

Multi-Scale Attention based CNN algorithm for Covid-19 classification

submitted by

Aritra Adhya

Registration No : **163596** of 2022-2023

Class Roll No : **002210503004**

Examination Roll No : **MCA244004**

2nd Year 2nd Semester

Master of Computer Application (MCA)

under the guidance & supervision of

Assistant Prof. Anasua Sarkar

in the

Department of **Computer Science and Engineering**

Faculty of Engineering and Technology

Jadavpur University

2024



Faculty of Engineering and Technology **Jadavpur University**

CERTIFICATE OF RECOMMENDATION

To Whom It May Concern

I hereby recommend that the project report entitled “*Multi-Scale Attention based CNN Algorithm for Covid-19 classification*” has been carried out by Aritra Adhya (Reg. No.: 163596 of 2022-23, Class Roll No.: 002210503004, Examination Roll No.: MCA244004), under my guidance and supervision and be accepted in partial fulfillment of the requirement for the degree of Master of COMPUTER APPLICATION (MCA) in the Department of COMPUTER SCIENCE and ENGINEERING, JADAVPUR UNIVERSITY during the academic year 2023-2024.

Assistant Prof. Anasua Sarkar

Project Supervisor

Department of Computer Science and Engineering
Jadavpur University

Prof. (Dr.) Nandini Mukhopadhyay

Head of the Department (HoD)

Department of Computer Science and Engineering
Jadavpur University

Prof. Dipak Laha

Dean, Faculty Council of Engineering & Technology
Jadavpur University, Kolkata-32



Faculty of Engineering and Technology **Jadavpur University**

CERTIFICATE OF APPROVAL

This is to certify that the project report entitled “*Multi-Scale Attention based CNN Algorithm for Covid-19 classification*” is a bonafide record of work carried out by ARITRA ADHYA, in fulfillment of the requirements for the award of the degree of Master of Computer Application (MCA) in the Department of Computer Science and Engineering, Jadavpur University. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein but approve the project report only for the purpose for which it has been submitted.

Signature of *Examiner 1*
Date :

Signature of *Examiner 2*
Date :



Faculty of Engineering and Technology **Jadavpur University**

DECLARATION OF COMPLIANCE WITH ACADEMICS ETHICS

I hereby declare that this project report entitled “*Multi-Scale Attention based CNN Algorithm for Covid-19 classification*” is an outcome of my genuine and independent work, as part of my Master of Computer Application program.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, all materials and results that are not original to this work have been fully cited and referenced.

Aritra Adhya

Registration No.: 163596 of 2022-23

Examination Roll No.: MCA244004

Project Title : *Multi-Scale Attention based CNN Algorithm for Covid-19 classification*

Student's Signature with Date

ACKNOWLEDGEMENT

With my most sincere gratitude, I would like to thank *Assistant Prof. Anasua Sarkar* (Department of Computer Science & Engineering, Jadavpur University), my supervisor, for her overwhelming support throughout the duration of the project. Her motivation always gave me the required inputs and momentum to continue with my work, without which the project work would not have taken its current shape. Her valuable suggestions and numerous discussions have always inspired new ways of thinking. I feel deeply honored that I got this opportunity to work under her.

I would like to express my sincere thanks to all my teachers for providing a sound knowledge base and cooperation.

I would like to thank all the faculty members of the Department of Computer Science & Engineering, Jadavpur University for their continuous support.

Last, but not the least, I would like to thank my batch mates for staying by my side when I needed them.

Aritra Adhya

Registration No : 163596 of 2022-2023

Class Roll No : 002210503004

Examination Roll No : MCA244004

Department of *Computer Science and Engineering*
Jadavpur University

Contents

● Acknowledgement	i
● Abstract	1
● Introduction	2
● Methodology	6
● Experimental Results & Analysis	16
● Conclusion	22
● References	25

Abstract

Coronavirus 2019 (COVID-19) is a new acute respiratory disease that has spread rapidly throughout the world. The main purpose of this project is to investigate and compare several different deep learning enhanced neural network techniques applied to CT-scan medical images for the detection of COVID-19. In this project, we used some powerful neural network models with deep learning enhancement, such as classical CNN, RNN, LSTM, Multi-scale CNN and Multi-scale CNN with Attention mechanism, for the COVID-19 CT-scan binary classification task. The proposed project framework is designed to find out the best architecture, pre-processing, and training parameters for the models largely automatically. The accuracy and F1-score were both above 98% and 91% respectively, in the diagnosis of COVID-19 using CT-scan images. In addition, we applied augmentation techniques to overcome the insufficient data and to improve the training time. The accuracy and validity of the algorithms were assessed on CT-scan images of well-known public datasets. The proposed method has better results for COVID-19 diagnosis than others related in literature. In our opinion, our work can help virologists and radiologists to make a better and faster diagnosis in the struggle against the outbreak of COVID-19.

This project proposes a lightweight or novel convolutional neural network (CNN) model named Multi-Scale Attention based CNN (AMS-CNN) is proposed, which is based on attention mechanism and multi-scale convolution. A multi-scale attention mechanism is designed to extract effective feature information from the COVID-19 CT images for classification. Moreover, the multiscale convolution layers are adopted as AMS-CNN's backbone to reduce the model size and parameters. The training, verification/ validation and testing are carried out on an interesting CT scan images dataset in Kaggle named '*sarscov2-ctscan-dataset*'. The AMSCNN model is compared with classical CNN, RNN, LSTM and Multi-scale CNN models. The experimental results show that the MS-CNN model with three multi-scale convolution layers with Attention mechanism (AMSCNN) has achieved accuracy of 98.02% on CT scan images dataset, which is higher than the other four models. The specificity and sensitivity are 90.71% and 92.18% on CT scan images dataset. The size of the AMSCNN model is only 57.28 MB, and in addition, the detection speed of the model on CT scan images is 4.23 ms per image, respectively. It is proved that our AMSCNN model can detect and classify COVID-19 faster with higher accuracy and efficiency.

Introduction

Coronavirus 2019 (COVID-19) is a persistent severe acute respiratory pandemic caused by a severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) infection. On 30 January 2020, the epidemic was declared an international public health emergency (PHEIC) by the World Health Organization (WHO). On 11 February 2020, the novel coronavirus disease was named 'COVID-19', and it was recognized as a pandemic on 11 March. Since the discovery of the first case, the disease has spread to almost every country; As of the statistics from the WHO by 28 June 2021, more than 180 million confirmed cases of COVID-19 have been reported globally since the outbreak began, with more than 3.9 million confirmed deaths.

The clinical symptoms of patients infected with COVID-19 are similar to other viral upper respiratory diseases. According to clinical observations, most COVID-19 cases have common features on X-ray and CT images, such as multiple small patchy shadows, interstitial changes, and glassy turbidity. Further symptoms developed into multiple ground-glass and infiltration shadows in both lungs. The first step in the treatment of COVID-19 is to screen patients in primary health centers or hospitals. Although the final diagnosis still relies mainly on transcription-polymerase chain reaction (PCR) tests. In case of people with strong respiratory symptoms the election protocol nowadays in hospitals relies on the medical imaging. In these severe cases, lung consolidation and pleural effusion may occur on the X-ray and CT images. In recent years, medical images analysis has been widely used in the diagnosis field due to its non-invasive and fast. Traditional manual diagnosis methods are time-consuming and laborious, and each doctor may have different diagnostic principles, resulting in the diversity of diagnosis results. Therefore, automatic classification of coronavirus ID-19 lesions in clinical Settings is quite necessary, which is the motivation of this study. Thus medical imaging is helping doctors to identify diseases and their effects more quickly. Following this protocol, patients that are suspected to suffer COVID-19 undergo first an X-Ray session and then, in the case that more details are needed, they take a CT-scan session. As a result of this protocol, computed tomography scan (CT scan) and X-ray images are being widely used in the clinic as alternative diagnostic tools for detecting COVID-19 and to find the effects of the virus.

Recent studies have shown that COVID-19 can be quickly and effectively diagnosed by observing the relevant features of lung CT/X-ray scan images. Due to that chest X-ray images and CT images can display the features of respiratory diseases, they become the important tools for rapid identification and early diagnosis of COVID-19. Therefore, to make the diagnosis, doctors depend on to visualize the lungs on X-ray or CT-scan images and search for symptoms of COVID-19 deformation. But, the high transmission rate of COVID-19 has resulted in a large influx of patients into hospitals in a short period of time, placing a significant burden on imaging physicians and often resulting in doctors' shortages

of the fight against the disease. This problem can be solved by using deep learning methods, which have continued to make significant progress in recent years, mainly due to the increasing computing power and the continuously growing amount of available data, as well as the continuous improvement of deep learning models and their algorithms, as demonstrated in challenge competitions to achieve record-breaking performances. The essence of deep learning is to learn more accurate features by building a multi-hidden layer machine learning model that is trained with a large amount of sample data to eventually improve the accuracy of classification or prediction. The related algorithms based on deep learning are recognized as the most effective approach to implement image classification quantitatively and qualitatively with advantages of the workload reduction and misdiagnosis decrease by manual diagnosis. With the continuous development of computer-aided diagnosis (CAD) technology, the detection accuracy and speed have become essential and significant for clinical application. A great deal of work has been paid to learning in medical images analysis. However, the analysis still requires expertise and includes a variety of algorithms to enhance, speed up and render an accurate diagnosis. Deep learning algorithms have achieved better efficiency in pneumonia detection and demonstrated high precision compared to previous approaches. However, currently, in hospitals doctors may find patients with pneumonia caused by flu, other viruses, and COVID-19 at the same time. Thus, there is a need for a fast and accurate method of detection that can identify both kinds of pneumonia. Therefore, advancing deep learning to detect and diagnose the lung medical images of patients with new COVID-19 pneumonia is needed.

In image processing, the feature extraction task is essential to reduce the redundant input information. Specifically, for COVID-19 detection, the extraction of efficient features is highly crucial. The algorithm may overfit and poorly generalize to new samples due to the poor features. The conventional methods are not able to extract robust features from the complex data like SARS-CoV-2s. Recently, automatic feature extraction using deep neural networks is becoming very popular. Moreover, the attention mechanisms are currently the essential building block of most state-of-the-art architectures, that accommodate more complex datasets and more flexible modeling representation. To help the model focus on important representation for SARS-CoV-2 classification, we exploit the idea from the attention mechanism. The basic idea of the attention mechanism is similar to human perception. For instance, in human perception, attention plays a vital role, enabling humans to pay attention to an essential portion of the picture, rather than proceeding with the complete image in its entirety. Due to its significance, researchers proposed attention approaches in the DL field to enhance CNN's performance in image segmentation and image classification tasks. For instance, Jie et al. developed a solid squeeze and excitation module to exploit the relationships between channels. They employed global average pooling to pool features and attain channel descriptors. Further, they used two fully connected layers to detect the relationships between channels.

The attention approach can be classified into spatial-wise and channel-wise attention modules. In a channel-wise approach, the attention module utilized an inter-channel

relationship with extra convolutional layers, which denotes the correlation between the key information and the current channel. With the larger weights, we can pay more attention to the channel. On the other hand, in a spatial-wise mechanism, the attention module detects prominent features by using the inter-spatial relationship from various locations of feature maps. Max et al. presented a spatial transformer network to convert the feature map within the network spatially. Moreover, some researchers have concatenated both channel and attention approaches to take synergetic effects. The existing literature shows that DL models obtained good results in processing medical images. However, the current research rarely considers the role of the attention mechanism for COVID-19 classification. It is thus unable to capture the spatial-wise and channel-wise relationship in a variety of scopes.

Therefore, in this work, we employed multi-scale attention to choose prominent features adaptively by accommodating different feature weights in the diverse scale domain. In contrast to the previous studies that were focused mainly on the spatial attention having numerous weight parameters or channel attention with fewer parameters, this work employs the attention mechanism that focuses on multi-scale attention for COVID-19 classification.

Since 2020, in order to combat COVID-19, lots of experts and researchers have employed artificial intelligence techniques to seek for the best or the fittest model to recognize COVID-19 accurately and efficiently from accessible X-ray and CT images. On this Background, Many deep learning methods have been used to diagnose COVID-19. Besides X-ray images, some researchers used CT images to classify COVID-19. The medical image classification method based on CNN has achieved good results. COVID-19 pneumonia detection approaches based on deep learning have been described by several groups. Alshazly used deep CNN architectures on CT-scan images to detect COVID-19 with accuracy, precision, and sensitivity of 93.96%, 99.13% and 94% respectively. Ayrton presented ResNet50 based deep transfer learning technique and reported the validation accuracy of 96.2% with a small dataset of 339 images for training and testing. Singh et al. designed a CNN model based on multi-objective differential evolution, which was optimized by the multi-objective fitness function. For binary classification on chest CT images, the accuracy of the model achieved 93.4%. Wang et al. proposed a lightweight residual projection-expansion-projection extension (PEPX) architecture named COVID-Net. The accuracy of the three classification tasks (COVID-19, normal and pneumonia) is 92.4% , in the four-category task (COVID-19, viral pneumonia, bacterial pneumonia, and normal) is 83.5%. Wang Proposed five pre-trained deep learning models, which the Xception model showed a relatively ideal effect, and the accuracy reached 96.75%. Li et al. trained a COVNet model based on the ResNet, and performed a binary classification experiment on 4356 chest CT images. The result showed a sensitivity of 90%. Zheng et al. designed a DeCovNet model based on a weak supervision method. The DeCovNet model used pre-trained U-Net to segment the lung area, and feed the segmented 3D lung area into a 3D deep neural network. The binary classification experiment was performed on 499 CT images. The model had an accuracy of 90.1% and a sensitivity of 90.7%. Song et al. designed a Details Relation Extraction neural model (DRE-Net), which is based on the pre-trained ResNet50 and added the Feature Pyramid Network (FPN), to extract the top-K details in the CT images and obtain

the image-level predictions. The DRE-Net model performed a binary classification experiment (COVID-19 and bacterial pneumonia) on 1485 CT images. The accuracy of the model achieved 94.0%. Oulefki et al. proposed a COVID-19 segmentation method, which enhanced image contrast by combining linear and logarithmic splices parameter, and used an image segmentation method to minimize the over-segmentation regions to segment CT tomography images. The method has strong robustness and simplicity with accuracy of 98%. At the same time, Oulefki et al. proposed a novel 3D visualization segmentation technique based on virtual reality, which has achieved good results in the recognition, measurement and analysis of COVID-19. Pathak et al. used transfer learning to classify COVID-19. The cost-sensitive top-2 smooth loss function is used to eliminate noise and unbalance of dataset categories. Experimental results show that this method has achieved a remarkable classification effect. The above medical image classification method based on CNN mainly uses local spatial information and ignores “global” indication, resulting in sub-optimal performance classification.

According to above literature investigations, it is clear that the accuracy of classification can continue to improve with complex structures and deeper layers. But complex structures and deeper layers will cause excessive parameters and large size of the model, which can reduce the detection speed. In practice, it is hoped to get the diagnosis result as soon as possible, so a lightweight CNN model is more practical and valuable in the clinic. Inspired by the ideas of Xception and Attention, this project designs a novel lightweight CNN model using the multiscale convolution and attention mechanism, aiming to improve the detection accuracy, speed, and efficiency. This project collects 2841 CT-scan images of COVID-19 to build datasets. However, the experimental dataset was small and unbalanced. Through training, validation and testing, the proposed model is compared with the mainstream deep learning technique enhanced neural network based models CNN, RNN, LSTM, Multi-scale CNN and Multi-scale CNN with Attention mechanism.

Methodology

Inspired by Multiscale CNN (MSCNN) and attention mechanism, a MSCNN with attention (AMSCNN) based COVID-19 infected species recognition method is proposed here.

After multiscale convolutions of 3X3, 3X3 and 3X3, attention is employed to make the spatial variation robustness of the input data stronger. Different levels of features are fused to improve the model's ability to express image semantic features, and SoftMax classifier is used for COVID-19 infected species recognition in the output decision layer. Different from MSCNN, AMSCNN utilizes the attention mechanism to capture rich contextual relationships for better feature representations and decrease the size of training and model parameters. MSC block has three convolutional operations with different kernel sizes. MSC within the same layers of CNN helps the model to secure low-frequency and high-frequency details of the original images.

CNN Part

CNN part uses a single-size convolutional kernel and single-channel mode to extract features from the input image, while extracting and enhancing feature information in the form of alternating convolution and pooling operations. Max pooling layer is applied to filter the extracted features by selecting the maximum value of each filter and then reduce the dimensionality. The convolution operation is output by the feature map of the current layer after the activation function. The calculation formula for the convolution layer is as follows:

$$x_j^l = f \left(\sum_{i=M_j} x_i^{l-1} \times k_i^l + b_j^l \right),$$

where k is the convolution kernel, M_j is the input feature, and b is the bias value.

The pooling operation is a form of non-linear downsampling, which can reduce the size of the feature maps extracted from the convolutional layers to achieve spatial invariance. The operation leads to faster convergence and improves the generalization performance. When the feature map x_j^l is passed to the pooling layer, the pooling operation is applied to the feature map x_j^l , which produces a pooled feature map x_j^{l+1} as the output. The max-pooling and average-pooling operations are often used, which are calculated respectively as follows :

$$x_j^{l+1} = \max_{i \in R_j} x_j^l,$$
$$x_j^{l+1} = \text{average}_{i \in R_j} x_j^l,$$

where, R_j is the j^{th} pooling region in feature map x_j^l and i is the index of each element within it.

The convolution operation is used to extract feature maps, while the pooling operation is to adjust the size of feature maps without changing the number of the feature maps.

In general, the non-linearity activation function ReLU (Rectified Linear Units) is used in the middle full-connection layer. The mathematical expression is as follows :

$$f(x_j^l) = \max(0, x_j^l), \text{ where } x_j^l \text{ is the feature map in the } l\text{th layer.}$$

Multi-Scale Convolution (MSC) Part

MSC can extract the multiscale features, containing several MSC blocks. Each block consists of an MSC layer and a flat convolutional layer, as shown in Figure just below.

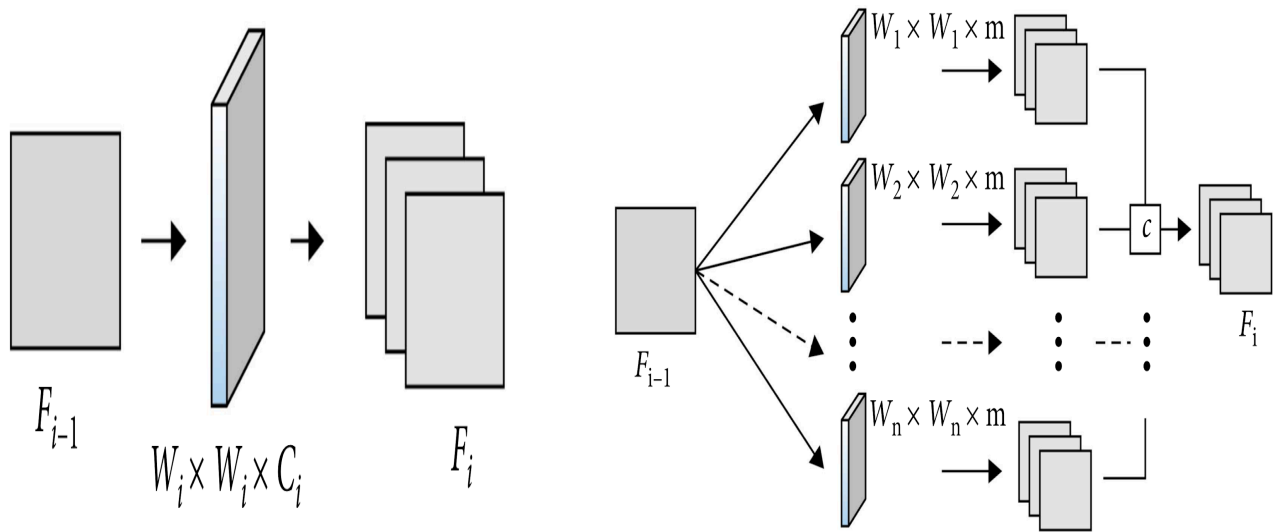


Figure : Difference between the basic convolutional layer of CNN and multiscale convolutions. (a) Convolutional layer with one convolution and (b) convolutional layer with MSCs.

The output feature map of the l th MSC layer with depth can be calculated as,

$$F_m^l = \max(0, \text{Concat}(f_1^l, f_2^l, f_3^l)), \text{ where } f_1^l, f_2^l, f_3^l \text{ are the feature maps obtained after MSC calculated as } f_1^l = w_1^l F_{i-1}^l + b_1^l, f_2^l = w_2^l F_{i-1}^l + b_2^l, f_3^l = w_3^l F_{i-1}^l + b_3^l, w_i^l (i = 1, 2, 3) \text{ is a convolutional filter, and } b_i^l (i = 1, 2, 3) \text{ is bias.}$$

These filters $w_i^l (i = 1, 2, 3)$ with sizes 3×3 , 3×3 and 3×3 are utilized to convolve with the F_{i-1}^l feature map. b_m^l is added to each feature map of the l^{th} MSC layer. Each convolution

operation is output in feature maps and then they are merged by concatenating along the spectral dimension. The next layer of MSC is a 3X3 convolutional layer to reduce the spectral dimension of feature maps. To facilitate gradient flow in the training process of the network, a skip connection is used after every two layers.

Attention Part

Attention mechanisms teach systems to pay attention to focus on important information and ignore irrelevant information. Suppose that the convolution kernel that inputs the attention structure is X , and it is retained as one of the inputs of the residual branch, X belongs to $R^{H \times W \times C}$. H , W , and C represent the length, width, and number of channels of the feature map. It is then sent to two separate branches for two different types of pooling operations. Let the global average-pooling process be F_{avg} and the global maximum pooling process be F_{max} and the outputs of F_{avg} and F_{max} be Att_{avg} and Att_{max} , the Att_{avg} belongs to $R^{1 \times 1 \times C}$, and Att_{max} belongs to $R^{1 \times 1 \times C}$. The one-dimensional weight sequence Att_{avg} can filter out the global background information of the target object, while Att_{max} can highlight the saliency of the target object. Let $X = [x_1, x_2, \dots, x_m]$, where x_c represents the parameters of the c^{th} convolution kernel. The calculation relation is as follows:

$$Att_{avg} = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W x_c(i, j) = F_{avg}(x_c),$$

$$Att_{max} = \operatorname{argmax} \left(\sum_{i=1}^H \sum_{j=1}^W x_c(i, j) \right) = F_{max}(x_c).$$

These two kinds of attention maps are input into 1x1 convolution and then are fused in a cascade way to generate attention map of the entire spatial information as follows :

$$Att = \delta \left(\operatorname{Conv}_{1 \times 1} \left[F_{avg}(x_c); F_{max}(x_c) \right] \right),$$

$$F = X \otimes Att,$$

where ‘delta’ is Sigmoid activation function, $\operatorname{Conv}_{1 \times 1}$ is to fuse two pooling operations with the size of 1x1 kernel, $[]$ is a cascading operation, and F is the attention weighted feature map.

The attention feature is obtained by adding the channel-wise representation and space-wise representation : $\mathbf{output}_{ch} = \mathbf{output}_{avg} + \mathbf{output}_{max}$, where $\mathbf{output}_{avg}$ and $\mathbf{output}_{max}$ are the channel-wise representation and space-wise representation, respectively.

Fusion Part

As mentioned earlier, the multiscale feature extraction module allows the network to result in a similar type of feature maps of each input image. In order to fuse the corresponding level of features from each source image, the extracted features are merged together by feature fusion concatenation operation as $\mathbf{F}_M = \text{Concat}(\mathbf{f}_1^m, \mathbf{f}_2^m)$, where $\mathbf{f}_1^m$ and $\mathbf{f}_2^m$ are the feature maps obtained by feature extraction from two original images I_1 and I_2 , respectively, and $\mathbf{F}_m$ is the fused feature representation. Later, this fused image representation is utilized in the reconstruction module as an input for the restoration of the fused image.

The simple fusion method is to concatenate the results of the MSC layer and flatten them in order to feed it to the classification module.

Activation Part

Two fully connected layers use ReLU to model the abstract representation of the leaf feature maps, which is calculated as follows : $f_{re} = \text{ReLU}(WX + b)$.

Classification Part

The image classification task is implemented by the SoftMax classifier. Its objective function is formulated as follows :

$$J(W) = -\frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C \ell(y_n == c) \log \frac{\exp(W_c^T X_i)}{\sum_{p=1}^C \exp(W_p^T X_i)},$$

where, $(X_i, y_i)(i = 1, 2, \dots, N)$ is the training set, X_i is an i th training sample, and y_i is the corresponding label. N and C are the numbers of training samples and classes, and $l(*)$ is an indicator function.

The above equation is incorporated with the proposed AMSCNN architecture and is optimized by using the stochastic gradient descent algorithm, and the deep learning toolbox is MatConvNet.

Gradient descent algorithm is adopted for optimization, and the weight of gradient descent is updated as

$$W_{\text{new}} = W_{\text{old}} - \eta \frac{\partial E}{\partial W_{\text{old}}},$$

where, W_{old} and W_{new} are weights before and after updating, respectively, n is the learning rate, and E is the composite function of weight W .

Experiments and Analysis

Recently, I came across an interesting dataset in Kaggle named 'sarscov2-ctscan-dataset' while searching for project ideas for my end-of-course Computer Science Final project assignment. It has 2481 detailed, full-resolution lung scans of patients who were diagnosed with the SARS-COV-2 along with segmentation of lungs and infections made by medical experts.

Preview Images and Labels

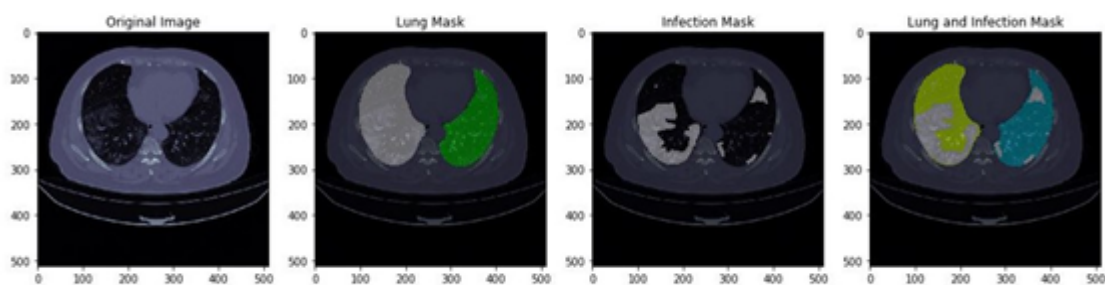


Fig. : 2-dimensional render of a CT scan showing the segmentation of healthy and infected pulmonary tissue

Reading through the python scripts and notebooks people have made utilizing the dataset got me thinking, what could I make using this data that could be applied in real-life while successfully completing my academic requirements?

Finally, after discussing it over multiple meetings with my respected guide ma'am, we arrived at the conclusion that we would attempt to create a model that could tell us the chances of being infected with the *corona virus* in the after pandemic situation before rushing to hospitals demanding invasive (defensive) procedures from already overworked medical staff around the world.

Before we started work on how the user(s) would interact with our compiled model, we had to select and train deep learning models on the found dataset and then test and validate the built model on new CT scans; so that our model could predict, classify and segment infected parts of the lung as seen in the scans.

After some research, we found that models specially designed for working on medical scan images, resulted in much higher mean accuracy compared to generalized CNN models or machine learning models that work directly on numeric data like linear SVMs and Logistic Regression models.

Here, It will be tried to explain the basic methodology and steps that go into preparing the raw data for this project and how image segmentation problems (our initial task) can be solved using popular deep learning techniques and Python language. Also, I

have included snippets of code and outputs wherever possible to help understand the comprehensive process being followed in brief.

We start by importing the required libraries and downloading the entire dataset to our environment of choice (Here, Microsoft VS Code IDE is used. Google's Colab platform can also be used for the extra ram and GPU benefits).

```
!kaggle datasets download plameneduardo/sarscov2-ctscan-dataset
```

```
[6]
```

```
... Downloading sarscov2-ctscan-dataset.zip to /content
99% 228M/230M [00:04<00:00, 20.3MB/s]
100% 230M/230M [00:04<00:00, 55.2MB/s]
```

Downloading the dataset as a .zip

```
!unzip sarscov2-ctscan-dataset.zip
```

```
[7]
```

```
... Archive: sarscov2-ctscan-dataset.zip
inflating: COVID/Covid (1).png
inflating: COVID/Covid (10).png
inflating: COVID/Covid (100).png
inflating: COVID/Covid (1000).png
inflating: COVID/Covid (1001).png
inflating: COVID/Covid (1002).png
inflating: COVID/Covid (1003).png
inflating: COVID/Covid (1004).png
inflating: COVID/Covid (1005).png
inflating: COVID/Covid (1006).png
inflating: COVID/Covid (1007).png
inflating: COVID/Covid (1008).png
inflating: COVID/Covid (1009).png
inflating: COVID/Covid (101).png
inflating: COVID/Covid (1010).png
inflating: COVID/Covid (1011).png
inflating: COVID/Covid (1012).png
inflating: COVID/Covid (1013).png
inflating: COVID/Covid (1014).png
inflating: COVID/Covid (1015).png
inflating: COVID/Covid (1016).png
inflating: COVID/Covid (1017).png
inflating: COVID/Covid (1018).png
inflating: COVID/Covid (1019).png
...
inflating: non-COVID/Non-Covid (996).png
```

Extracting and accessing the metadata file

The metadata of our dataset has 4 major categories of images labeled as the original scans, lung masks, infection masks, and combined masks for all the images in the dataset. Running the shape command we get,

```
list
```

```
[12]
```

```
data_df=pd.DataFrame(data, columns = ['filename','ID','disease_type'])
data_df
```

```
[13]
```

	filename	ID	disease_type
0	Covid (675).png	0	COVID
1	Covid (366).png	1	COVID
2	Covid (368).png	2	COVID
3	Covid (557).png	3	COVID
4	Covid (582).png	4	COVID
...	...	...	...
2476	Non-Covid (750).png	2476	non-COVID
2477	Non-Covid (557).png	2477	non-COVID
2478	Non-Covid (614).png	2478	non-COVID
2479	Non-Covid (393).png	2479	non-COVID
2480	Non-Covid (896).png	2480	non-COVID

2481 rows x 3 columns

2481 full-fledged images having corresponding copies under each of the 3 labels highlighting different aspects of the original scan.

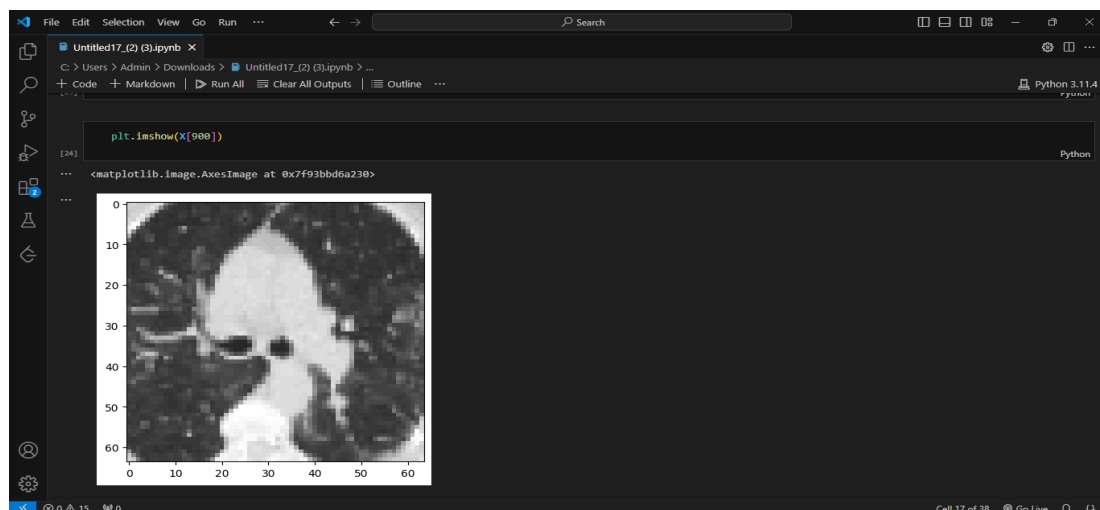
After looking through the data, the first major step we need in each ML/DL problem is its analysis and adequate pre-processing which will help us reduce common problems such as bias, code complexity, training time and such. The main steps that has been followed in the pre-processing stage are:

1. Removing incomplete and faulty images

This is a major step when using a huge amount of unverified data from multiple sources as many of the images which we downloaded for use may end up being cropped, low-res and unevenly colored which may lead to problems during training the model. We can make interpolations to account for missing numeric data, but usually faulty image data is truncated to avoid problems in the subsequent steps.

2. Separate model for empty mask prediction

To ensure that our model does not overcompensate for slices that do not contain any part of the infection we will create an independent model to calculate empty masks in the input CT and only train our first model on slices that have visible infections in the render. On figuring out that 497 slices were complete black masks, we will exclude these from pre-processing as we do not want to bother the segmentation model with this.



3. Use of enhancement algorithms to improve overall performance

Using popular image enhancement techniques on individual scans we can drastically increase the performance of our model by helping it to distinguish between healthy and infected tissue more easily. In this approach I have used Contrast Limited

Adaptive Histogram Equalization (CLAHE) to enhance the difference between the pieces of intertwined infected cells between the pulmonary cells.

Using custom read functions for the .nii format of the scans (`read_nii`) and plotting the original and enhanced scans along with their respective histograms, we can easily see the effect a single function has on separating the part of an image we need for our model.

4. Cropping the Region of Interest (ROI) using Otsu's binarization and other methods

The images possess a lot of black space containing no part of the infection and parts that we are not interested in like the diaphragm below the lungs. These will take up valuable RAM space and unnecessary computing power. A possible solution is cropping the slices to only contain the ROI as per problem statement and use case. After cropping, we use Otsu's thresholding method to avoid having to choose discrete values which are dynamically assigned by the algorithm to return infection and/or lung masks.

5. Data Augmentation

Neural Networks are only as good as the quality and quantity of data you provide it with. In image segmentation and classification problems specifically, if the ratios of the number of images under different labels are skewed or images under different training labels are too similar, it may lead to bias errors which will lead to wildly incorrect classification and/or segmentation. Data augmentation methodologies aim to eliminate this problem by using the already existing data to create new iterations which differ slightly from their source to sensitize your model to new variables which will in turn, help increase performance on new data that the model has never encountered. It also helps increase the apparent number of training examples as the model treats every new iteration, as a different example. To achieve efficient augmentation in our dataset, we will define a pipeline that takes in our already existing images and returns a sequence of scan slices after our user-defined transformations have been applied to it.

```
import imgaug.augmenters as iaa

class_id = {'COVID': 0, 'non-COVID': 1}
folder_path = '/content/'

X = []
Y = []

for i in range(len(data_df)):
    file_path = folder_path + '/' + data_df.iloc[i, 2] + '/' + data_df.iloc[i, 0]
    img = cv2.imread(file_path)
    img_resize = cv2.resize(img, (64, 64))
    img_resize_normalise = img_resize / 255
    X.append(img_resize_normalise)
    Y.append(class_id[data_df.iloc[i, 2]])

    vflip = iaa.Flipud(p=1)
    hflip = iaa.Fliplr(p=1)
    translating = iaa.Affine(translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)})
    scaling = iaa.Affine(scale={"x": (0.8, 1.2), "y": (0.8, 1.2)})
    rotation = iaa.Affine(rotate=(-40, 40))
    shearing = iaa.Affine(shear=(-40, 40))

    augmented_images = [vflip, hflip, translating, scaling, rotation, shearing]
    for aug in augmented_images:
        augmented = aug.augment_image(img_resize_normalise)
        X.append(augmented)
        Y.append(class_id[data_df.iloc[i, 2]])
```

```
X = []
Y = []

for i in range(len(data_df)):
    file_path = folder_path + '/' + data_df.iloc[i, 2] + '/' + data_df.iloc[i, 0]
    img = cv2.imread(file_path)
    img_resize = cv2.resize(img, (64, 64))
    img_resize_normalise = img_resize / 255
    X.append(img_resize_normalise)
    Y.append(class_id[data_df.iloc[i, 2]])

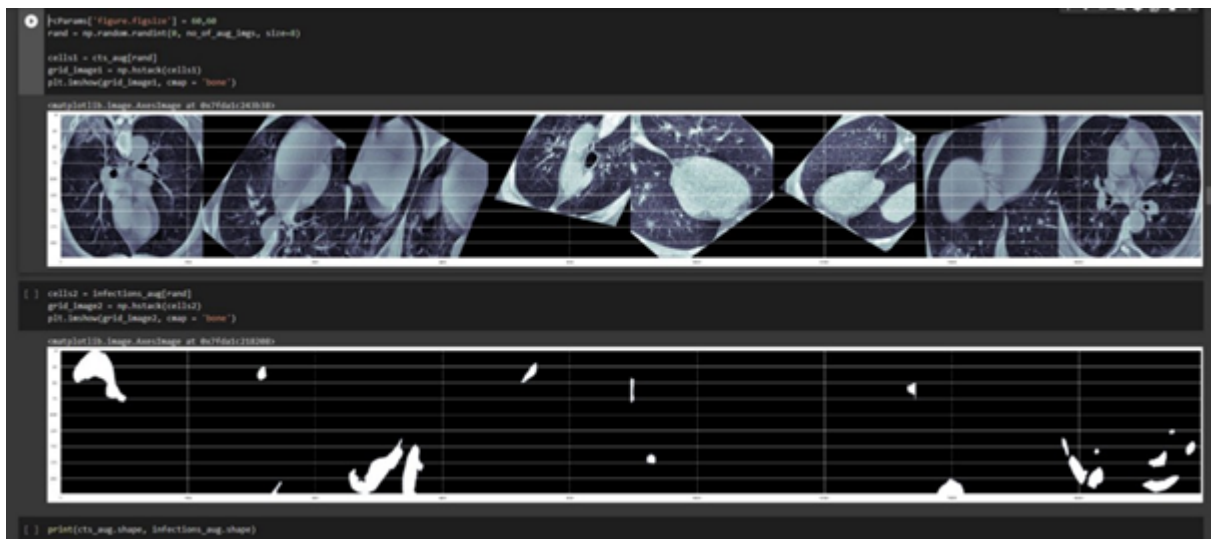
    vflip = iaa.Flipud(p=1)
    hflip = iaa.Fliplr(p=1)
    translating = iaa.Affine(translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)})
    scaling = iaa.Affine(scale={"x": (0.8, 1.2), "y": (0.8, 1.2)})
    rotation = iaa.Affine(rotate=(-40, 40))
    shearing = iaa.Affine(shear=(-40, 40))

    augmented_images = [vflip, hflip, translating, scaling, rotation, shearing]
    for aug in augmented_images:
        augmented = aug.augment_image(img_resize_normalise)
        X.append(augmented)
        Y.append(class_id[data_df.iloc[i, 2]])

assert len(X) == len(Y)
```

The Augmentation Pipeline

After execution, we can plot some random slices to see a part of our new set of images created after passing our original dataset through the pipeline.



Newly Generated Slices

Our preprocessing stage is now finished, now we will overlay our infection masks over their corresponding CT scans before creating and running our model on the augmented dataset.

Now, we split the test and train data and define the loss function and metrics to be used for our model.

```
File Edit Selection View Go Run ... Search
Untitled17_2 (3).ipynb X
C:\Users\Admin\Downloads> Untitled17_2 (3).ipynb > ...
+ Code + Markdown + Run All + Clear All Outputs + Outline ... Python 3.11.4
0.20784313725490197

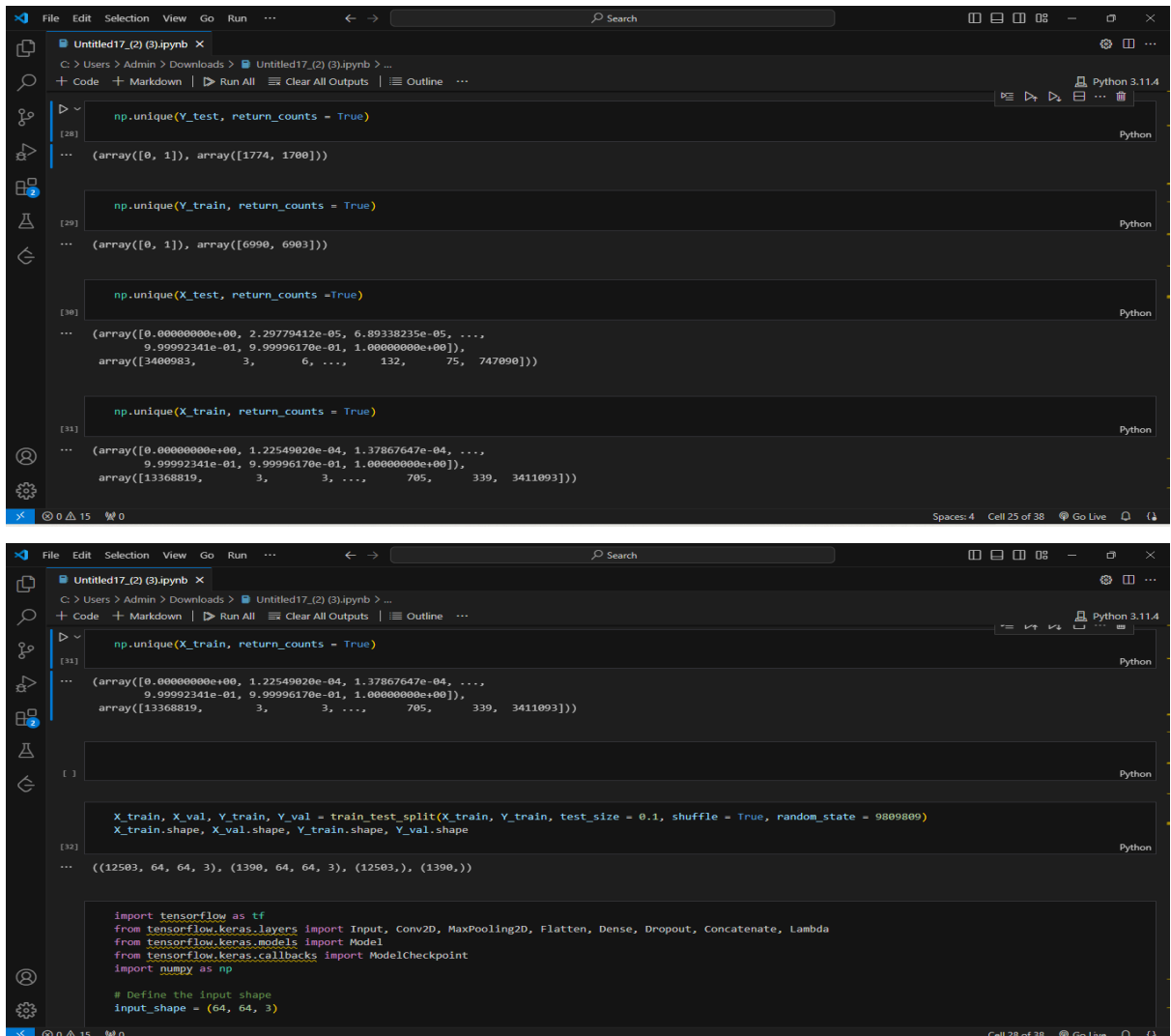
X=np.array(X)
Y=np.array(Y)
[26] Python

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, shuffle = True, random_state = 7657645)
X_train.shape,X_test.shape,Y_train.shape,Y_test.shape
[27] Python
... ((13893, 64, 64, 3), (3474, 64, 64, 3), (13893,), (3474,))

np.unique(Y_test, return_counts = True)
[28] Python
... (array([0, 1]), array([1774, 1700]))

np.unique(Y_train, return_counts = True)
[29] Python
... (array([0, 1]), array([6990, 6903]))

Cell 26 of 38 Go Live
```



```
np.unique(Y_test, return_counts = True)
Out[28]: (array([0, 1]), array([1774, 1700]))

np.unique(Y_train, return_counts = True)
Out[29]: (array([0, 1]), array([6990, 6903]))

np.unique(X_test, return_counts = True)
Out[30]: (array([0.00000000e+00, 2.29779412e-05, 6.89338235e-05, ...,
          9.99992341e-01, 9.99996170e-01, 1.00000000e+00]),
          array([3400983, 3, 6, ..., 132, 75, 747090]))

np.unique(X_train, return_counts = True)
Out[31]: (array([0.00000000e+00, 1.22549020e-04, 1.37867647e-04, ...,
          9.99992341e-01, 9.99996170e-01, 1.00000000e+00]),
          array([13368819, 3, 3, ..., 705, 339, 3411093]))

np.unique(X_train, return_counts = True)
Out[31]: (array([0.00000000e+00, 1.22549020e-04, 1.37867647e-04, ...,
          9.99992341e-01, 9.99996170e-01, 1.00000000e+00]),
          array([13368819, 3, 3, ..., 705, 339, 3411093]))

X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = 0.1, shuffle = True, random_state = 9809809)
X_train.shape, X_val.shape, Y_train.shape, Y_val.shape
Out[32]: ((12503, 64, 64, 3), (1390, 64, 64, 3), (12503,), (1390,))

import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Concatenate, Lambda
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
import numpy as np

# Define the input shape
input_shape = (64, 64, 3)
```

All we must do now is define, compile and fit our model on the data and then use the metrics of our choice to evaluate the performance of our model. Tuning our model hyperparameters is essential here, as it can mean the difference between a highly accurate and efficient program and an inaccurate, slower model. The optimal learning rate will be dependent on the topology of our loss landscape, which is in turn dependent on both our model architecture and our dataset. While using a default learning rate (the defaults set by our deep learning library) may provide decent results, we can often improve the performance or speed up training by searching for an optimal learning rate. Using exponential decaying learning rates and a cosine annealing scheduler are popular methods which produce good results.

After fitting we can now look at our model's predicted infection masks on the test data scans. Further information can be obtained in post-processing by analyzing DICE and IOU scores using the metrics we defined earlier. Using other metrics like precision and recall may also be used to test the performance of the model.

Experimental results & analysis

Multi-Scale CNN model built for Covid-19 Classification

The following figure describes the architecture of our proposed MS-CNN model built as a part of this project for ‘Covid-19 CT-scan medical images classification’.

```
... Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 64, 64, 3)]	0	[]
conv2d (Conv2D)	(None, 64, 64, 32)	896	['input_1[0][0]']
conv2d_1 (Conv2D)	(None, 64, 64, 32)	9248	['conv2d[0][0]']
max_pooling2d (MaxPooling2D)	(None, 32, 32, 32)	0	['conv2d_1[0][0]']
conv2d_2 (Conv2D)	(None, 32, 32, 64)	18496	['max_pooling2d[0][0]']
conv2d_3 (Conv2D)	(None, 32, 32, 64)	36928	['conv2d_2[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 64)	0	['conv2d_3[0][0]']
conv2d_4 (Conv2D)	(None, 16, 16, 128)	73856	['max_pooling2d_1[0][0]']
conv2d_5 (Conv2D)	(None, 16, 16, 128)	147584	['conv2d_4[0][0]']
global_average_pooling2d (GlobalAveragePooling2D)	(None, 32)	0	['conv2d_5[0][0]']

```
...  
Total params: 15014948 (57.28 MB)  
Trainable params: 15014948 (57.28 MB)  
Non-trainable params: 0 (0.00 Byte)
```

Figure : Architectural configuration of proposed Multi-Scale CNN model

Here is a brief but vital description of the architecture of our proposed CNN model built for ‘Covid-19 CT-scan medical images classification’ according to the above figure. Firstly, an input layer of output-shape (None, 64, 64, 3) is created. This ensures that the images are given as input into the model according to the extended dataset created by implementing data augmentation technique on the original public dataset available in Kaggle named ‘sarscov2-ctscan-dataset’ i.e., a well-defined dataset consisting of a moderate number of detailed, full-resolution lung CT-scan medical images of patients who were diagnosed with the SARS-COV-2 along with segmentation of lungs and infections made by medical experts. After that, first scale of Convolution layers built with two layers of 2D convolution both of output-size (None, 64, 64, 32), before a 2D max-pooling layer, is introduced to perform the convolution operation twice between the original image matrix and the filter or mask matrix, as the data moves deeper into the layers of our proposed model. A 2D Max-Pooling layer of output-shape (None, 32, 32, 32) is then created by our proposed model, responsible for reducing the spatial size of

the Convolved Feature. Indeed, this is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, this layer is useful for extracting the rotational and positional invariant dominant features to maintain the model-training process effectively, and performs as a Noise Suppressant that discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. After capturing the output of this first pooling layer efficiently, it is carried forward through another scale of Convolutional layers, which again consists of two 2D convolution layers both of output-size (None, 32, 32, 64); then followed by a 2D Max-Pooling layer of output-shape (None, 16, 16, 64). The intermediate output of this second pooling layer is then efficiently captured to continue through the third scale of Convolutional layers made of the same two 2D convolution layers both of output-size (None, 16, 16, 128). Ultimately a 2D Global Average-Pooling layer of output-size (None, 32) is framed as the last layer of our proposed model, to replace the Fully Connected layers in classical CNNs, which indeed helps in generation of one feature map for each corresponding category of the classification task in the last ‘*mlpconv*’ layer that consists of multiple fully connected layers with a non-linear activation function.

In the task of using Convolutional Networks, more specially Multi-Scale CNN with Attention mechanism for ‘*Covid-19 CT-scan medical images classification*’, accuracy and precision, we divide our augmented dataset into 3:1 ratio for train and test respectively. With higher up given architectural configuration of Multi-Scale CNN and the given parameters we train our model for 20 epochs and then perform the testing. Finally, it provides the train-test accuracy of 98.02% and 92.81% respectively.

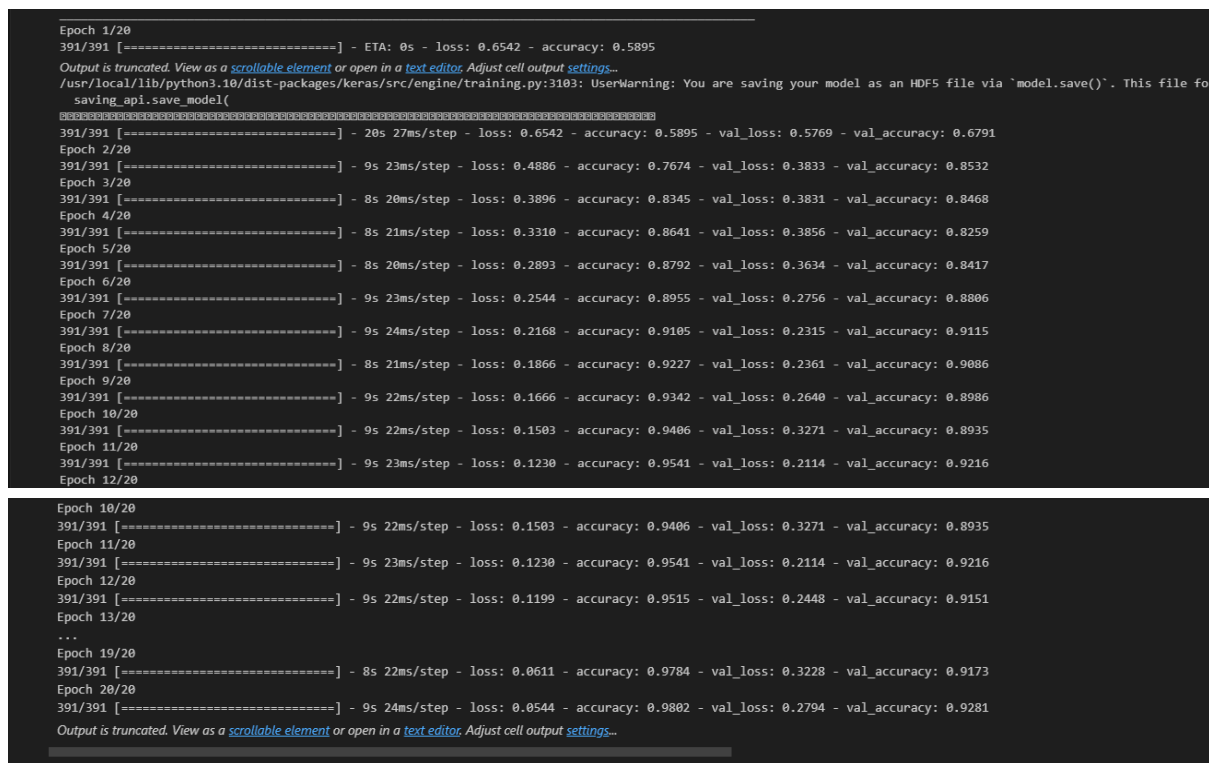


Figure : Training epochs of proposed Multi-Scale CNN model

The following multiple-line graph efficiently demonstrates the accuracy, loss, validation accuracy and validation loss according to the above model-training epochs corresponding to our proposed Multi-Scale CNN model with Attention mechanism (AMS-CNN).

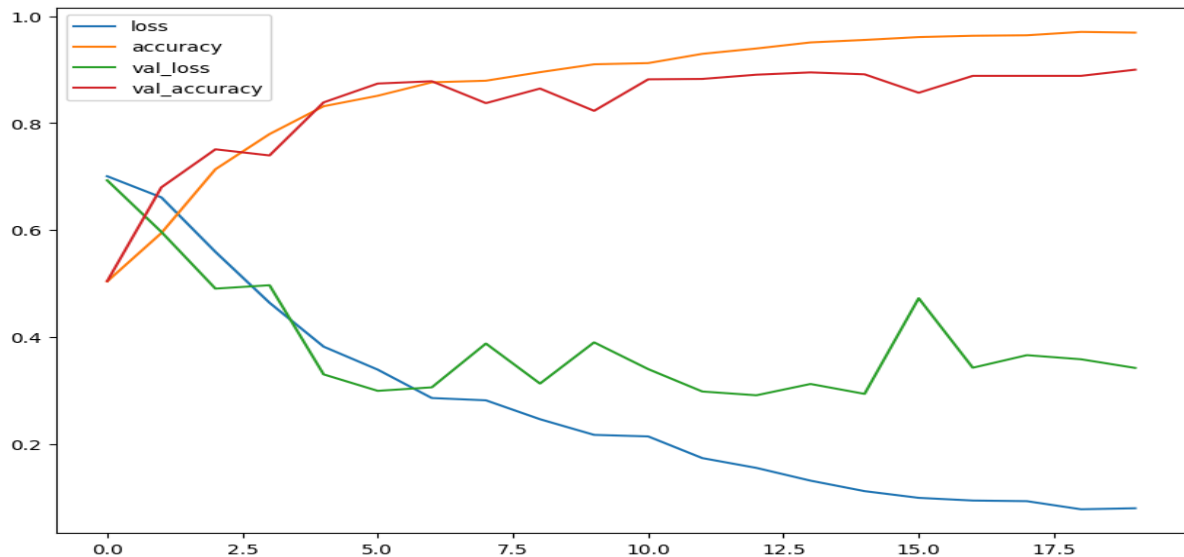


Figure : Multiple line graph for Accuracy, Loss, Validation-Accuracy & Validation-Loss representation

Our proposed Multi-Scale CNN model with Attention mechanism comes up with a very handsome Test-Accuracy of 91.45%, as seen in the next succeeding figure.

```

best_model.evaluate(X_test, Y_test)
[38]
... 109/109 [=====] - 1s 7ms/step - loss: 0.2961 - accuracy: 0.9145
... [0.29608291387557983, 0.9145077466964722]

```

Figure : Test-Accuracy and Test-Loss evaluation for the proposed Multi-Scale CNN model

An Receiver Operating Characteristic (ROC) curve, a graph showing the performance of a classification model at all classification thresholds, corresponding to the proposed Multi-Scale CNN model with Attention mechanism and deep learning enhancements is being manifested just down below.

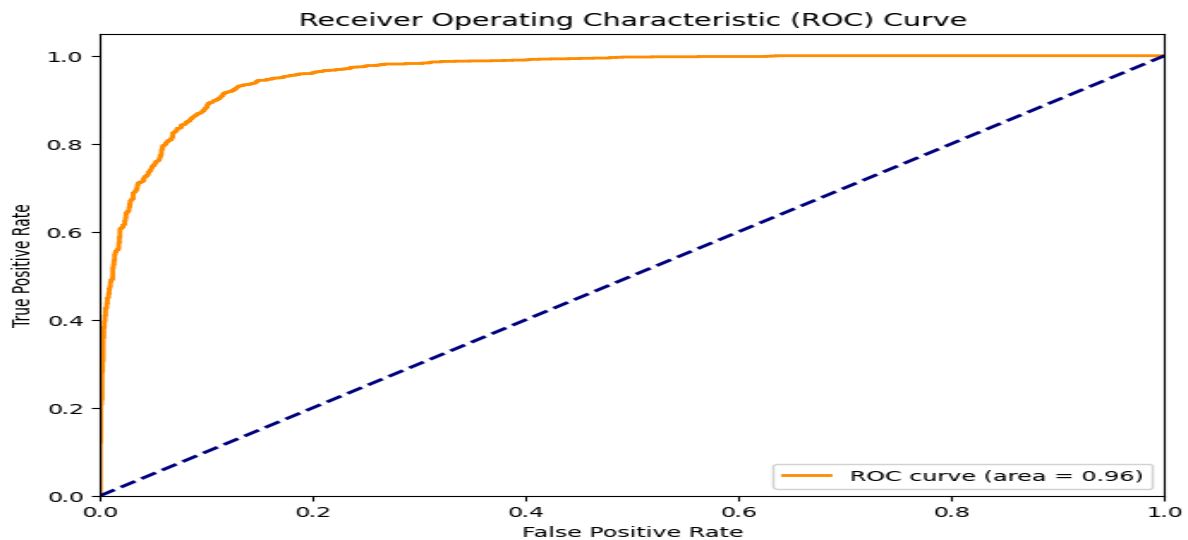


Figure : Receiver Operating Characteristics (ROC) Curve

We also calculated the 'Confusion Matrix' for the proposed model to evaluate the results (accuracy) produced by this model using standard evaluation matrices and to show the effectiveness and performance measurement for this machine learning classification; as given any convolutional network, this matrix shows where the model is getting confused during prediction. The generated results are shown below :

```
from sklearn.metrics import confusion_matrix
y_probs = best_model.predict(X_test)
y_pred = (y_probs > 0.5).astype(int)
conf_matrix = confusion_matrix(Y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

[41]

```
... 109/109 [=====] - 2s 14ms/step
Confusion Matrix:
[[1614  160]
 [ 137 1563]]
```

Figure : 'Confusion Matrix' calculation

```

tp = conf_matrix[0, 0]
tn = conf_matrix[1, 1]
fp = conf_matrix[0, 1]
fn = conf_matrix[1, 0]
precision = tp / (tp + fp)
recall = tp / (tp + fn)
f1_score = 2 * (precision * recall) / (precision + recall)
specificity = tn / (tn + fp)
mcc = (tp * tn - fp * fn) / np.sqrt((tp + fp) * (tp + fn) * (tn + fp) * (tn + fn))
print("Precision:", precision)
print("Recall (Sensitivity):", recall)
print("F1-score:", f1_score)
print("Specificity:", specificity)
print("Matthews correlation coefficient:", mcc)

```

[42]

```

... Precision: 0.9098083427282976
Recall (Sensitivity): 0.92175899486008
F1-score: 0.9157446808510639
Specificity: 0.9071387115496228
Matthews correlation coefficient: 0.8290588912501928

```

Figure : Precision, Recall (Sensitivity), F1-score, Specificity and Matthews Correlation Coefficient evaluation

<i>Model Type</i>	<i>No. of Layers</i>	<i>No. of Parameters</i>	<i>Input Image Shape</i>	<i>No. of output classes</i>	<i>Model Size</i>	<i>Processing Unit Type</i>
Classical CNN	7	3591	64, 64, 3	2	14.03 KB	CPU
RNN	8	14846065	64, 64, 3	2	56.13 MB	CPU
LSTM	9	14846465	64, 64, 3	2	56.13 MB	CPU
Multi-Scale CNN	19	59041601	64, 64, 3	2	225.23 MB	GPU
Multi-Scale Attention CNN	30	15014948	64, 64, 3	2	57.28 MB	GPU

Table 01 : No. of layers, No. of parameters, Input image shape, No. of output classes, Model Size & Processing Unit Type comparison

<i>Model Type</i>	<i>Test Accuracy</i>	<i>Loss</i>	<i>Train Accuracy</i>	<i>ROC AUC</i>	<i>Training Time</i>
Classical CNN	0.8031	0.3724	0.8244	0.92	4 mins 22 secs
RNN	0.8610	0.3358	0.9041	0.94	1 hr 46 mins 2 secs
LSTM	0.8385	0.3604	0.8258	0.92	2 hrs 3 mins 20 secs
Multi-Scale CNN	0.8981	0.3345	0.9694	0.96	5 mins 52 secs
Multi-Scale Attention CNN	0.9145	0.2961	0.9802	0.97	3 mins 2 secs

Table 02 : Test Accuracy, Loss, Train Accuracy, ROC AUC & Training Time comparison

<i>Model Type</i>	<i>Precision</i>	<i>Recall (Sensitivity)</i>	<i>F1-score</i>	<i>Specificity</i>	<i>Matthews Correlation Coefficient (MCC)</i>
Classical CNN	0.8224	0.8659	0.8436	0.8239	0.6896
RNN	0.8754	0.8556	0.8654	0.8668	0.7219
LSTM	0.7818	0.8885	0.8318	0.7977	0.6826
Multi-Scale CNN	0.8833	0.9142	0.8985	0.8824	0.7967
Multi-Scale Attention CNN	0.9098	0.9218	0.9157	0.9071	0.8291

Table 03 : Precision, Recall (Sensitivity), F1-score, Specificity & Matthews Correlation Coefficient evaluation (MCC) comparison

Conclusion

'Attention based Multi-Scale CNN algorithm for Covid-19 Classification' deals with the classification of CT-scan medical images based on the features that these images contain along with choosing prominent features adaptively by accommodating different feature weights in the diverse scale domains. *Covid-19 CT-scan medical images classification* can be widely used to automatically classify these images mainly as *'infected'* or *'non-infected'*. This project aimed at working on *'Attention based Multi-Scale CNN algorithm for Covid-19 Classification'* on an interesting public dataset available in Kaggle named *'sarscov2-ctscan-dataset'* i.e., a well-defined dataset consisting of a moderate number of detailed, full-resolution lung CT-scan medical images of patients who were diagnosed with the SARS-COV-2 along with segmentation of lungs and infections made by medical experts. This analysis can be used to recognize or separate the Covid-19 infected lungs depending on their CT-scan images based on the features that these images contain along with choosing prominent features adaptively by accommodating different feature weights in the diverse scale domains for rapid identification and early diagnosis of COVID-19 for helping doctors to identify diseases and their effects more quickly to avoid the resulting in the diversity of diagnosis results through medical imaging. Although, the main focus of this project work was to explore the implementation of Multi-Scale CNN with Attention mechanism for the post-segmentation classification of CT-scan medical images to identify the Covid-19 infected lungs more quickly and efficiently; Here the performance of our proposed model is also comparatively quantified on the basis of the traditional distinct neural networks, such as, basic CNN, RNN, LSTM and Multi-Scale CNN etc. Our proposed Attention based Multi-Scale CNN (AMSCNN) methodology outperforms all the other conventional neural network models just mentioned before, when we want our model to learn from different feature weights in the diverse scale domains for rapid identification or classification. The extra ability to choose prominent features adaptively by accommodating different feature weights in the diverse scale domains of our proposed Multi-Scale network with added attention mechanism, pushes it further steps ahead of the other discussed conventional neural networks.

We completed building our AMS-CNN model for post-segmentation classification of a public dataset *'sarscov2-ctscan-dataset'* available in Kaggle into mainly two *'infected'* and *'non-infected'* categories. We hoped to find a conclusion that the CNN model would do significantly well with recognizing or classifying the CT-scan medical images as it's recognizing the pattern where convolution thrives, instead what we found is quite amazing and that is any conventional model with fixed weight performs worse than what our proposed AMS-CNN model. We started off with a horrifying result of below 80% with only a classical CNN model, and the data was not adequately preprocessed. We know that CNN with added deep learning enhancements performs well for complex image data structures, but this time after providing so numerous augmented dataset

extending the data-size from 2,841 to more than 10,000 where we finally ended. The ultimate training accuracy was well above 90% but the testing accuracy was stuck only at 80%. We found little hope given that we gave so much data to the model and still were not able to get a better result. Later, we appended the deep learning technique and attention mechanism along with multi-scale configuration to it and our training accuracy was 98.02% after 20 epochs, but the testing accuracy did increase by 2% which we finally submitted. Of course, we did afford more epochs for the AMS-CNN but it was not overfitting at all, the same cannot be guessed for all the CNN models. In fact, we observed validation accuracy going down after training more epochs and finally made it to only 20 epochs which is considerably low from that we have used at the other discussed conventional models.

So, we guess the use of Multi-Scaled deep learning technique enhanced trainable convolution layers with attention mechanism, does increase the accuracy significantly. In contrast to the previous studies that were focused mainly on the spatial attention having numerous weight parameters or channel attention with fewer parameters, this work employs the attention mechanism that focuses on multi-scale attention for COVID-19 classification. Indeed, we also tried making the layers of different standard pre-trained models trainable and almost all of them did give that much accuracy which is just a little less than what we got for the Multi-scale CNN model that we tried out. Also, we have gone through a comparative study of our proposed model with respect to different discussed conventional neural network models where the test accuracy came to be higher enough at what we have got in our proposed model in 20 epochs only. As it can be seen, our model performs surprisingly good on a relatively low number of scans due to our use of attention mechanism on the quality of scans during pre-processing and augmenting our existing data to allow the model to be trained for a greater variety of images from a smaller dataset. As marked out in the Results section, we needed at least 3-Scale trainable Convolution layers to get to a respectable accuracy. We guessed that the accuracy would be worse but 'above 98%' was not a bad one, so we conclude that Multi-Scale CNN with Attention mechanism recognizes a general pattern well. In fact, despite having a standard dataset with adequate amount of data for the project we find a good test accuracy as well.

Further improvements can be made to this approach to CT scans segmentation by using newer image enhancement techniques and creating an even larger augmented dataset of images by increasing the number of copies created for each slice and/or adding more transformations functions to the augmentation pipeline. This attention based classification technique using Multi-Scale CNN methodology can also be further utilized in identification, recognition or classification of other similar type diseases. So, further work is needed to create a model for the classification of CT scans showing whether the patient has COVID-19 or some other pulmonary defect using the infection masks predicted by our code defined here. Finally, if we want to formulate an application which uses our model or incorporate it within any application, everything has to be linked to a Python web framework like [Streamlit](#) or [Flask](#) to create a user interface easily usable by everyone as a utility application. We could also try a transfer learning approach using

multiple models for the classification task of this problem which will allow us to improve every subsequent model by forwarding the errors made by the last generation. As it is being expected that the next wave of Covid cases might be started popping up around the world since the start of winter in the northern hemisphere, the research conducted and products built by the open-source community to provide additional support to our society and the medical industry, are slowly becoming vital in maintaining the quality of life around the globe. Whereas we now can also explore further applying this type of enhanced convolutional neural network to different domain problems nearly related to ours and build new use cases.

References

1. Jose Manuel Saavedra. ***Convolutional Neural Models and Multi-Scale Representations.***
Link - <https://medium.com/impresee/convolutional-neural-models-and-multi-scale-representations-e1f4008c5fff>
2. Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie. ***Feature Pyramid Networks for Object Detection.***
3. Jonathan Long, Evan Shelhamer, Trevor Darrel. ***Fully Convolutional Networks for Semantic Segmentation.***
4. Mani Batra. ***Creating a Multi Scale Neural Network.***
Link - <https://medium.com/@manibatra23/creating-a-multi-scale-neural-network-3d9543a7f628>
5. Esma Sert. ***Medical Image Pre-Processing with Python.***
Link - <https://towardsdatascience.com/medical-image-pre-processing-with-python-d07694852606>
6. Emmanuelle Gouillart, Juan Nunez-Iglesias & Stéfan van der Walt. ***Analyzing Microtomography data with Python and the scikit-image library.***
Link - <https://ascimaging.springeropen.com/articles/10.1186/s40679-016-0031-0>
7. ***Detecting COVID-19 from Chest X-Ray images using CNN.***
Link - <https://www.geeksforgeeks.org/detecting-covid-19-from-chest-x-ray-images-using-cnn/>
8. ***Image Segmentation using TensorFlow.***
Link - <https://www.geeksforgeeks.org/image-segmentation-using-tensorflow/>
9. Wang Gaihua, Lin Jinheng , Cheng Lei, Dai Yingying, Zhang Tianlun. ***Instance segmentation convolutional neural network based on multi-scale attention mechanism.***
Link - <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0263134>
10. ***U-Net lung segmentation (Montgomery + Shenzhen).***
Link - <https://www.kaggle.com/code/eduardomineo/u-net-lung-segmentation-montgomery-shenzhen/notebook>
11. Xianfeng Wang, Chuanlei Zhang and Shanwen Zhang. ***Multiscale Convolutional Neural Networks with Attention for Plant Species Recognition.***
Link - <https://www.hindawi.com/journals/cin/2021/5529905/#experiments-and-analysis>
12. Kanishka Pandey. ***COVID-19 Infection and Lung Segmentation using CT Scans.***
Link - <https://lightning201220.medium.com/covid-19-infection-and-lung-segmentation-using-ct-scans-a5ab5b379272>
13. Geng Hong, Xiaoyan Chen, Jianyong Chen, Miao Zhang, Yumeng Ren & Xinyu Zhang. ***A multi-scale gated multi-head attention depthwise separable CNN model for recognizing COVID-19.***
Link - <https://www.nature.com/articles/s41598-021-97428-8>

14. Keying Ren, Geng Hong, Xiaoyan Chen & Zichen Wang. *A COVID-19 medical image classification algorithm based on Transformer*.
Link - <https://www.nature.com/articles/s41598-023-32462-2>
15. Dandi Yang, Cristhian Martinez, Lara Visuña, Hardev Khandhar, Chintan Bhatt & Jesus Carretero. *Detection and analysis of COVID-19 in medical images using deep learning techniques*.
Link - <https://www.nature.com/articles/s41598-021-99015-3>
16. Zahid Ullah, Muhammad Usman, Siddique Latif & Jeonghwan Gwak. *Densely attention mechanism based network for COVID-19 detection in chest X-rays*.
Link - <https://www.nature.com/articles/s41598-022-27266-9>