dataset and 40 samples in the test dataset while maintaining relative data distribution.

5.8 CLASSIFICATION MODEL

A 3D convolutional neural network (CNN) is defined for classification of CT Scans. It consists of several layers of convolution, max pooling, batch normalisation followed by dense layers at the end.

This model takes a 3D Volume of 64 slices, each of the dimension 128 * 128 as input.

Final activation is sigmoid function (since it is binary classification). Sigmoid function outputs a single value in the range of 0 to 1.

The model is trained with validation done at the end of each epoch with Binary Cross Entropy as loss function and Adam as optimizer function.

5.9 SCORING MODEL

A 3D convolutional neural network (CNN) is defined for predicting the severity score of CT Scans. It consists of several layers of convolution, max pooling, batch normalisation followed by dense layers at the end.

This model takes a 3D Volume of 64 slices, each of the dimension 128 * 128 as input.

Final activation is linear function (since it is similar to regression). The output is the score in the range of 0 to 100. The model is trained with validation done at the end of each epoch with Mean Squared Error as optimizer and loss function.

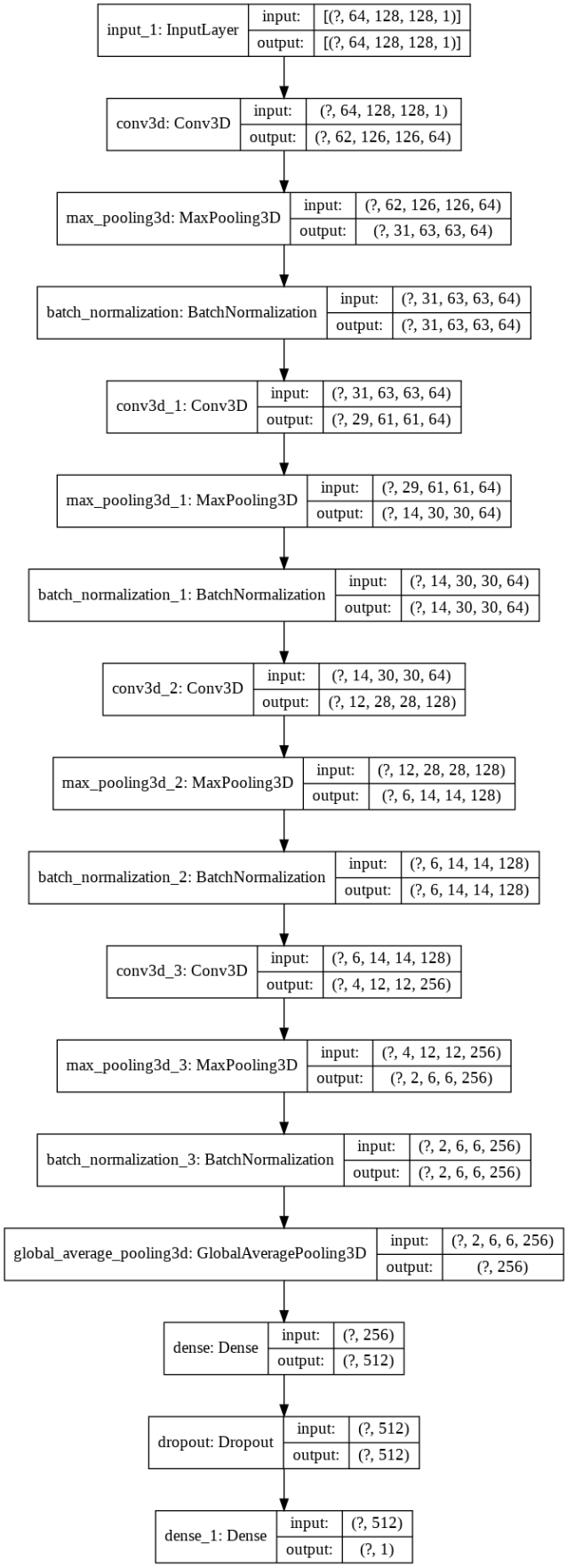

Fig 5.1 - 3D-CNN model for Classification and Scoring

The output of the scoring model is interpreted as follows:

Score	Severity Classification
0 to19	Low
20 to 39	Mild
40 to 59	Moderate
60 to 79	Severe
80 and above	Critical

Table 5.1 - Severity based on predicted score

5.10 ALGORITHM

Import all python modules required

Load the data and perform anonymization

Do {

```
def check_scans(start, end)
"""Returns the CT scan of Patients with ID"""
def get_patient_DCM_folder(id)
"""Returns path to the Patient's DCM folder"""

def get_non_patient_DCM_folder(id)
"""Returns path to Non patient's DCM folder"""

def load_scan(path)
"""Load all DICOM files in the path and return as array of slices"""

def get_pixels_hu(scans)
        '''Convert raw voxel values to HU and return image as numpy array'''

def resize_volume(img):
        """Resize across z-axis"""

def process_scan(path):
         """Read and resize volume"""

def apply_mask(img):
        """Returns original sampled images and segmented images after masking"""
```

```
def segmentation(patient_id_from, patient_id_to, COVID=True/False):
        """Perform segmentation for patients in the given range of IDs"""

def get_images(path):
        """Load image from .npy file in the given path"""

def store_many_hdf5(h5file, images, labels, scores):
        """Stores the images, labels, scores as dataset"""

def read_many_hdf5(h5file):
        """Loads dataset from given h5file"""

def get_model(width, height, depth):
        """Build a 3D convolutional neural network model for classification"""
# Train the model, doing validation at the end of each epoch

def get_scoring_model(width, height, depth, final_activation="linear"):
        """Build a 3D convolutional neural network model for scoring"""
# Train the model, doing validation at the end of each epoch

def get_severity_class(score):
  """Returns severity for the given score"""
} while true
```

CHAPTER 6

RESULTS

This chapter depicts the output of various steps that were performed for classifying and predicting the severity score from CT scans of the patients, followed by performance analysis of the models on training and validation datasets.

The preprocessed training dataset has 170 patient records and it consists of 10,880 slices. The preprocessed validation dataset has 40 patient records and it consists of 2,560 slices.

6.1 STEPS INVOLVED IN BUILDING THE MODEL

STEP 1: Anonymization

Input: The original patient scans in DICOM format along with DICOM header.

Output: Anonymised CT scans with patient details like name, age, etc., removed from the DICOM Header.

STEP 2: Segmentation to isolate lung portion using pre-trained U- net 231 model

Input: Anonymized patient scans with 600-700 slices in 512 x 512-pixel resolution

Output: Segmented scans with the shape 64 x 128 x 128

```
08-11-2020-12:24:53 Patient ID = 116
About to read DICOM at /content/drive/My Drive/covid/patient-116/
CT Scan read successful from /content/drive/My Drive/covid/patient-116/
<class 'numpy.ndarray'> (635, 512, 512)
Resizing successful
(64, 128, 128)
Downloading: "https://github.com/JoHof/lungmask/releases/download/v0.0/unet_r231-d5d2fc3d.pth" to /root/.cache/
100% 119M/119M [00:01<00:00, 66.7MB/s]

100%|| 4/4 [00:01<00:00,  2.78it/s]
100%|| 35/35 [00:00<00:00, 165.34it/s]
08-11-2020-12:32:49 Saving file at = /content/drive/My Drive/data_do_not_delete/dataset/covid/masked_116.npy
08-11-2020-12:32:50 Segmentation and saving successful for Patient 116
```

Fig 6.1. Segmentation Process

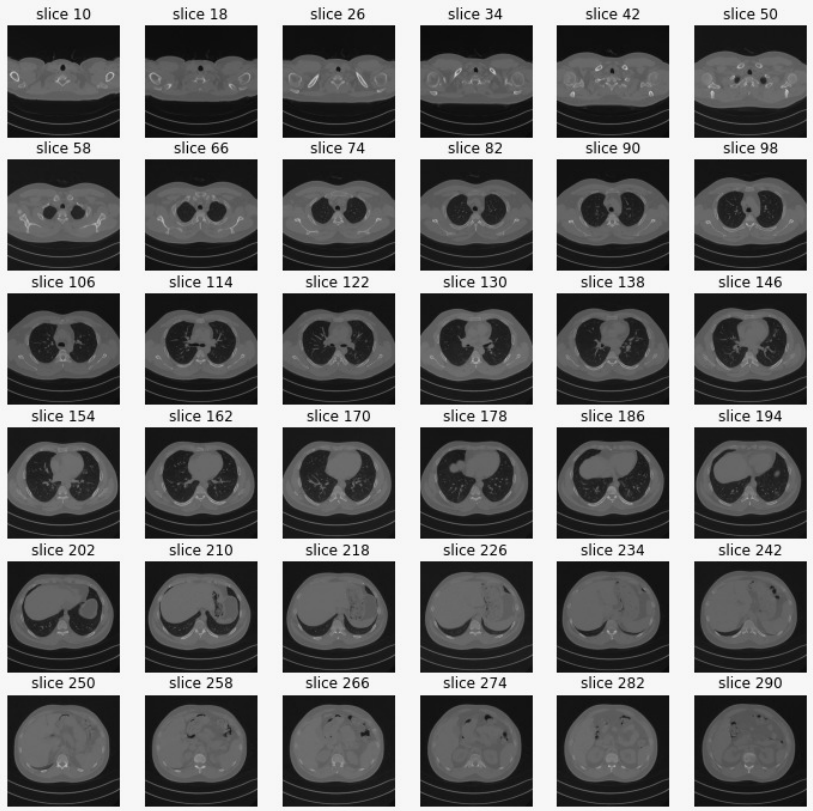

Fig.6.2. CT Slices before segmentation

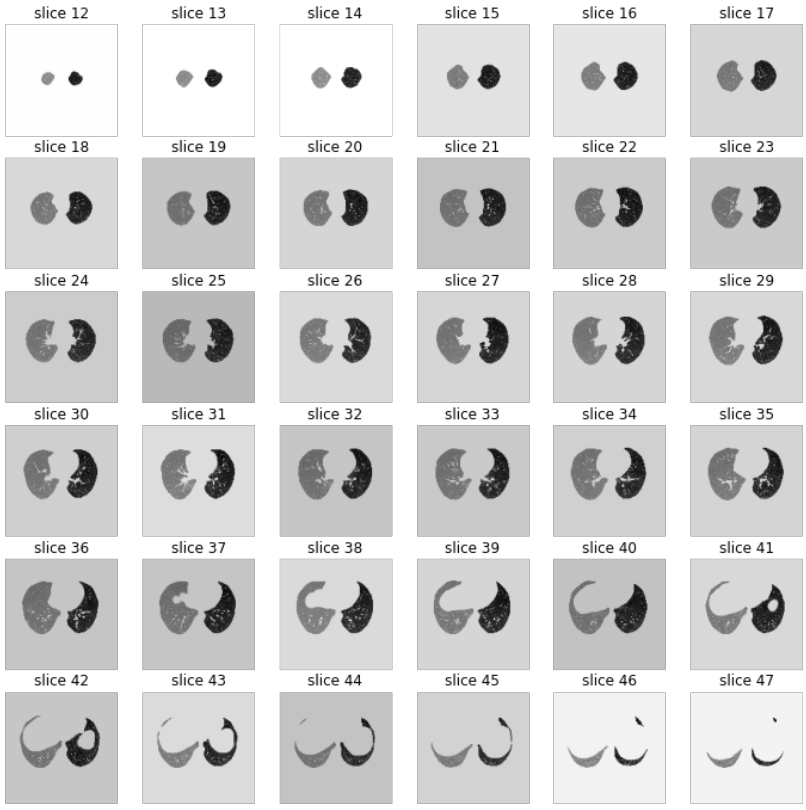

Fig.6.3. Segmented slices after applying mask

Step 3: Train 3D - CNN model to perform binary classification

Input: The original train dataset containing slices along with COVID test result (i.e positive-1, negative -0).

Output: Trained model after 18 epochs.

```
Epoch 1/100
WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time
85/85 - 14s - loss: 0.6710 - acc: 0.6412 - val_loss: 0.7463 - val_acc: 0.4500
Epoch 2/100
85/85 - 13s - loss: 0.6645 - acc: 0.5235 - val_loss: 0.5841 - val_acc: 0.6750
Epoch 3/100
85/85 - 14s - loss: 0.6432 - acc: 0.6059 - val_loss: 0.5154 - val_acc: 0.8000
Epoch 4/100
85/85 - 14s - loss: 0.6120 - acc: 0.6353 - val_loss: 0.6157 - val_acc: 0.5750
Epoch 5/100
85/85 - 14s - loss: 0.6057 - acc: 0.6882 - val_loss: 0.5143 - val_acc: 0.6250
Epoch 6/100
85/85 - 13s - loss: 0.5803 - acc: 0.6941 - val_loss: 0.8270 - val_acc: 0.6000
Epoch 7/100
85/85 - 14s - loss: 0.5641 - acc: 0.7059 - val_loss: 0.5046 - val_acc: 0.7250
Epoch 8/100
85/85 - 13s - loss: 0.5676 - acc: 0.7118 - val_loss: 0.7727 - val_acc: 0.5250
Epoch 9/100
85/85 - 14s - loss: 0.5312 - acc: 0.7235 - val_loss: 0.6290 - val_acc: 0.6250
Epoch 10/100
85/85 - 14s - loss: 0.5558 - acc: 0.6882 - val_loss: 0.4977 - val_acc: 0.6750
Epoch 11/100
85/85 - 14s - loss: 0.5195 - acc: 0.7412 - val_loss: 0.4738 - val_acc: 0.7500
Epoch 12/100
85/85 - 14s - loss: 0.4397 - acc: 0.7647 - val_loss: 0.4074 - val_acc: 0.7500
Epoch 13/100
85/85 - 13s - loss: 0.5045 - acc: 0.7588 - val_loss: 0.4661 - val_acc: 0.7000
Epoch 14/100
85/85 - 14s - loss: 0.4701 - acc: 0.7471 - val_loss: 0.5543 - val_acc: 0.7000
Epoch 15/100
85/85 - 14s - loss: 0.4249 - acc: 0.7647 - val_loss: 1.3606 - val_acc: 0.6000
Epoch 16/100
85/85 - 14s - loss: 0.4380 - acc: 0.7941 - val_loss: 0.5209 - val_acc: 0.6500
Epoch 17/100
85/85 - 14s - loss: 0.4253 - acc: 0.7882 - val_loss: 0.3824 - val_acc: 0.7750
Epoch 18/100
85/85 - 14s - loss: 0.4440 - acc: 0.7765 - val_loss: 0.5184 - val_acc: 0.7500
<tensorflow.python.keras.callbacks.History at 0x7f122e7e6ef0>
```

Fig.6.4. 3D CNN Classification Model Training of 18 epochs

Step 4: To score the severity of lung infection

Input: 64 x 128 x 128 scans which have COVID positive along with severity score

Output: Model trained to predict the severity score in the range of 1 to 100

```
Epoch 1/100
WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared to the batch time (
85/85 - 20s - loss: 913.5384 - mse: 913.5384 - val_loss: 994.6662 - val_mse: 994.6662
Epoch 2/100
85/85 - 20s - loss: 655.0367 - mse: 655.0367 - val_loss: 350.4662 - val_mse: 350.4662
Epoch 3/100
85/85 - 19s - loss: 505.2814 - mse: 505.2814 - val_loss: 497.1805 - val_mse: 497.1805
Epoch 4/100
85/85 - 19s - loss: 459.2404 - mse: 459.2404 - val_loss: 353.0202 - val_mse: 353.0202
Epoch 5/100
85/85 - 19s - loss: 349.4583 - mse: 349.4583 - val_loss: 563.1483 - val_mse: 563.1483
Epoch 6/100
85/85 - 19s - loss: 373.1845 - mse: 373.1845 - val_loss: 515.8281 - val_mse: 515.8281
Epoch 7/100
85/85 - 19s - loss: 357.4106 - mse: 357.4106 - val_loss: 455.1203 - val_mse: 455.1203
Epoch 8/100
85/85 - 19s - loss: 330.5908 - mse: 330.5908 - val_loss: 510.1369 - val_mse: 510.1369
Epoch 9/100
85/85 - 19s - loss: 299.9180 - mse: 299.9180 - val_loss: 1621.2931 - val_mse: 1621.2931
Epoch 10/100
85/85 - 19s - loss: 383.1299 - mse: 383.1299 - val_loss: 368.1598 - val_mse: 368.1598
Epoch 11/100
85/85 - 19s - loss: 264.8126 - mse: 264.8126 - val_loss: 443.3730 - val_mse: 443.3730
Epoch 12/100
85/85 - 19s - loss: 350.0805 - mse: 350.0805 - val_loss: 2032.7949 - val_mse: 2032.7949
Epoch 13/100
85/85 - 19s - loss: 269.6869 - mse: 269.6869 - val_loss: 1070.5349 - val_mse: 1070.5349
Epoch 14/100
85/85 - 19s - loss: 243.5412 - mse: 243.5412 - val_loss: 737.8983 - val_mse: 737.8983
Epoch 15/100
85/85 - 19s - loss: 228.4870 - mse: 228.4870 - val_loss: 723.9706 - val_mse: 723.9706
Epoch 16/100
```

Fig.6.5. 3D CNN Scoring Model Training of first 15 of 34 epochs

Step 5: Testing of severity score

Input: Single Patient's CT scan

Output: Predicted Severity Score Percentage

```
Affected Score Percentage = 43.03343 %
Predicted CTSI =  10.0

   Actual Score Percentage = 52 %
   Actual CTSI =  13
```

Fig.6.6. Test scan and severity score

6.2 PERFORMANCE ANALYSIS

The accuracy and loss for both classification and scoring for the training and the validation sets are plotted. Since the validation set is class-balanced, accuracy provides an unbiased representation of the model's performance.

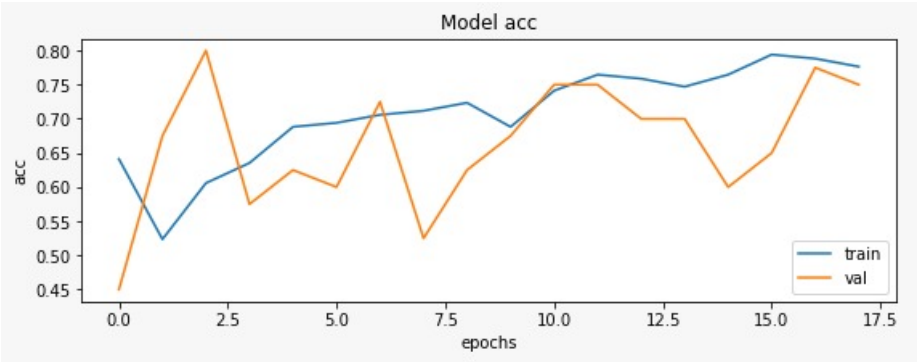

Fig.6.7. Classification Model Accuracy for Training and Validation dataset per epoch

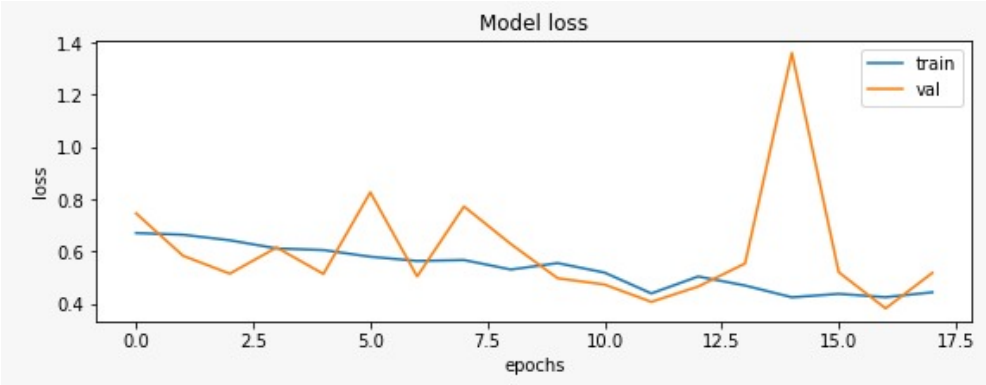

Fig.6.8. Classification Model Loss for Training and Validation dataset per epoch

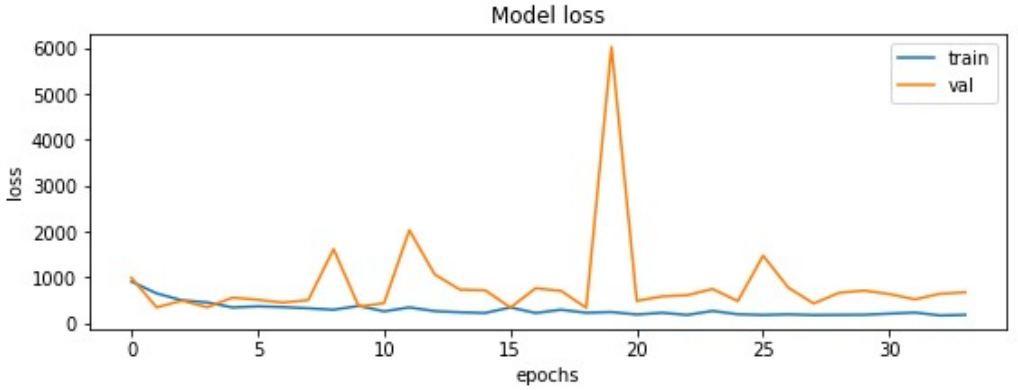

Fig.6.9. Scoring Model Loss for Training and Validation dataset per epoch

The following table shows the results of the 3D Scoring model for the test / validation dataset (which consists of a total of 40 samples) after converting the score (0 to 100) to 5 distinct classes - Low, Mild, Moderate, Severe and Critical.

Categorizing by Scoring	Number of samples	%
Predicted class **same** as actual class	23	57.5
Predicted class differs from the actual class by 1 class	13	32.5
Predicted class differs from the actual class by 2 classes	3	7.5
Predicted class differs from the actual class by 3 classes	1	2.5
Predicted class differs from the actual class by 4 classes	0	0

Table.6.1. Output of Scoring Model for Validation dataset

The table above shows that the number of exactly classified severity is 23 out of 40 samples. However, if we consider the samples where the prediction differs from the actual class by a single class (±1), the number of correctly classified samples will be 36 out of 40 samples(23 + 13 = 36).

Hence, the accuracy (±1) of the scoring model on the test / validation dataset is 90% (57.5 + 32.5 = 90).

The results of the models are tabulated as below.

Model	Training Accuracy	Test Accuracy
3D CNN for Classification	78.82%	77.50%
3D CNN for Scoring	96.47%	90%

Table.6.2. Accuracy of the 3D Models

CHAPTER 7

CONCLUSION

GPU enabled CNN based COVID-19 HRCT Images Classifier recognises the CT Scan images in the dataset and gives the severity scoring based on the lungs infection. This is done by using CNN (Convolution Neural Network), a class of deep neural networks which is used to analyze the visual imagery.

The advantage of using CNN is, it is a lot faster than manually analyzing each image and finding the results. The images in the dataset are raw images. These images are preprocessed before using. Then the preprocessed images are used for segmentation. In segmentation, the lung images in the CT scans are partitioned into segments so that the lungs can be viewed separately and clearly.

Then the segmented images are used for classification. In the classification model, the segmented images are classified into different classes for easy understanding and this classified data is used for building the scoring model. Then the model is tested with the testing dataset to check the accuracy. The platform used for building this model is Google Colab.

The future work which can be carried out in this project are,

- Finding the volume of the lungs affected
- Finding what type of infection from the data gathered from the lung images

Thus, this model can be used to know the severity score of the lung, and from the images, it can also classify between COVID affected patients and non-COVID affected patients.

BIBLIOGRAPHY

[1] Hofmanninger, J., Prayer, F., Pan, J. et al. Automatic lung segmentation in routine imaging is primarily a data diversity problem, not a methodology problem. Eur Radiol Exp 4, 50 (2020). https://doi.org/10.1186/s41747-020-00173-2

[2] Mohammad R, Seyed M et al. “A fully automated deep learning-based network for detecting COVID-19 from a new and large lung CT scan dataset”. medRxiv, 2020. https://doi.org/10.1101/2020.06.08.20121541

[3] Kang Zhang, Xiaohong Liu, Jun Shen, et. al., “Clinically Applicable AI System for Accurate Diagnosis, Quantitative Measurements, and Prognosis of COVID-19 Pneumonia Using Computed Tomography ”, Cell, 2020. https://doi.org/10.1016/j.cell.2020.04.045

[4] Lessmann, Nikolas & Sánchez et al. “Automated Assessment of CO-RADS and Chest CT Severity Scores in Patients with Suspected COVID-19 Using Artificial Intelligence”. Radiology. 202439. https://doi.org/10.1148/radiol.2020202439

[5] Fei N, Yaozong G, et al. “Lung Infection Quantification of COVID-19 in CT Images with Deep Learning.” Arxiv, 2020. https://arxiv.org/abs/2003.04655

[6] F. Pan, T. Ye, P. Sun, S. Gui, B. Liang, L. Li, et al., "Time course of lung changes on chest CT during recovery from 2019 novel coronavirus (COVID-19) pneumonia" Radiology, p. 200370, 2020. https://doi.org/10.1148/radiol.2020200370

[7] A. A. Ardakani, A. R. Kanafi, U. R. Acharya, N. Khadem, A. Mohammadi, “Application of deep learning technique to manage COVID-19 in routine clinical practice using CT images: Results of 10 convolutional neural networks”, Computers in Biology and Medicine, ScienceDirect, 2020. https://doi.org/10.1016/j.compbiomed.2020.103795

[8] Xin He, Shihao Wang et al., “Benchmarking Deep Learning Models and Automated Model Design for COVID-19 Detection with Chest CT Scans”, medRxiv, 2020. https://doi.org/10.1101/2020.06.08.20125963

[9] Montagnon, E., Cerny, M., Cadrin-Chênevert, A. et al., “Deep learning workflow in radiology: a primer”, Insights Imaging, 2020. https://doi.org/10.1186/s13244-019-0832-5

[10] D. Strigl, K. Kofler and S. Podlipnig, "Performance and Scalability of GPU-Based Convolutional Neural Networks," 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, Pisa, 2010. https://doi.org/10.1109/PDP.2010.43

[11] X. Li, G. Zhang, H. H. Huang, Z. Wang and W. Zheng, "Performance Analysis of GPU-Based Convolutional Neural Networks," 2016 45th International Conference on Parallel Processing (ICPP), Philadelphia, PA, 2016. https://doi.org/10.1109/ICPP.2016.15

[12] DICOM Processing and Segmentation in Python using Machine Learning method https://www.raddq.com/dicom-processing-segmentation-visualization-in-python/

[13] 3D Image Classification - https://keras.io/examples/vision/3D_image_classification/

[14] Zunair H., Rahman A., Mohammed N., Cohen J.P. (2020) Uniformizing Techniques to Process CT Scans with 3D CNNs for Tuberculosis Prediction. In: Rekik I., Adeli E., Park S.H., Valdés Hernández M..C. (eds) Predictive Intelligence in Medicine. PRIME 2020. Lecture Notes in Computer Science, vol 12329. Springer, Cham. https://doi.org/10.1007/978-3-030-59354-4_15

APPENDIX

```
# -*- coding: utf-8 -*-
"""3DCNN.ipynb"""

"""
# Installation and Custom Functions
"""

!pip install pydicom
!pip install git+https://github.com/JoHof/lungmask
import numpy as np
import os
import pydicom
import h5py
import pandas as pd
import matplotlib.pyplot as plt
from lungmask import mask
import SimpleITK as sitk
from datetime import datetime
import pytz
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from scipy import ndimage

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

covid_path = '/content/drive/My Drive/covid/'
normal_path = '/content/drive/My Drive/normal/'
covid_masked_path = '/content/drive/My Drive/data_do_not_delete/
dataset/covid/'
normal_masked_path = '/content/drive/My Drive/data_do_not_delete/
dataset/normal/'
masking_log_file_start_path = '/content/drive/My Drive/
data_do_not_delete/logs/mask_log/masking_'


def display_sample_stack(stack, rows=8, cols=8, start_with=0,
show_every=1, cmap='gray', size=12):
  """Plots the given np.ndarray as rows x cols stack of images"""
  fig,ax = plt.subplots(rows,cols,figsize=[size, size])
  for i in range(rows*cols):
      ind = start_with + i*show_every
```

```
      ax[int(i/rows),int(i % rows)].set_title('slice %d' % ind)
      ax[int(i/rows),int(i % rows)].imshow(stack[ind],cmap=cmap)
      ax[int(i/rows),int(i % rows)].axis('off')
  plt.show()

def get_time_stamp():
  """Returns current time as time stamp in IST"""
  IST = pytz.timezone('Asia/Kolkata')
  return datetime.now(IST).strftime("%d-%m-%Y-%H:%M:%S")


def log_and_print(file, content):
  """Prints to log file and console"""
  file.write("\n" + get_time_stamp() + "\t" + content)
  print(get_time_stamp(), content)

def check_scans(start, end):
  """Returns the CT scan of Patients with ID"""
  op = []
  for id in range(start, end+1):
    try:
      curr = get_patient_DCM_folder(id)
      filepath = os.listdir(curr)[0]
      dcmfile = pydicom.dcmread(curr + '/' + filepath)
      ctscan = dcmfile[(0x0040, 0x0275)][0][(0x0040,
0x0009)].value[:9]
      op.append([id, ctscan])
    except:
      pass
  return op

def get_patient_DCM_folder(id):
  """Returns path to the Patient's DCM folder"""
  return covid_path + 'patient-%02d/'%(id)


def get_non_patient_DCM_folder(id):
  """Returns path to Non patient's DCM folder"""
  return normal_path + 'npatient-%02d/'%(id)

def read_dicom_file(filepath):
  """Read and load volume"""
  img = load_scan(filepath)
  scan = get_pixels_hu(img)
  return scan
```

```
def resize_volume(img):
  """Resize across z-axis"""
  desired_depth = 64
  desired_width = 128
  desired_height = 128

  current_depth = img.shape[0]
  current_width = img.shape[1]
  current_height = img.shape[-1]
  depth = current_depth / desired_depth
  width = current_width / desired_width
  height = current_height / desired_height
  depth_factor = 1 / depth
  width_factor = 1 / width
  height_factor = 1 / height
  img1 = ndimage.zoom(img, (depth_factor, width_factor,
height_factor), order=1)
  return img1


def process_scan(path):
  """Read and resize volume"""
  # Read scan
  print('About to read DICOM at', path)
  volume = read_dicom_file(path)
  print('CT Scan read successful from', path)
  print(type(volume), volume.shape)

  # Resize width, height and depth
  volume = resize_volume(volume)
  print('Resizing successful')
  print(volume.shape)
  plt.imshow(volume[30], cmap="gray")
  return volume


def apply_mask(img):
  """Returns original sampled images and segmented images after
masking"""
  # The original resized image
  orig = sitk.GetImageFromArray(img, isVector=False)
  # Apply Mask
  seg = mask.apply(orig)
  return img, np.multiply(img, seg)
```

```
def segmentation(patient_id_from, patient_id_to=None, covid=True):
  """Perform segmentation for patients in the given range of IDs"""
  patient_id_to = patient_id_from if patient_id_to is None else
patient_id_to
  mask_log = masking_log_file_start_path + get_time_stamp() + '.txt'
  log = open(mask_log, "a")

  for patient_id in range(patient_id_from, patient_id_to + 1):
    log_and_print(log, ("" if covid else "Non ") + "Patient ID = " +
str(patient_id))
    # Segmentation and mask
    path = get_patient_DCM_folder(patient_id) if covid else
get_non_patient_DCM_folder(patient_id)
    img = process_scan(path)
    orig_np, seg_np = apply_mask(img)
    # Save
    save_path = (covid_masked_path if covid else (normal_masked_path
+ "n")) + "masked_%02d.npy" % (patient_id)
    log_and_print(log, "Saving file at = " + save_path)
    np.save(save_path, seg_np)
    log_and_print(log, "Segmentation and saving successful for " +
("" if covid else "Non ") + "Patient " + str(patient_id) + "\n\n")

  # After Segmentation of last patient
  display_sample_stack(np.load((covid_masked_path if covid else
(normal_masked_path + "n")) + "masked_%02d.npy"%(patient_id_to)))
  log.close()


def get_images(path):
  """Load image from .npy file in the given path"""
  location = '/content/drive/My Drive/data_do_not_delete/dataset/' +
path
  print(get_time_stamp())
  print('Loading', location)
  img = np.load(location)
  print(img.shape)
  return np.array(img)


def store_many_hdf5(h5file, images, labels, scores):
  """Stores the images, labels, scores as dataset"""
  num_images = len(images)
  # Create a dataset in the file
  dataset = h5file.create_dataset(
```

```
      "images", np.shape(images), h5py.h5t.STD_I32BE, data=images
  )
  meta_set = h5file.create_dataset(
      "labels", np.shape(labels), h5py.h5t.STD_U8BE, data=labels
  )
  scores_set = h5file.create_dataset(
      "scores", np.shape(scores), h5py.h5t.STD_U8BE, data=scores
  )


def read_many_hdf5(h5file):
  """Loads dataset from given h5file"""
  images = np.array(h5file["images"]).astype("int32")
  labels = np.array(h5file["labels"]).astype("uint8")
  scores = np.array(h5file["scores"]).astype("uint8")
  return images, labels, scores

"""# Segmentation (Do not execute again)"""

# segmentation(1, 125, covid=True)

# segmentation(1, 85, covid=False)

"""# Dataset Preprocessing"""

df = pd.read_csv('/content/drive/My Drive/data_do_not_delete/
data.csv')
temp_x = df['path'].values
temp_y = df['covid'].values
temp_scores = df['score'].values

# Split data 210 => 170 (train) + 40 (test)
split = 170
x_train = np.array([get_images(path) for path in temp_x[:split]])
y_train = np.array(temp_y[:split])
scores_train = np.array(temp_scores[:split])

print(x_train.shape, y_train.shape, scores_train.shape)

# split = 170
x_val = np.array([get_images(path) for path in temp_x[split:]])
y_val = np.array(temp_y[split:])
scores_val = np.array(temp_scores[split:])

print(x_val.shape, y_val.shape, scores_val.shape)
```

```
# Commented out IPython magic to ensure Python compatibility.
print(
    "Number of samples in train and validation are %d and %d."
#     % (x_train.shape[0], x_val.shape[0])
)

x_train = x_train.reshape(len(x_train), 64, 128, 128, 1)
x_val = x_val.reshape(len(x_val), 64, 128, 128, 1)
y_train = y_train.reshape(len(y_train), 1)
y_val = y_val.reshape(len(y_val), 1)
scores_train = scores_train.reshape(len(scores_train), 1)
scores_val = scores_val.reshape(len(scores_val), 1)
print(x_train.shape, y_train.shape, scores_train.shape)
print(x_val.shape, y_val.shape, scores_val.shape)

# train = h5py.File('/content/drive/My Drive/data_do_not_delete/
dataset/training.h5', 'w')
# store_many_hdf5(train, x_train, y_train, scores_train)
# train.close()

# vali = h5py.File('/content/drive/My Drive/data_do_not_delete/
dataset/validation.h5', 'w')
# store_many_hdf5(vali, x_val, y_val, scores_val)
# vali.close()

"""# Load Dataset"""

train = h5py.File('/content/drive/My Drive/data_do_not_delete/
dataset/training.h5', 'r')
x_train, y_train, scores_train = read_many_hdf5(train)
train.close()

print(x_train.shape, y_train.shape, scores_train.shape)

vali = h5py.File('/content/drive/My Drive/data_do_not_delete/
dataset/validation.h5', 'r')
x_val, y_val, scores_val = read_many_hdf5(vali)
vali.close()

print(x_val.shape, y_val.shape, scores_val.shape)

"""# Classification """

"""## Model 1 - 3D CNN"""

model = get_model(width=128, height=128, depth=64)
```

```
model.summary()

dot_img_file = 'model_basic.png'
tf.keras.utils.plot_model(model, to_file=dot_img_file,
show_shapes=True)

results = model.evaluate(x_val, y_val, batch_size=10)
model.load_weights(“3d_image_classification_weights.h5")
prediction = model.predict(x_val)
np.column_stack((prediction * 100, y_val))

"""# Scoring """

"""## Model 3 - 3D CNN for Scoring (with Linear Activation)"""

scoring_model_linear = get_scoring_model(width=128, height=128,
depth=64, final_activation="linear")
scoring_model_linear.summary()

scoring_model_img_linear = '/content/drive/My Drive/
data_do_not_delete/dataset/scoring_model_linear.png'
tf.keras.utils.plot_model(scoring_model_linear,
to_file=scoring_model_img_linear, show_shapes=True)

"""## 3D CNN Linear - Prediction""

scoring_model_linear.load_weights("/content/drive/My Drive/
data_do_not_delete/dataset/scoring_linear_3d.h5")
prediction = scoring_model_linear.predict(np.expand_dims(x_val[0],
axis=0))[0]
print("Affected Score Percentage =", prediction[0], "%")
print("Predicted CTSI = ", prediction[0]//4)

print("Actual Score Percentage =", scores_val[0][0], "%")
print("Actual CTSI = ", scores_val[0][0]//4)

def get_severity_class(score):
  """Returns severity for the given score"""
  if 0 <= score < 20:
    return 'Low'
  if 20 <= score < 40:
    return 'Mild'
  if 40 <= score < 60:
    return 'Moderate'
  if 60 <= score < 80:
    return 'Severe'
```

```
  if score >= 80:
    return 'Critical'

pred_score = scoring_model_linear.predict(x_val, batch_size=5)
pred_class = np.reshape(np.array([get_severity_class(score) for
score in pred_score]), (len(pred_score), 1))
actual_class = np.reshape(np.array([get_severity_class(score) for
score in scores_val]), (len(pred_score), 1))

array = np.concatenate((pred_class, actual_class), axis = 1)
data_frame = pd.DataFrame(data = array,
                  columns = ['Predicted Severity', 'Actual
Severity'])
data_frame.to_csv('predictions.csv')
```