Using Deep Learning for detecting COVID-19 using Chest X-Ray Images

Ruchit Khushu*, Siddharth Kapoor*, Sujit Biswas*, Pankaj Chauhan*
College of Computing, Georgia Institute of Technology

Github

Abstract

The novel Coronavirus also known as COVID-19 has caused a global pandemic responsible for over 268 million infections as well for over 5.8 million deaths.There are multiple ways of diagnosing Covid-19 in patients e.g.*: - RT-PCR, Chest X-rays etc. In this project we propose to use Deep Learning Models and techniques to help with detection of COVID-19 through radiographic images (X-rays).*

1. Introduction/Background/Motivation

COVID-19 is a highly contagious disease that has resulted in a worldwide pandemic.One of the key challenges with any disease is an early and accurate diagnosis.Popular diagnostic tools for COVID-19 like RT-PCR ,CT-Scans or Chest X-rays have challenges like availability, accuracy and delays in getting back the results.

One of most widely used techniques of diagnosing COVID-19 currently is RT-PCR however the test kits are expensive, not available in bulk and test results take more than 24 hrs to come.

Another important diagnostic tool is the Chest X-ray. Chest X-ray centers are more easily accessible and the results take much lesser time.However visually/manually examining the X-ray images of COVID-19 cases can be challenging specifically distinguishing them from Viral Pneumonia and other lung infections is a challenge. This can sometimes result in false positives/negatives which in turn can lead to greater costs,efforts and risks.

We think there is a strong case for Deep Learning models to be utilized for early diagnosis of COVID-19 using X-ray images and that is what we have attempted in this project. We feel the medical community and specifically radiologists will be greatly benefited by having a faster, accurate and automated Deep-Learning based technique for detection of COVID-19 using chest X-ray images.

We used existing pre-trained SOTA models as well create our own CNN model to meet our objectives.

We used an existing chest X-ray image data-set "COVID-19 Radiography Database" prepared by the authors of the following paper: Can AI Help in Screening Viral and COVID-19 Pneumonia?. Dataset can be downloaded from here

The data-set consists of X-ray images for these 4 classes:- COVID-19, Lung-Opacity, Viral-Pneumonia and Normal with the following distribution: 3616 COVID , 1345 Viral Pneumonia, 6012 Lung Opacity and 10192 Normal X-ray images. Our aim is to classify each X-ray image into one of these 4 classes.

2. Approach

Our approach has fundamentally two aspects to it :

i) We chose five existing SOTA image detection/prediction models:- SqueezeNet (1_1) , MobileNet_V2 , ResNet18 ,VGGN19 and DenseNet201. We used transfer learning by starting with pre-trained models and then re-training the models on the X-ray image data-set with image augmentation to increase prediction accuracy. These SOTA models are pre-trained on ImageNet dataset.

ii) We built our own Convolution Neural Network (CNN) model (We named it OurModel) and then try to make predictions using it. The aim was atleast get 90% accuracy with our model.

We trained/tested the CNN models on a Google COLAB machine with 4 logical cores (Intel(R) Xeon(R) CPU @ 2.20GHz) and 25 GB RAM . We used a single GPU : Tesla P100-PCIE-16GB

We used Python standard Deep learning library Pytorch for developing and training the CNN models.The loss was calculated using Cross Entropy. We used Adam Optimizer for the Gradient descent. We initially also experimented with Stochastic Gradient Descent however but decided to go with Adam Optimizer as it gave better accuracy.

2.1. Data Preparation

The X-ray images from the data-set are in gray-scale and their dimensions are 299x299.

As part of data preparation we performed a 'Gray-scale to RGB' conversion upon them. We also resized the images

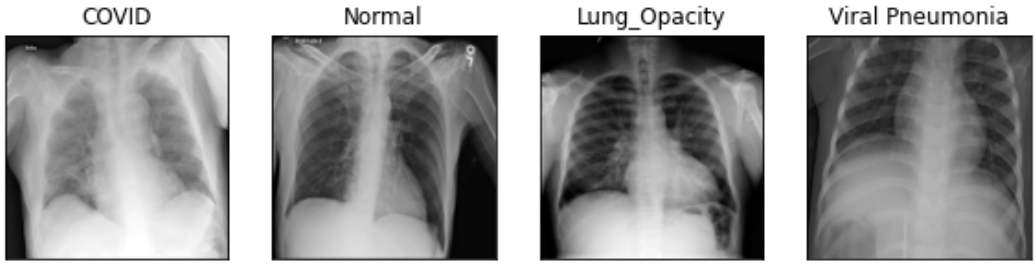


Figure 1: Sample X-ray Images

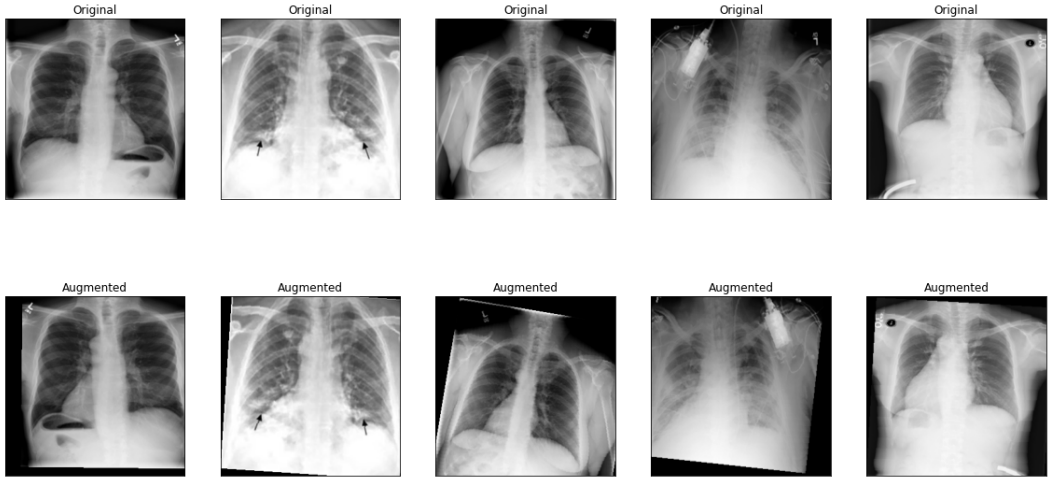


Figure 2: Sample Augmented X-ray Images

according to the model being used. We used an image size 227x227 for Squeezenet and 224x224 for all other models including our custom model.

We split up the data into training and test data in the ratio 80:20. The training set was further split up into final training and validation data-sets in the ratio of 90:10. Table[1] gives details of data distribution across labels for training, validation and testing data-sets for both augmentation and non-augmentation scenarios

In order to improve accuracy and have a bigger training data-set we augmented the final training data . Data Augmentation was done by using various techniques like image rotation,translation,vertical and horizontal flip. These techniques were applied to images randomly. Table [2] gives the augmentation details

Figure [2] show the results of changes that happen to the images on being subjected to these augmentation techniques:

2.2. Training and Testing

We trained and validated using the training and validation data-sets respectively for a number of iterations. For each iteration before training we shuffled the training data. Next the training data set was divided into multiple batches and we trained with these batches. We experimented with multiple batch sizes but the results were best with batch size of 32 for each of our models.

We kept on tracking the performance of the model and saved the model parameters with the highest accuracy for validation data-set as the best model. Once the training was done we tested the testing data-set against the best model . For this project we have considered the accuracy of predictions using testing data as final accuracy of the model and it is a key indicator of a given model's performance.

The whole approach is depicted in Figure[3]

The following algorithm has been used for data preparation, training and testing:

Dataset	Non-Augmented			Augmented		
Class	**Train**	**Validation**	**Testing**	**Train**	**Validation**	**Testing**
COVID	2627	278	711	5254	278	771
Normal	7304	839	2049	14608	839	2049
Lung Opacity	4339	469	1024	8678	469	1024
Viral Pneumonia	968	108	269	1936	108	269
Total	15238	1694	4053	30476	1694	4053

Table 1: Class wise Data set distribution per epoch

Technique	Details
Rotation	Rotation Range: [**-10°** to **10°**]
Vertical Translation	Vertical translation in the range[**-10%** to **10%** of image height]
Horizontal Translation	Horizontal translation in the range[**-10%** to **10%** of image width]
Horizontal Flip	Flipping image about its horizontal axis (with **70%** probability)
Vertical Flip	Flipping image about its vertical axis (with **20%** probability)

Table 2: Data Augmentation Techniques and Details

Algorithm 1 Data-Split,Preparation,Training and Testing Algorithm

Input: X-ray images
Output: Classified X-ray images
for epochs=1,N **do**
  RGB conversion and resize of X-ray images;
  split data into train,validation and test datasets;
  shuffle training dataset
  for each training-data batch **do**
    perform data augmentation
    perform forward pass and get predictions
    compute loss and gradient of loss
    update model parameters
  end for
  perform validation using validation dataset
  save the best performing model and its parameters
end forperform testing using test data-set

2.3. Issues Anticipated

We anticipated issues with the memory consumption by the models. Some of models like Densenet and OurModel consumed as high as 24 GB of memory. However since our COLAB engine supported 25 GB of RAM we were able to handle this. Had we needed more memory we would have opted for a Google Colab Pro.

We also anticipated issues with data augmentation. Sometimes data Augmentation on medical images can reduce the model performance. We didn't however face this for any of the models except Squeezenet. However we noticed that data augmentation performed on each image (effective doubling the data set) outperformed the scenario where we performed data augmentation on only COVID, Lung Opacity and Viral Pneumonia in order to address Class Imbalance. As specified above Normal X-rays alone made up almost 47% of the original dataset. In other words dataset with greater class imbalance outperformed dataset with lesser class imbalance.

We also anticipated challenges with hyper-parameter tuning given we had to work with 6 models. We had to perform several iterations with various hyper-parameter values before arriving at highest accuracy values.

3. Architecture of Custom Model

3.1. Custom Model Design

Our custom model consists of 8 convolution layers, 8 ReLU layers, 4 MaxPool layers, 4 BatchNorm layers and 2 Fully Connected Linear layers and 1 AdaptiveAvgPool layer as shown in Figure [4] .

Input image size of (224x224) with 3 channels (RGB) is used. Kernel size of (5x5) is used with padding of 2 and stride length 1 to scan the entire image area. Max pooling is used with 2x2 and stride of 2. Since we are doing a 4-class classification problem, the output of the last Fully Connected layer is 4.

3.2. Rationale behind Custom Model Design:

In our custom model we have chosen to use smaller 5x5 convolution in contrast to many other architectures using larger kernels like 11x11. It is more beneficial for deeper

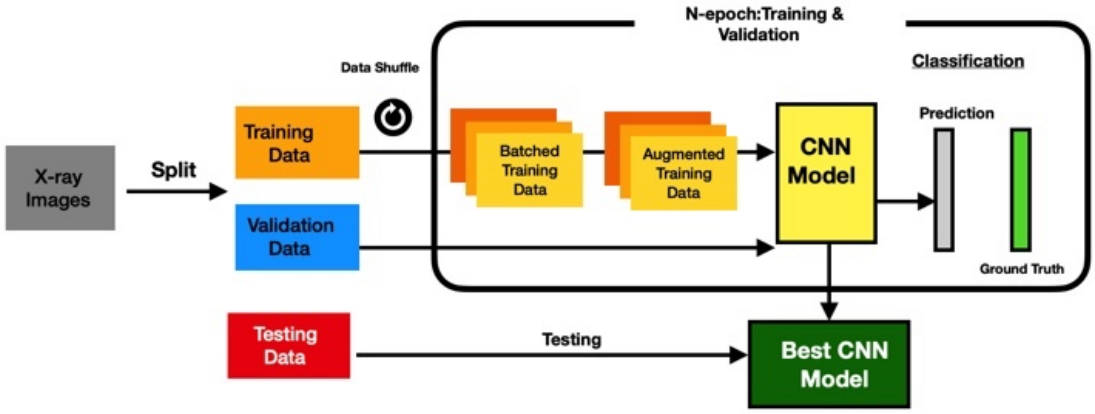


Figure 3: Training-Testing Workflow

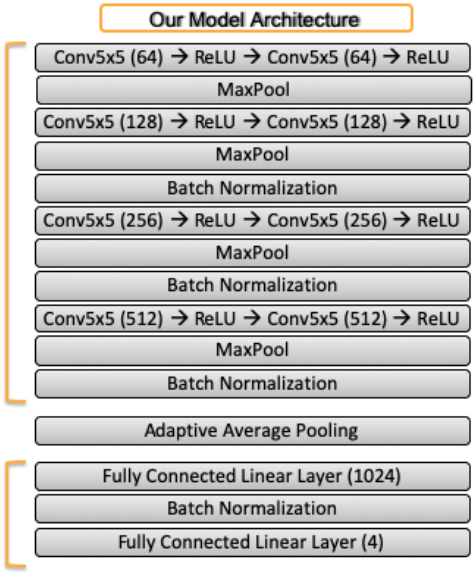


Figure 4: Our Model Architecture

networks to use smaller convolutions (i.e., smaller kernels). This is because larger convolutions like 11x11 will need way more hyper-parameters which eventually reduces the efficiency of deep networks.

We have used Batch Normalization after ReLu layer as it is believed to perform consistently better than using Batch Normalization before ReLU layer.

4. Experiments and Results

4.1. Measuring Success

Success is measured in two ways. i) Overall accuracy for a model against the testing dataset. Our objective was to get 94% or greater accuracy for all pre-trained models and atleast 90% accuracy for our custom model. ii) Accurately predict the COVID class with atleast 95% accuracy.

We were successful in making the models perform better than the accpetable minimum levels described above.

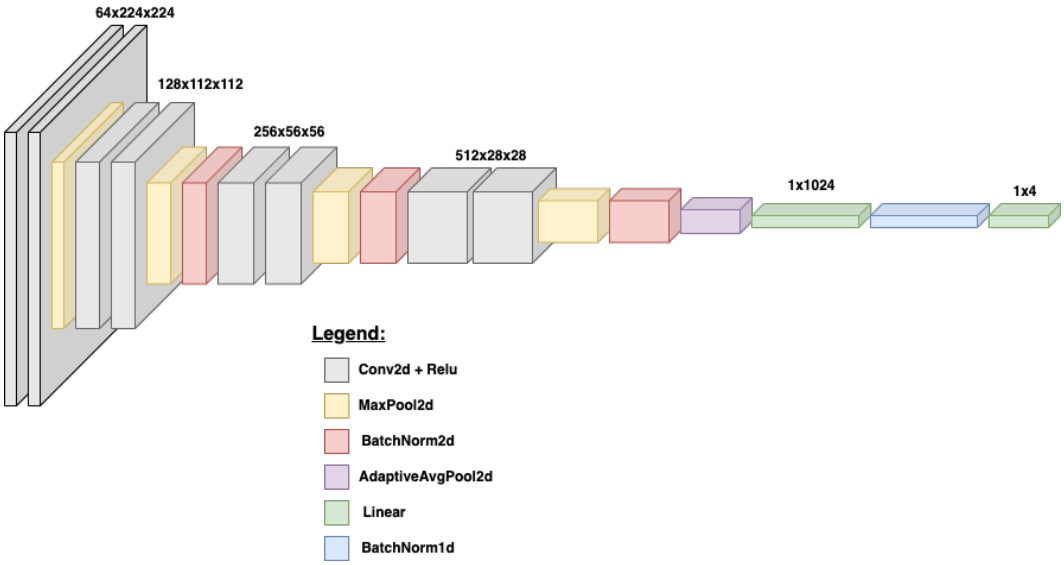


Figure 5: Our Model Architecture

4.2. Experiment : Experiments with Data Augmentation Techniques

We tried multiple data augmentation techniques before we settled on the ones we are using. One of the techniques we tried and decided not to use was Random Crop. Random crop is a powerful data augmentation technique for most image processing but in our case it resulted in reduced accuracy. This is because we are mostly looking at only certain areas of the image and therefore unrelated parts which are obtained as part of random crop add no value.

We also experimented with 5% horizontal and vertical translation also before got better performance with 10% translation hence decided to go with 10% translation.

4.3. Experiment : Hyper-parameter tuning

We ran our experiments on four different existing models and our custom model to compare and contrast the performance in each of the settings. Our yardstick for judging the performance on any model was the accuracy of model against the test dataset.

Hyper-parameter tuning was first performed on each of these pre-trained models with image augmentation and following experiments were ran on 4 class classification scheme.

Table[3] shows the list of tuned hyper-parameters for each of the pre-trained models with image augmentation.

4.4. Experiment : Pre-trained model with and without image augmentation

We tried to analyze the impact of augmentation on model performance. We observed that all existing state of art models show only a marginal improvement in terms of accuracy when fed augmented data. However the performance of our custom model deteriorates marginally when trained on augmented data.

Table[4] shows the performance metrics for all the models for our four-class classification problem with and without image augmentation.

4.5. Experiment : Models with and without Pre-training.

Pre-trained models show considerably better prediction accuracy compared to models trained from scratch. This is the expected behavior.

Table[5] shows the performance metrics for all the models for our four-class classification problem with and without model pre-training.

4.6. Results

As we can notice from Table[4] best overall accuracy is obtained with DensetNet201 (96.55%) for augmented dataset followed closely by MobileNet_V2 (96.19%). Our model returned an accuracy of 94.31%.

For the non-augmented (original) dataset also, DenseNet201 performs the best with 95.22% accuracy while OurModel return an accuracy of 93.88%.

Pre-trained model with Image Augmentation				
Models	**Batch Size**	**Epoch**	**Learning Rate**	**Learning Rate Schedule**
Resnet18	32	10	1e-4	[4]
VGG19	32	5	1e-4	[4,7,9]
Squeezenet	32	9	1e-4	[5,8]
Mobilenet_V2	32	10	1e-4	[6,8]
DenseNet201	32	4	1e-4	[2,6]
OurModel	32	7	1e-4	[4,8]

Table 3: Final Hyper-Parameter Values After Model Tuning

	Non-Augmented			**Augmented**		
Models	**Accuracy**	**Precision**	**Sensitivity**	**Accuracy**	**Precision**	**Sensitivity**
Resnet18	96.01	96.78	96.55	96.33	96.99	96.68
VGG19	95.69	96.59	95.62	96.10	96.94	96.23
Squeezenet	94.32	95.58	94.49	94.10	95.00	94.26
Mobilenet_V2	95.63	96.57	96.12	96.19	97.07	96.62
DenseNet201	96.22	97.13	96.21	96.55	97.45	96.45
OurModel	93.88	94.85	94.73	94.31	95.25	94.42

Table 4: Experiment 1 - Performance Metrics for Four-Class Classification With and Without Image Augmentation on Pre-Trained weights

Figure [6] shows a comparison of the highest training,validation and testing accuracy scores for each model.

When it comes medical images; augmentation can sometimes lead to poorer performance of the model .However as per our observations augmented dataset performs slightly better than the original non-augmented dataset for all the models with the only exception of Squeezenet. Squeezenet has an accuracy of 94.32% for non-augmented data which reduces to 94.10% for augmented data resulting in a slight deterioration 0.22%. In contrast to this MobileNet_V2 shows the highest improvement of 0.56%. Our custom model shows an improvement of 0.27%.

When it comes to the accuracy for COVID class we got an accuracy of 99.16% for Densenet201 followed closely by MobileNet_V2 with 98.31 . Squeezenet had the lowest accuracy with 95.78% while for our custom model had an accuracy of 97.47% Figure [7] depicts the Confusion Matrices for all the 6 models.

Figure [8] shows the accuracy of predicting COVID cases for each model.

5. Visualization and Analysis

5.1. Grad-Cam Analysis on Feature Identification.

Selvaraju R.,Cogswell M.,*et al*. in their 2019 paper [7] proposed a technique for producing "visual explanations" for decisions made from a large class of CNN-based models therefore making the models more transparent. The technique is know as Gradient-weighted Class Activation Mapping or simply **Grad-CAM**.

The Grad-CAM technique through the explainable view of deep learning models, helps us to clearly interpret the model's decision. The deep learning model needs to be interpretable and these visualizations give us confidence in our results. We have created the Grad-Cam visualizations on the X-ray images for every class and for each model. We at random choose 2 images belonging to each class from the test data set for the visualization purposes. The models were trained using augmented dataset.

Figure [9(a)] shows the correctly predicted X-Ray images and their corresponding GradCam visualization when using DenseNet201. Top images are the actual X-ray images while the bottom ones is GradCam visualization. Actual class of every image is mentioned on left side of label and right side of the label is the predicted class and on the bottom we have created Grad-Cam images. As we can see, except for the "Normal" classified image, the deep learning model is looking into the lungs of the X-Rays. Based on the X-Ray texture, the model can classify them into Covid, Viral Pneumonia or Lung Opacity.

In the Figure [9(b)], the top X-Ray images are incorrectly classified. This is for Squeezenet. We can clearly see that in case of miss-classification the model is not looking at right places and thus is miss-classifying.

We chose to show DenseNet GradCam visualizations for correctly classified cases because it was the best performing

	Pre-trained			Without Pre-training		
Models	**Accuracy**	**Precision**	**Sensitivity**	**Accuracy**	**Precision**	**Sensitivity**
Resnet18	96.33	96.99	96.68	93.26	94.24	93.76
VGG19	96.10	96.94	96.23	91.50	92.84	91.62
Squeezenet	94.10	95.00	94.26	85.63	87.00	85.24
Mobilenet_V2	96.19	97.07	96.62	88.80	89.98	89.06
DenseNet201	96.55	97.45	96.45	92.88	94.32	92.72
OurModel	NA	NA	NA	NA	NA	NA

Table 5: Experiment 2 - Performance Metrics for Four-Class Classification With and Without Model Pre-Training on Augumented Images

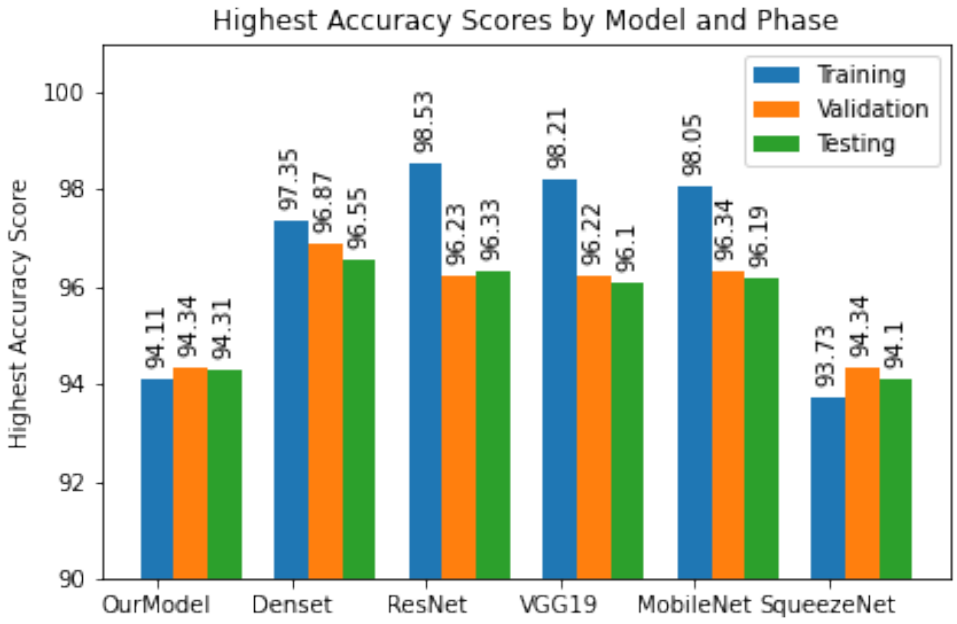


Figure 6: Training,Validation and Testing Accuracy

model and we chose Squeezenet GradCam visualizations for incorrectly classified cases because it was the worst performing model.

Figures [9(c)] and [9(d)] respectively show the GradCam visualizations for images correctly and incorrectly predicted by OurModel.

5.2. Activation of Intermediate Layers.

We also tried to capture the activation of intermediate layers. Figure [10] shows the intermediate activation for DenseNet201 and OurModel. In both cases this is for last layer before fully connected layer.

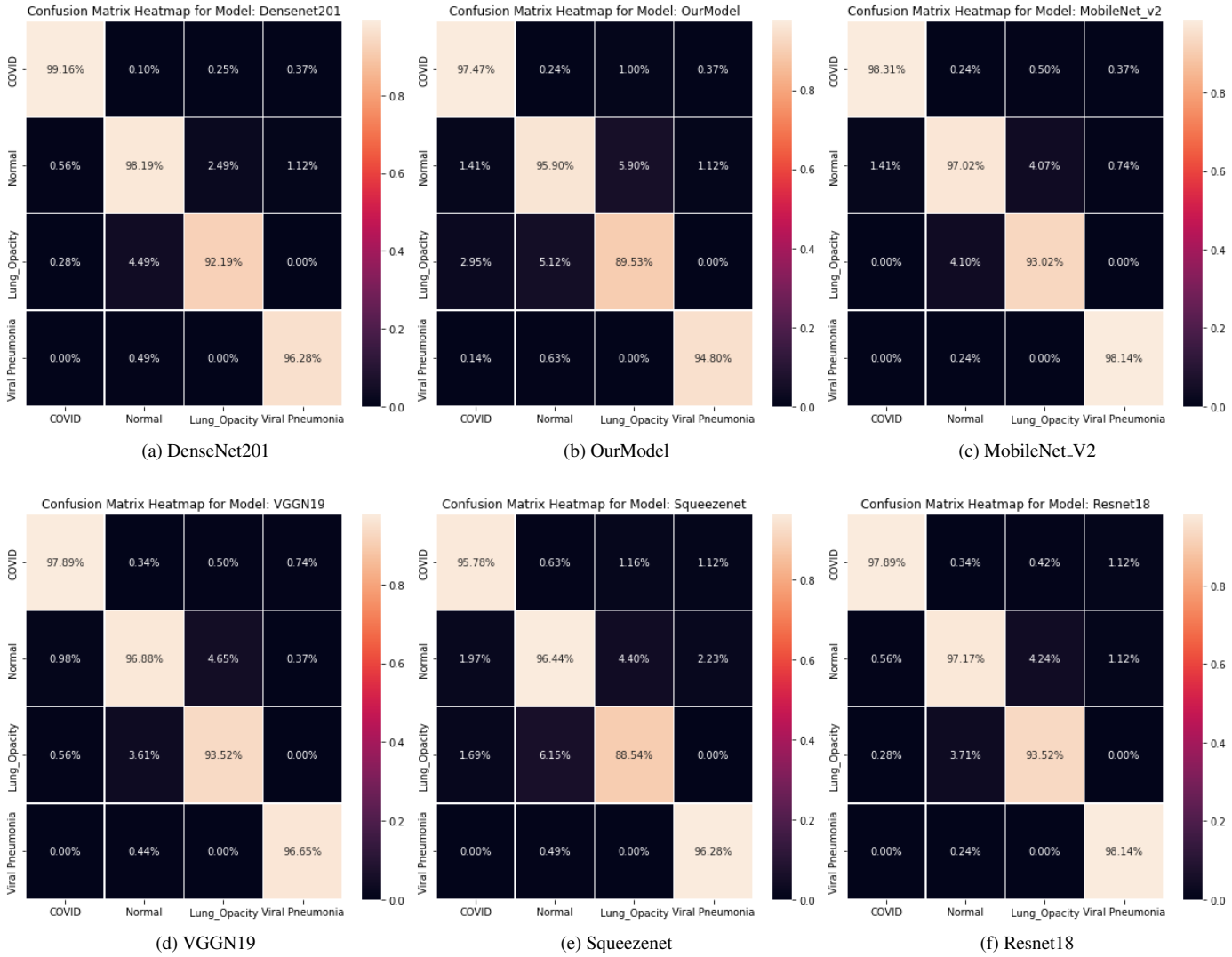
Further Figure [11] shows images of the more highly activated channels on different layers of OurModel. Figure [11(a)] is the original image, Figure [11(b)] shows the channel 55 of the first convolution layer and Figure [11(c)] shows channel 203 of last convolution layer.Figure [11(d)] is the GradCam visualization. This X-ray is for a positive COVID-19 case and OurModel correctly diagnosed it.

5.3. Accuracy and Loss Curves

Accuracy and Loss curves for all the 6 models are available here : Figures [12] and [13].

6. Code Repository

Our code can be found here: Github Repo



(a) DenseNet201 (b) OurModel (c) MobileNet_V2

(d) VGGN19 (e) Squeezenet (f) Resnet18

Figure 7: Confusion Matrix: DenseNet and OurModel.

7. Work Division

The details of individual contributions to the project are provided in Table [6]

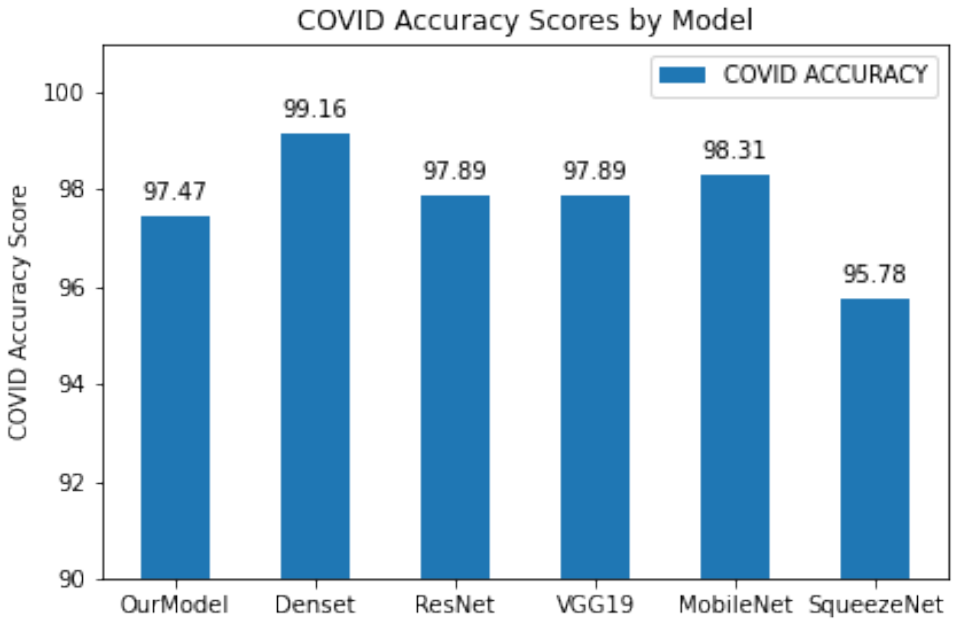
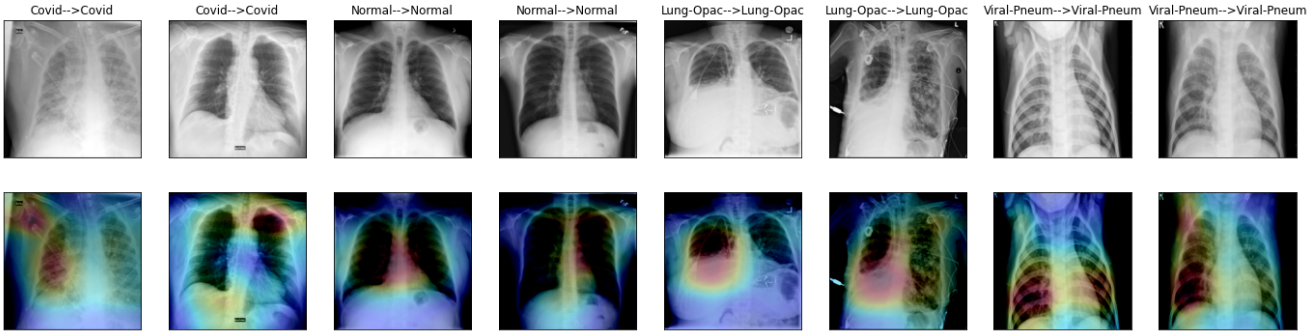


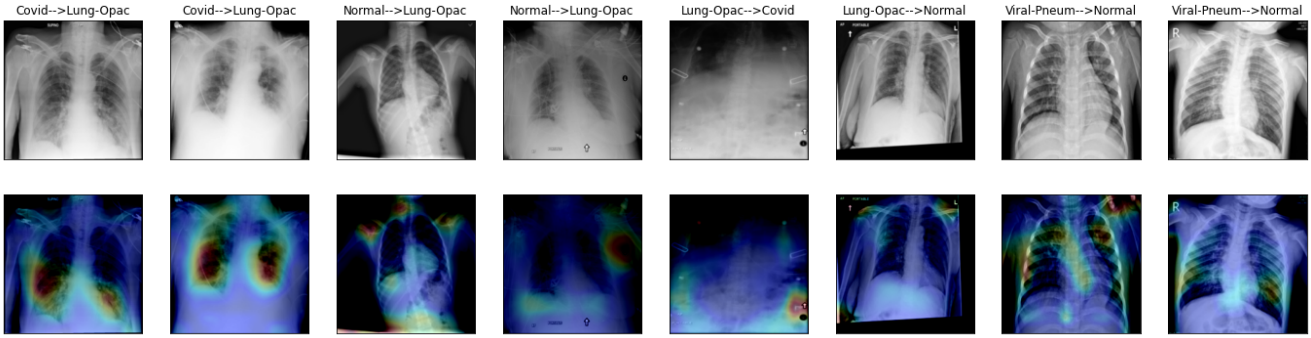
Figure 8: Model Accuracy for COVID predictions

Student Name	Contributed Aspects	Details
Ruchit Khushu	Implementation and Analysis	Conceptualization, Data Preparation and Augmentation, Architecture and Coding of OurModel ,Coding for DenseNet201, and MobileNet_v2 Models, Hyperparameter tuning, testing and debugging, writing project report,Visualization
Siddharth Kapoor	Implementation and Analysis	Conceptualization, Architecture and coding of OurModel, VGGN, and Squeezenet_v2 Models, Hyperparameter tuning, testing, writing project report, GradCam coding and Visualization.
Sujit Biswas	Implementation and Analysis	Analysis, coding and hyper-parameter tuning of Resnet18 and VGG19. Analysis and coding of Custom Model. Writing project report and coding visualization curves, metrics, train, test logic, debugging and testing.
Pankaj Chauhan	Implementation and Analysis	Conceptualization, Data Preparation and Augmentation, coding of OurModel, MobileNet_v2 Models, hyperparameter tuning, testing, writing project report, GradCam Coding visualization, Code testing

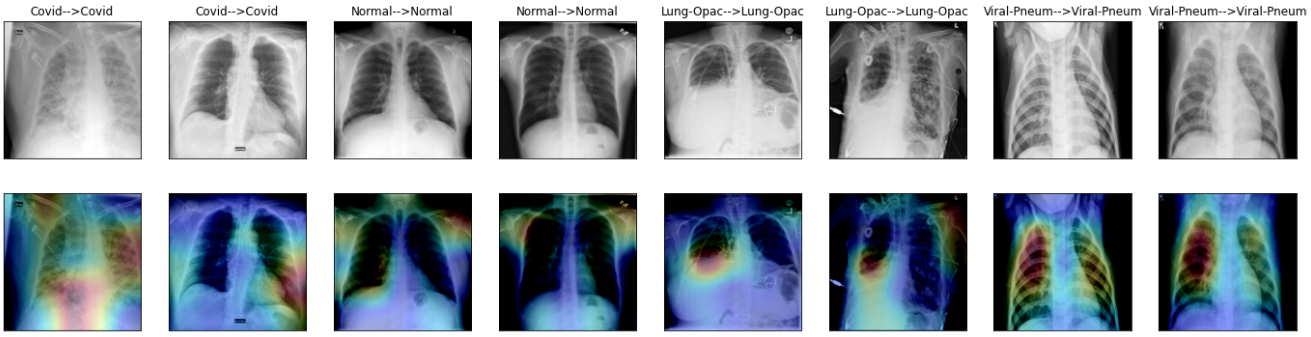
Table 6: Contributions of team members.



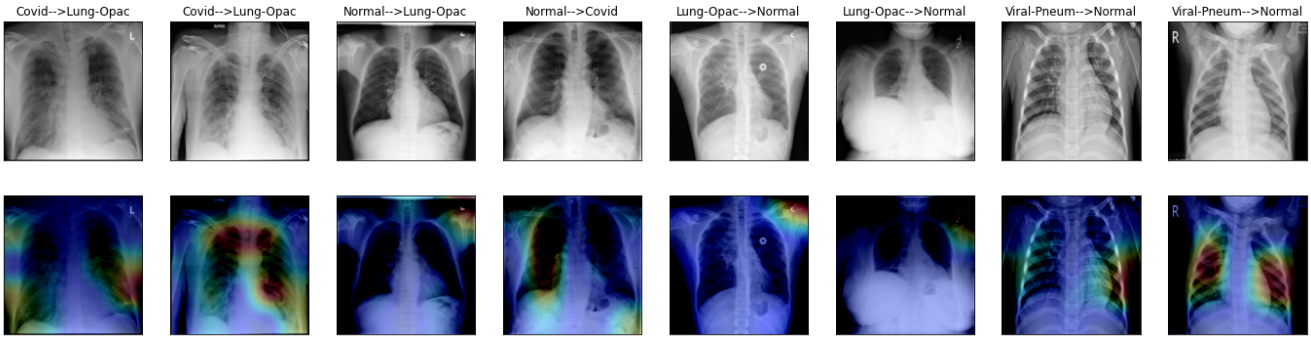
(a) DenseNet201 Correct Classification



(b) Squeeznet Incorrect Classification

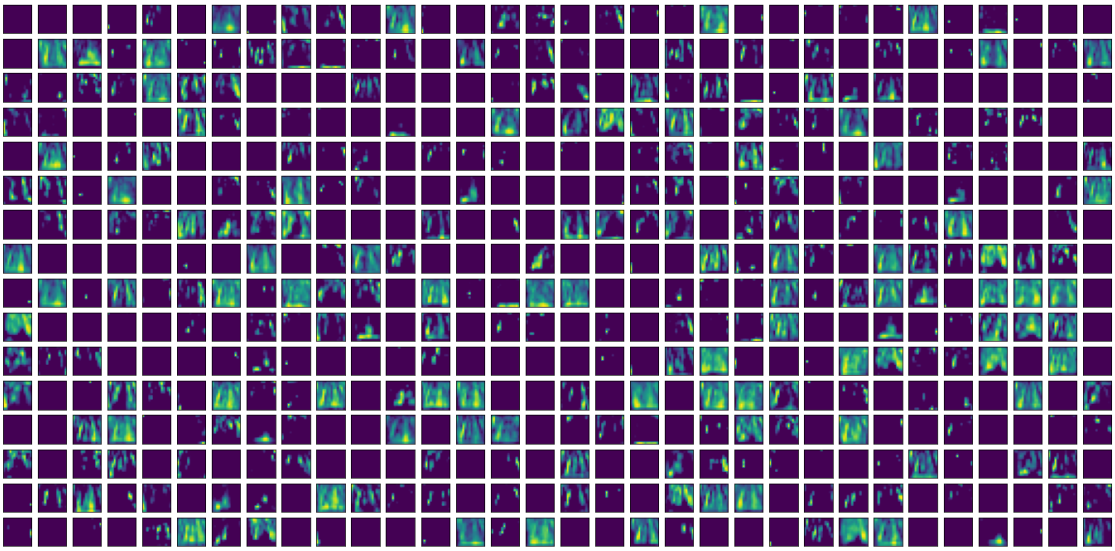


(c) Our Model Correct Classification

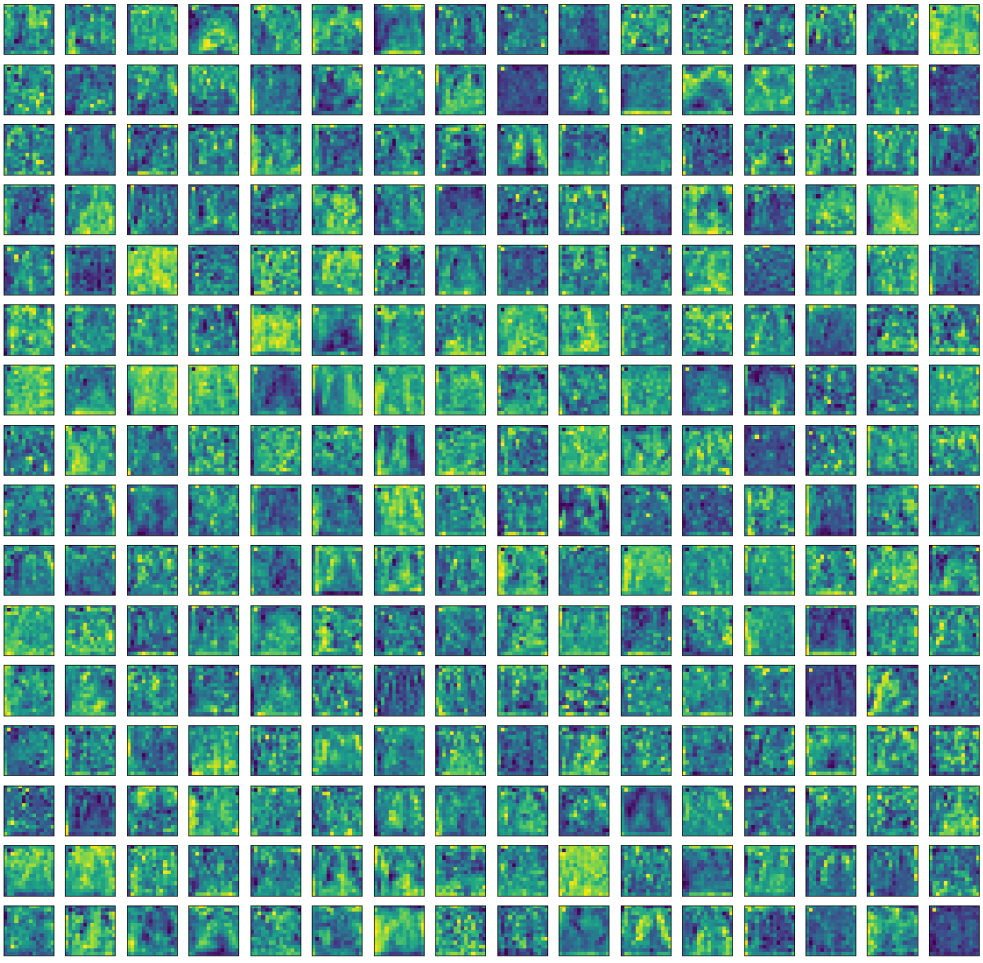


(d) OurModel Incorrect Classification

Figure 9: Gradcam Visualizations

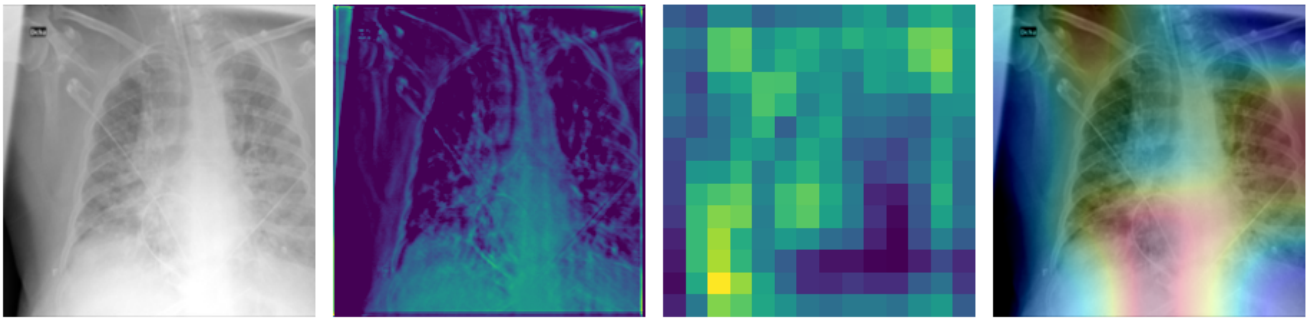


(a) Our Model Intermediate Activation



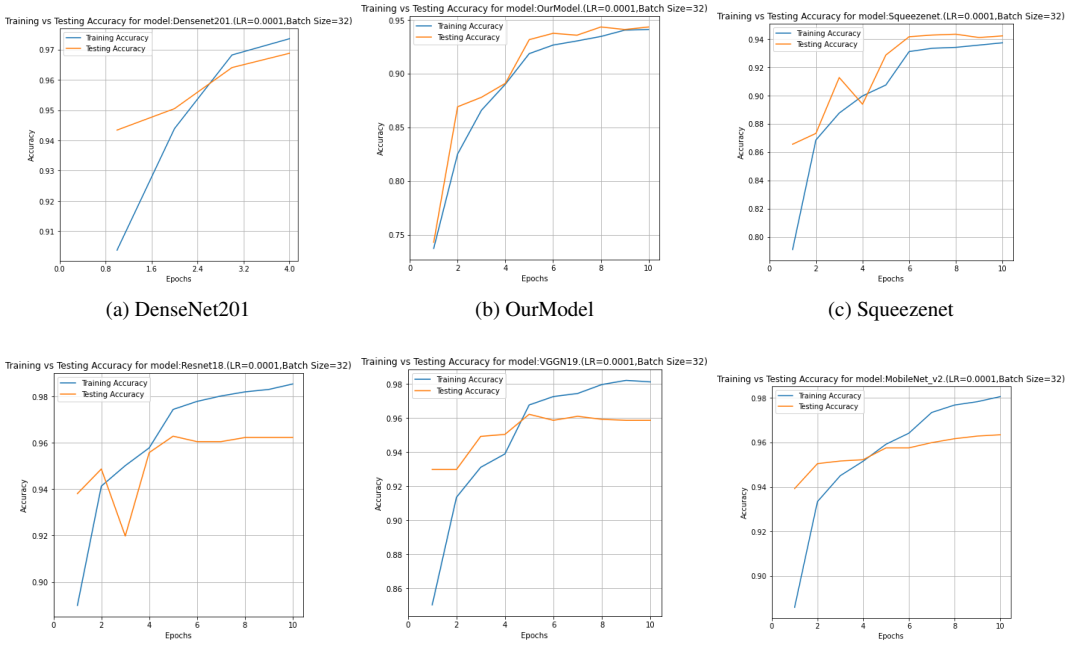
(b) DenseNet201 Intermediate Activation

Figure 10: Intermediate Activation



(a) Original Image (b) Channel 55 of first conv. layer (c) Channel 203 of last conv. layer (d) GradCam Visualization

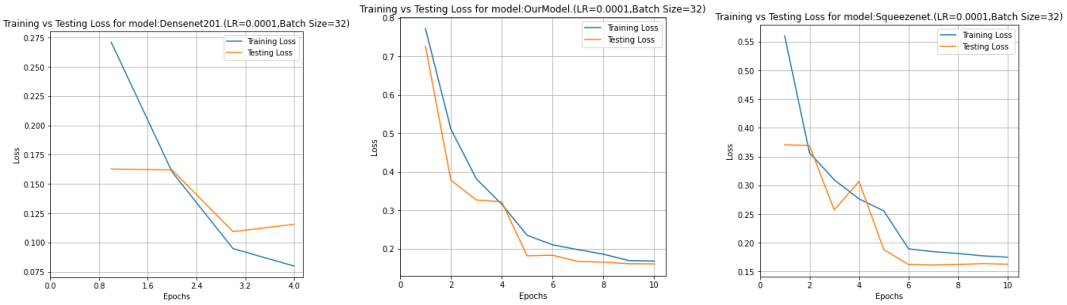
Figure 11: Intermediate Activation: Individual channel for OurModel



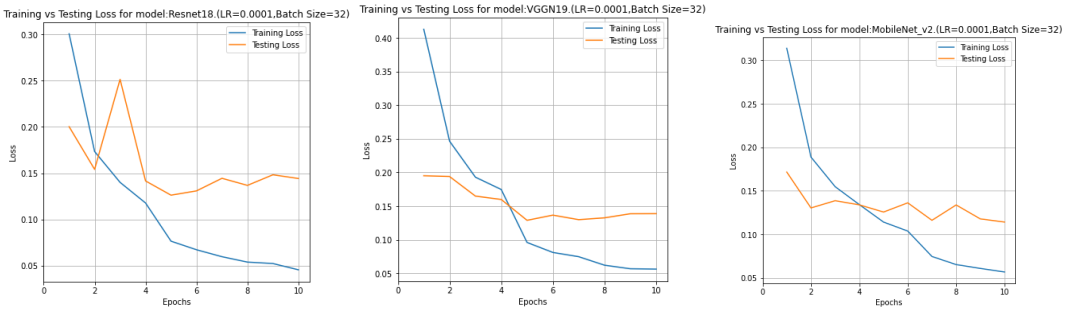
(a) DenseNet201 (b) OurModel (c) Squeezenet

(d) Resnet18 (e) VGG19 (f) MobileNet$_v$2

Figure 12: Accuracy curves for models



(a) DenseNet201 (b) OurModel (c) Squeezenet



(d) Resnet18 (e) VGG19 (f) MobileNet$_v$2

Figure 13: Loss curves for all models

References

[1] M.E.H. Chowdhury, T. Rahman, A. Khandakar *et al*. *IEEE*, Can AI help in screening Viral and COVID-19 pneumonia?

[2] Rahman, T., Khandakar, A., Qiblawey, Y.*et al*. *compbiomed 2020*, Exploring the Effect of Image Enhancement Techniques on COVID-19 Detection using Chest X-ray Images.

[3] L. Wang and A. Wong *arXiv:2003.09871, 2020*,COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest X-ray images.

[4] A. S. Joaquin, *Using Deep Learning to Detect Pneumonia Caused by NCOV-19 From X-Ray Images*

[5] Stefanos Karakanis, Georgios Leontidis *compbiomed 2020*,Lightweight deep learning models for detecting COVID-19 from chest X-ray images

[6] Eduardo Luz, Pedro Silva, Rodrigo Silva, Ludmila Silva*et al*. *arXiv:2004.05717*,Towards an Effective and Efficient Deep Learning Model for COVID-19 Patterns Detection in X-ray Images

[7] Ramprasaath R. Selvaraju,Michael Cogswell*et al*. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*arXiv:1610.02391*,Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

[8] Reddit:*Batch Normalization before or after ReLU*

Appendices

A. DataSet

https://www.kaggle.com/tawsifurrahman/covid19-radiography-database