

Final Project Report

Pneumonia Detection using CNN and Transfer

Learning on Chest X-ray Image Dataset



Instructor: Professor Hong Man

Subject: AAI 646-B Data Acquisition and Processing I

Students: Abhishek Amberkar (#10469370), Pranavi Vashishtha
(#10459638) and Shreeya Kokate (#20005256)

DECLARATION

We Abhishek Amberkar, Pranavi Vashishtha and Shreeya Kokate Master of Science students hereby declare that project titled “Pneumonia Detection using CNN and Transfer Learning on Chest X-ray Image Dataset” which is submitted by us to ECE Department, Stevens Institute of Technology, Hoboken, New Jersey, in partial fulfilment of requirement for the completion of the course – ‘AAI 646-B Pattern Recognition and Classification’ and we have submitted this keeping in mind Stevens code and conduct .

12th May, 2022

Abhishek Amberkar, Pranavi Vashishtha and Shreeya Kokate

CERTIFICATE

On the basis of declaration submitted by Abhishek Amberkar, Pranavi Vashishtha and Shreeya Kokate, students of ECE and Computer Science Dept. at Stevens Institute of Technology, we hereby certify that the project entitled “Pneumonia Detection using CNN and Transfer Learning on Chest X-ray Image Dataset” which is submitted by us to ECE Department, Stevens Institute of Technology, Hoboken, New Jersey, in partial fulfilment of the requirement for the completion of this course, is an original contribution with existing knowledge and faithful record of work carried out by them under the guidance and supervision of the professor and has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

12th May, 2022

Abhishek Amberkar, Pranavi Vashishtha and Shreeya Kokate

ACKNOWLEDGEMENTS

This project has been a great experience of learning. We would sincerely like to express our gratitude to Prof. (Dr.) Hong Man, Stevens Institute of Technology, for his time, guidance, support and wealth of information that he shared with us to help us complete this research. We would also like to thank my friends and batch-mates, who have helped me throughout the process of this study.

Abhishek Amberkar, Pranavi Vashishtha and Shreeya Kokate

TABLE OF CONTENTS

Declaration	i
Certificate	ii
Acknowledgments	iii
Table of contents	iv
List of figures	vi
Abstract	viii
Introduction	7
1. Methodology	8
1.1 Workflow	8
1.2 The Dataset	8
2 Algorithms Implemented.....	11
2.1 CNN Model	11
2.2 VGG Model	12
2.3 ResNet50 Model.....	13
3 Conclusion	14

References

LIST OF FIGURES

Number	Caption	Page
1	The Model	7
2	Workflow of our approach	8
3	Pneumonia affected lungs	9
4	Normal person's lungs	10
5	Data Representation	10
6	Classifying Images	11
7	Different Layers in CNN architecture	12
8	Output of CNN model	13
9	Learning Curve of CNN model	13
10	Validation accuracy score	14
11	Test accuracy score	14
12	Summary of VGG model	15
13	Output of VGG model	16
14	Train of VGG model	16
15	Test accuracy of VGG model	16
16	With skip connection (left hand side) & Without skip connections (Right hand side)	17
17	Train of Resnet50 model	17
18	Accuracy Score of the ResNet50 model	18
19	Output of the Finetuned ResNet50 model	18
20	Accuracy Score of Finetuned ResNet50 model	19

ABSTRACT

Pneumonia is one of the most common diseases that kill people all over the world. Pneumonia can be caused by viruses, bacteria, or fungi. However, judging pneumonia solely based on chest X-rays is tricky. The goal of this study is to make pneumonia detection easier for both specialists and beginners. Using the notion of transfer learning and CNN, we propose a deep learning system for the identification of pneumonia based on the classical classification problem (to classify if the person has Pneumonia). In this method, picture features are retrieved using several neural network models that have been pre-trained classifiers for prediction. We built three distinct models – CNN, VGG and ResNet50, and compared their results. The results that we achieve are the following train accuracy – 93%, 90% and 95% respectively. These results show that ResNet50 gives the best accuracy to classify the Pneumonia chest X Ray images.

INTRODUCTION

Deep learning algorithms can now analyse and segment an image with human-level accuracy. Deep learning may play a large role in the medical industry, particularly in imaging, which is one of the most notable fields where deep learning can play a significant role. Deep learning has advanced to the point that it is now a significant element of the medical business. Deep learning can be applied to a wide range of applications, including the detection of tumours and lesions in medical images, computer-aided diagnostics, the analysis of electronic health-related data, treatment and drug intake planning, environment recognition, and the brain-computer interface, with the goal of providing decision support for the evaluation of a person's health. The ability of neural networks to learn high level abstractions from input raw data via a general-purpose learning technique is a critical component of deep learning's effectiveness.

The interpretation of chest radiography is critical in medical diagnosis and treatment. According to the Centres for Disease Control and Prevention (CDC, Atlanta, GA, USA), over 1.7 million individuals in the United States seek hospital treatment for pneumonia each year, with around 50,000 people dying from pneumonia in 2015 [15]. In the United States, chronic obstructive pulmonary disease (COPD) is the leading cause of death, and it is expected to rise by 2020. Several researchers have recently offered several artificial intelligence (AI)-based solutions for various medical concerns. Researchers have used convolutional neural networks (CNNs) to achieve successful outcomes in a variety of medical challenges, including breast cancer diagnosis, brain tumour identification and segmentation, illness categorization in X-ray pictures, and more.

In this project, we are trying to the notion of transfer learning and CNN. we propose a deep learning system for the identification of pneumonia based on the classical classification problem (to classify if the person has Pneumonia). In this method, picture features are retrieved using several neural network models that have been pre-trained classifiers for prediction. We built three distinct models – CNN, VGG and ResNet50, and compared their results

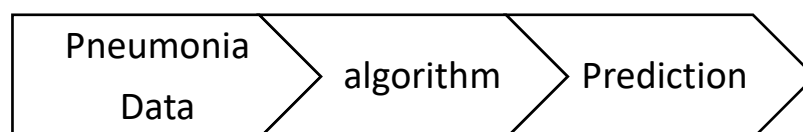


Fig 1 The Model

1. METHODOLOGY

1.1. Workflow

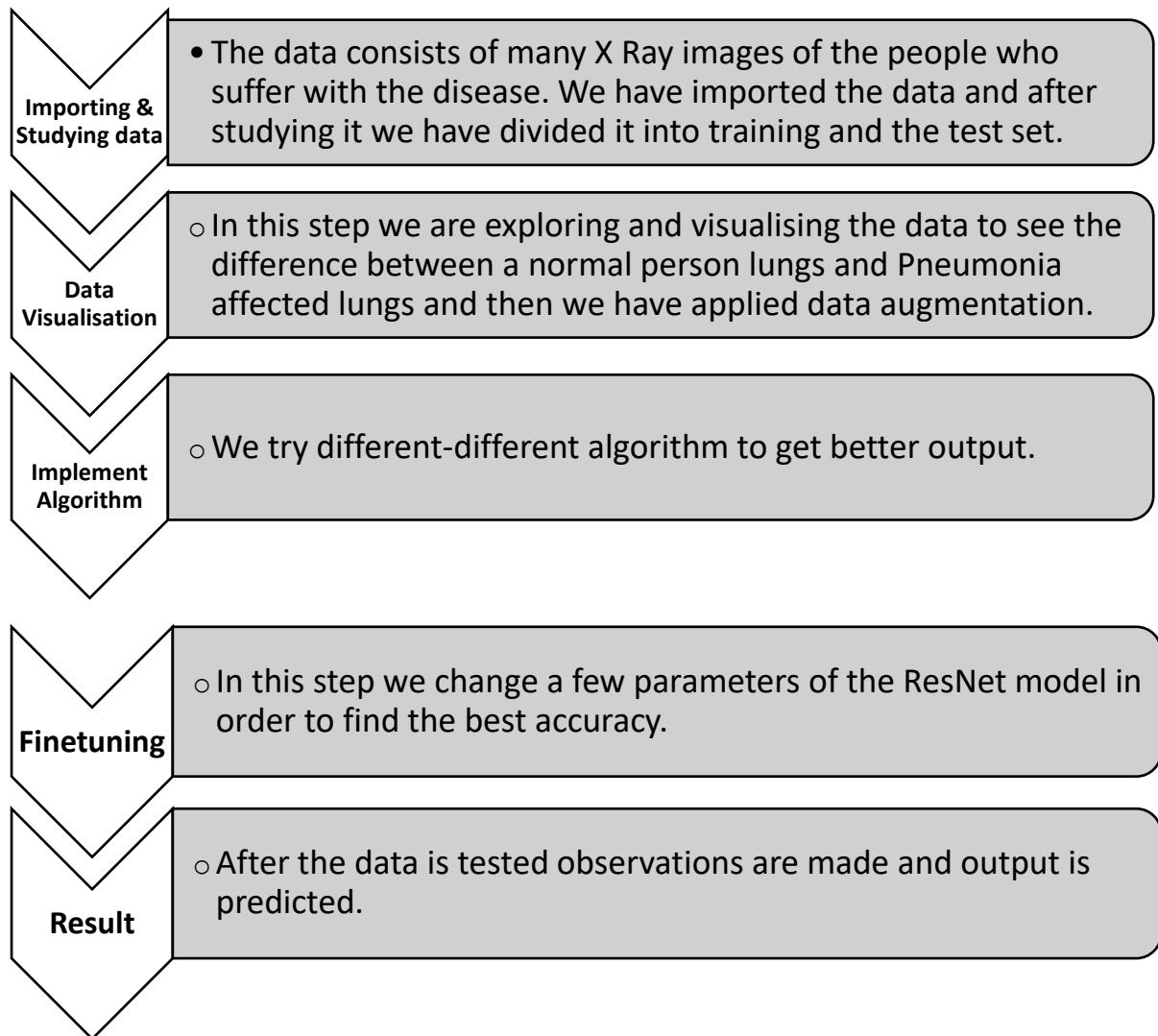


Fig 2 Workflow of our approach

1.2 The Dataset

The dataset is organized into 3 folders (train, test, Val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal). The training data consists of 5,216 chest x-ray images with 3,875 images shown to have pneumonia and 1,341 images shown to be normal. The validation data is

relatively small with only 16 images with 8 cases of pneumonia and 8 normal cases. The testing data consists of 624 images split between 390 pneumonia cases and 234 normal cases. Chest X-ray images (anterior posterior) were selected from retrospective cohorts of paediatric patients one to five years old from Guangzhou Women and Children's Medical Centre, Guangzhou. For the analysis of chest x-ray images, all chest radiographs were initially screened for quality control by removing all low-quality or unreadable scans. The diagnoses for the images were then graded by two expert physicians before being cleared for training in the AI system. In order to account for any grading errors, the evaluation set was also checked by a third expert.

Data Visualization & Pre-processing

The distributions from these datasets are a little different from each other. Both are slightly imbalanced, having more samples from the positive class (Pneumonia), with the training set being a little more imbalanced. As, the data seems imbalanced. To increase the no. of training examples, we will use data augmentation. We perform a grayscale normalization to reduce the effect of illumination's differences. Moreover the CNN converges faster on $[0..1]$ data than on $[0..255]$.

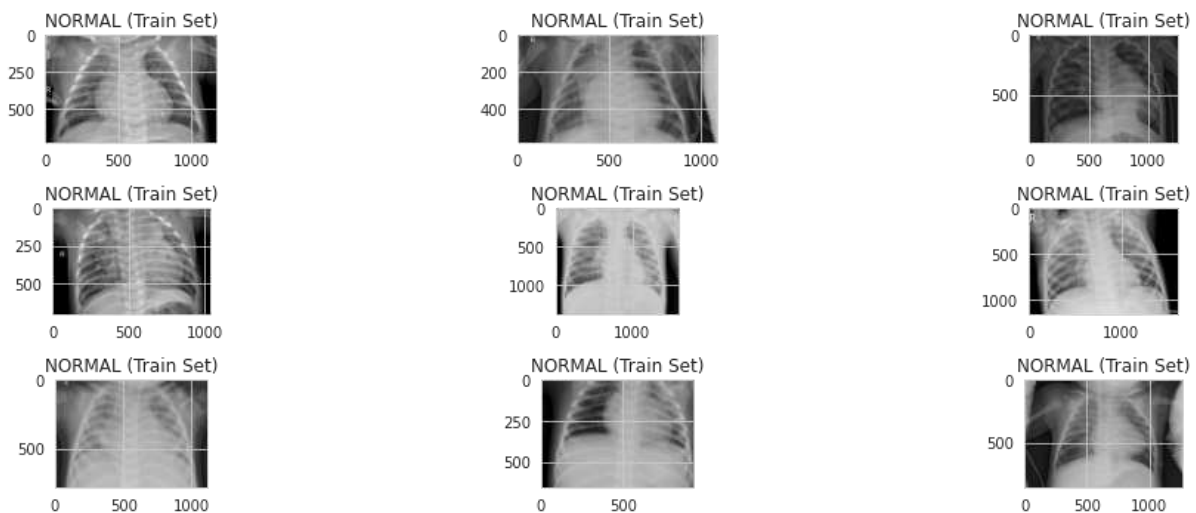


Fig 3 Pneumonia affected lungs.

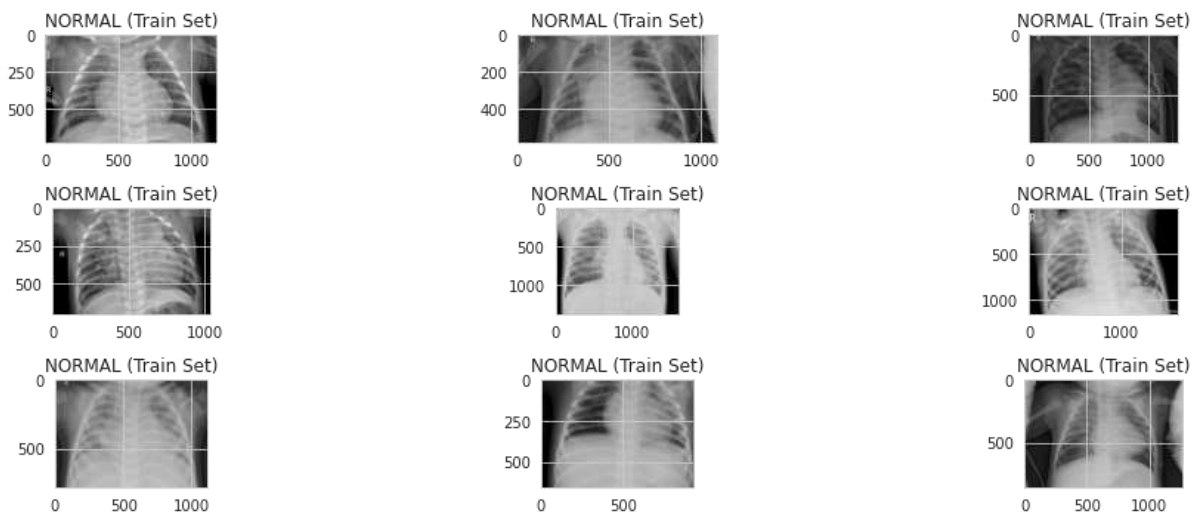


Fig 4 Normal person's lungs.

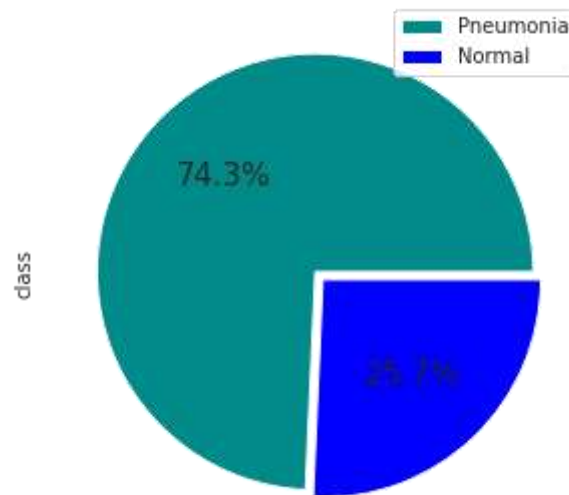


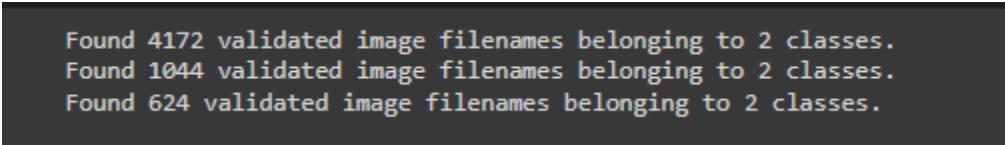
Fig 5 Data Representation

Data Augmentation

In order to avoid the overfitting problem, we artificially expanded our dataset. In order to make the existing dataset even larger. The idea is to alter the training data with small transformations to reproduce the variations. Approaches that alter the training data in ways that change the array representation while keeping the label the same are known as data augmentation techniques. Some popular augmentations people use are grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations, and much more. By applying just a

couple of these transformations to our training data, we can easily double or triple the number of training examples and create a very robust model. For the data augmentation, we:

2. Randomly rotate some training images by 30 degrees
3. Randomly Zoom by 20% some training images
4. Randomly shift images horizontally by 10% of the width
5. Randomly shift images vertically by 10% of the height
6. Randomly flip images horizontally. Once our model is ready, we fit the training dataset.



```
Found 4172 validated image filenames belonging to 2 classes.  
Found 1044 validated image filenames belonging to 2 classes.  
Found 624 validated image filenames belonging to 2 classes.
```

Fig 6 Classifying Images

2. ALGORITHM IMPLEMENTATION

2.1 CNN Model

The CNN is made up of three types of layers: convolutional layers, pooling layers, and fully-connected (FC) layers. A CNN architecture will be constructed when these layers are layered. In addition to these three layers, the dropout layer and the activation function are crucial components. Convolution Layer is the first layer utilized to extract various characteristics from the input photos. This layer performs the mathematical action of convolution between the input picture and a filter of size $M \times M$. The result contains information about the picture such as its corners and edges which is then supplied to further layers, which learn various different features from the input image. A Convolutional Layer is usually followed by a Pooling Layer. Pooling layer's major goal is to lower the size of the convolved feature map in order to reduce computational expenses. This is accomplished by reducing the connections between layers and operating independently on each feature map. There are several sorts of Pooling procedures depending on the approach utilized like Max Pooling, Average Pooling etc. The Pooling Layer is commonly used as a link between the Convolutional Layer and the FC Layer. The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the

output layer and form the last few layers of a CNN Architecture. The preceding layers' input images are flattened and supplied to the FC layer in this step. In this layer the classification process takes place. In order to overcome the problem of overfitting the training dataset Dropout Layers are used. A few neurons are dropped from the neural network during the training process resulting in reduced size of the model. Activation Functions are used to learn and approximate any kind of continuous and complex relationship between variables of the network. It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. This is how we have used the different layers in CNN Architecture –

```
model.compile(loss='binary_crossentropy', optimizer='keras.optimizers.Adam')
model.summary()
```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
conv2d (Conv2D)	(None, 222, 222, 16)	448
batch_normalization (Batch Normalization)	(None, 222, 222, 16)	64
activation (Activation)	(None, 222, 222, 16)	0
max_pooling2d (MaxPooling2D)	(None, 111, 111, 16)	0
dropout (Dropout)	(None, 111, 111, 16)	0
conv2d_1 (Conv2D)	(None, 109, 109, 32)	4640
batch_normalization_1 (Batch Normalization)	(None, 109, 109, 32)	128
activation_1 (Activation)	(None, 109, 109, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 32)	0
dropout_1 (Dropout)	(None, 54, 54, 32)	0
conv2d_2 (Conv2D)	(None, 52, 52, 64)	18496
conv2d_3 (Conv2D)	(None, 50, 50, 64)	36928
batch_normalization_2 (Batch Normalization)	(None, 50, 50, 64)	256
activation_2 (Activation)	(None, 50, 50, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 25, 25, 64)	0
dropout_2 (Dropout)	(None, 25, 25, 64)	0
flatten (Flatten)	(None, 40000)	0
dense (Dense)	(None, 64)	2560064
dropout_3 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65

=====
Total params: 2,621,089
Trainable params: 2,620,865
Non-trainable params: 224

Fig 7 Different Layers in CNN architecture

Below is the output of the CNN model that we have implemented:

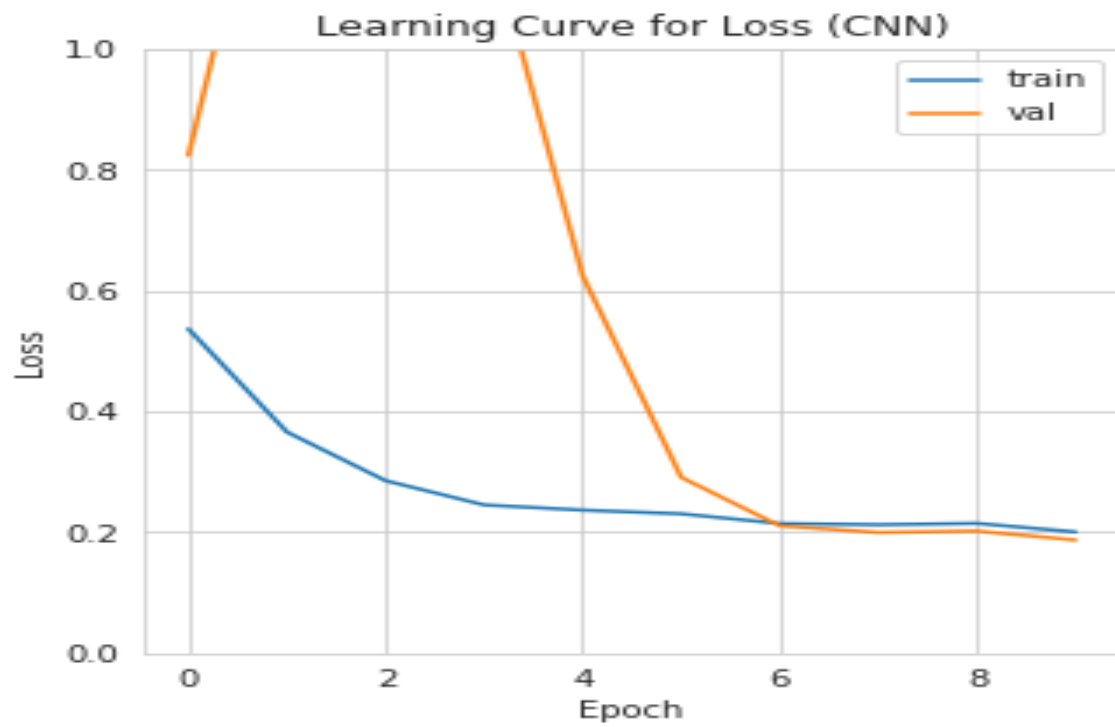


Fig 8 Output of CNN model

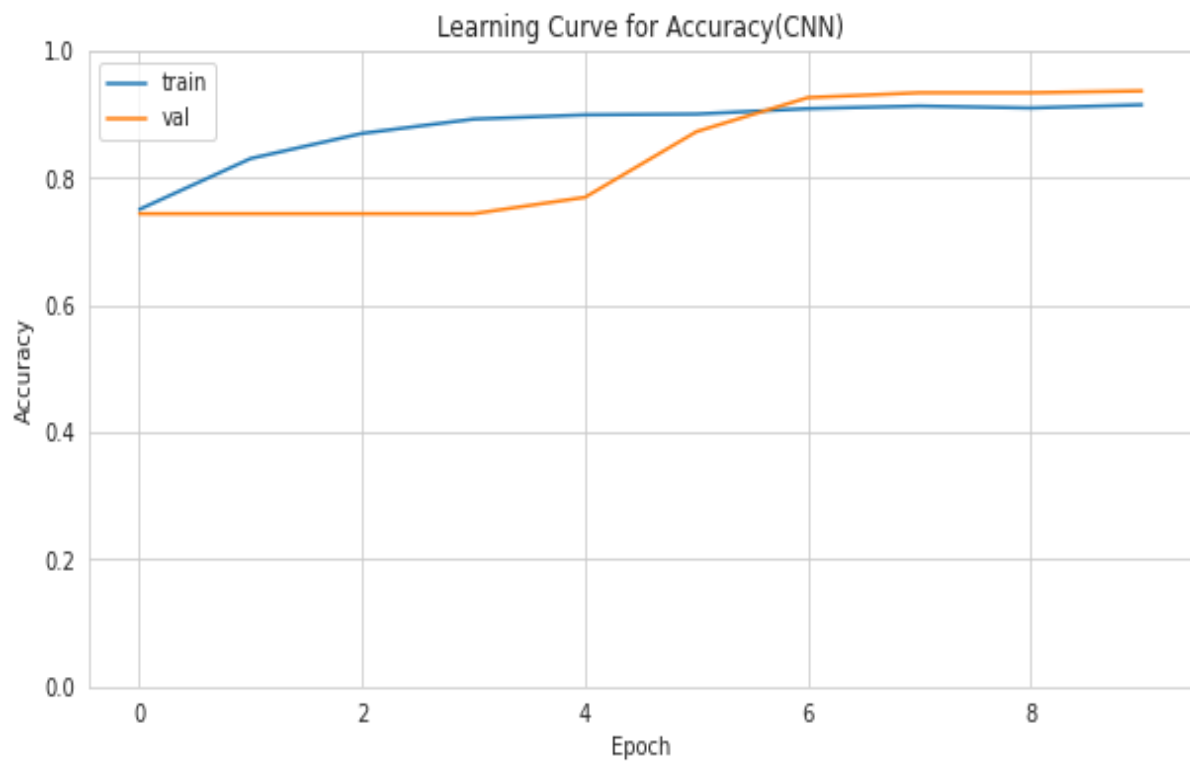


Fig 9 Learning Curve of CNN model

Accuracy Scores –

Validation Accuracy -

```
[ ] score1_val = model.evaluate(ds_val, steps = 100, verbose = 0)

print('Validation set loss:', score1_val[0])
print('Validation set accuracy:', score1_val[1])

WARNING:tensorflow:Your input ran out of data; interrupting training.
Validation set loss: 0.18721286952495575
Validation set accuracy: 0.9369024634361267
```

Fig 10 Validation accuracy score

Test Accuracy -

```
score1_test = model.evaluate(ds_test, steps = 100, verbose = 0)

print('Test loss:', score1_test[0])
print('Test accuracy:', score1_test[1])

WARNING:tensorflow:Your input ran out of data; interrupting training.
Test loss: 0.6109903454780579
Test accuracy: 0.7804487347602844
```


Fig 11 Test accuracy score

2.2 VGG Model

VGG is a deep CNN used to classify images. It contains 19 layers of CNN and fully connected layers and a final layer for softmax function. In a VGG, a fixed size of (224 * 224) RGB image is given as input to this network. In pre-processing, mean RGB value is subtracted from each pixel and computed over the whole training set. It uses kernels of (3 * 3) size with a stride size of 1 pixel, this enabled them to cover the whole notion of the image. Spatial padding is used to preserve the spatial resolution of the image. A max pooling is followed by Rectified linear unit (ReLU) to introduce non-linearity to make the model classify better and to improve computational time. The final layer is a softmax function.

We constructed our model using a pre-trained VGG 19 model. We used Adam optimizer and binary_crossentropy as loss function. In order to avoid overfitting we used callbacks.

Below was the summary of the model –

 model2.summary()

Model: "model_1"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, None, None, 3)]	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_conv4 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_conv4 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_conv4 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
dense_3 (Dense)	(None, 1)	129

=====
Total params: 20,090,177
Trainable params: 65,793
Non-trainable params: 20,024,384

Fig 12 Summary of VGG model



Fig 13 Output of VGG model

```
[ ] # Evaluating the model on Train data
score2_train = model2.evaluate(train)

print("Training Set Loss: ", score2_train[0])
print("Training Set Accuracy: ", score2_train[1])

164/164 [=====] - 370s 2s/step - loss: 0.2416 - accuracy: 0.9018
Training Set Loss: 0.24156710505485535
Training Set Accuracy: 0.9018369913101196
```

Fig 14 Train of VGG model

```
[ ] # Evaluating the model on Test data
score2_test = model2.evaluate(test)

print("Testing Set Loss: ", score2_test[0])
print("Testing Set Accuracy: ", score2_test[1])

20/20 [=====] - 34s 2s/step - loss: 0.4660 - accuracy: 0.7740
Testing Set Loss: 0.46601343154907227
Testing Set Accuracy: 0.7740384340286255
```

Fig 15 Test Accuracy of VGG model

2.2 ResNet50 Model

ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. The fundamental breakthrough with ResNet was it allowed us to train

extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients. ResNet first introduced the concept of skip connection. The diagram below illustrates skip connection. The figure on the left is stacking convolution layers together one after the other. On the right we still stack convolution layers as before but we now also add the original input to the output of the convolution block. This is called skip connection.

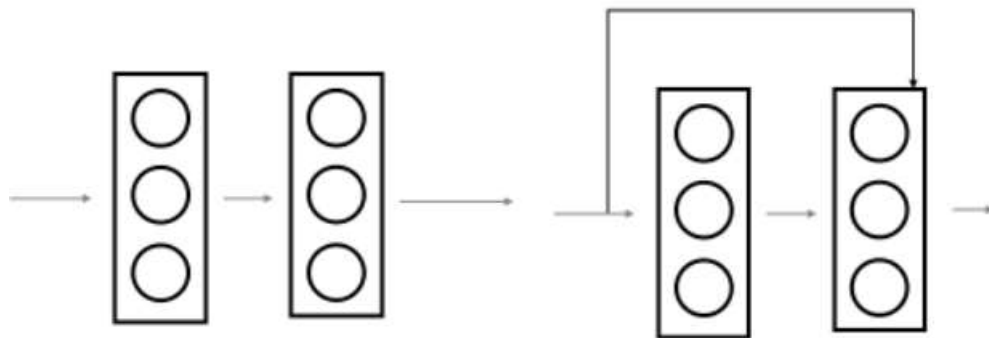


Fig 16 With skip connection (left hand side) & Without skip connections (Right hand side)

The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. We need skip connections because – 1) They mitigate the problem of vanishing gradient by allowing this alternate shortcut path for gradient to flow through. 2) They allow the model to learn an identity function which ensures that the higher layer will perform at least as good as the lower layer, and not worse.

The layers of ResNet50 are as follows: The ResNet-50 consists of five convolutional layers namely; conv1, conv2_x, conv3_x, conv4_x and conv5_x. Once the input image is loaded, it is passed through a convolutional layer with 64 filters and kernel size of 7 X 7 (conv1 layer) followed by a max pooling layer of stride length of 2 in both cases. The output of the model that we have got is as follows:

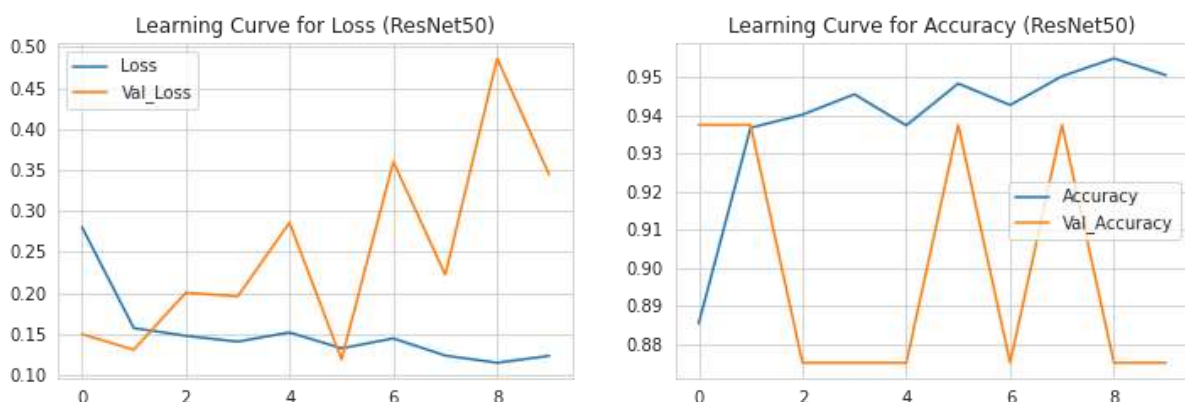


Fig 17 Train of Resnet50 model

```
# Evaluating the model on train and test
score3_train = model3.evaluate(train)
print("Train Loss: ", score3_train[0])
print("Train Accuracy: ", score3_train[1])

score3_test = model3.evaluate(test)
print("\nTest loss: ", score3_test[0])
print("Test Accuracy: ", score3_test[1])

164/164 [=====] - 196s 1s/step - loss: 0.1062 - accuracy: 0.9577
Train Loss: 0.10620119422674179
Train Accuracy: 0.9577114582061768
20/20 [=====] - 13s 630ms/step - loss: 0.3015 - accuracy: 0.8830

Test loss: 0.301524817943573
Test Accuracy: 0.8830128312110901
```

Fig 18 Accuracy Score of the ResNet50 model

Fine Tuning of the Model

To increase performance of ResNet50 even further we tried to train (or "fine-tune") the weights of the top layers of the pre-trained model alongside the training of the classifier you added. The training process forced the weights to be tuned from generic feature maps to features associated specifically with the dataset.

Following where the results of the Finetuned ResNet50 –

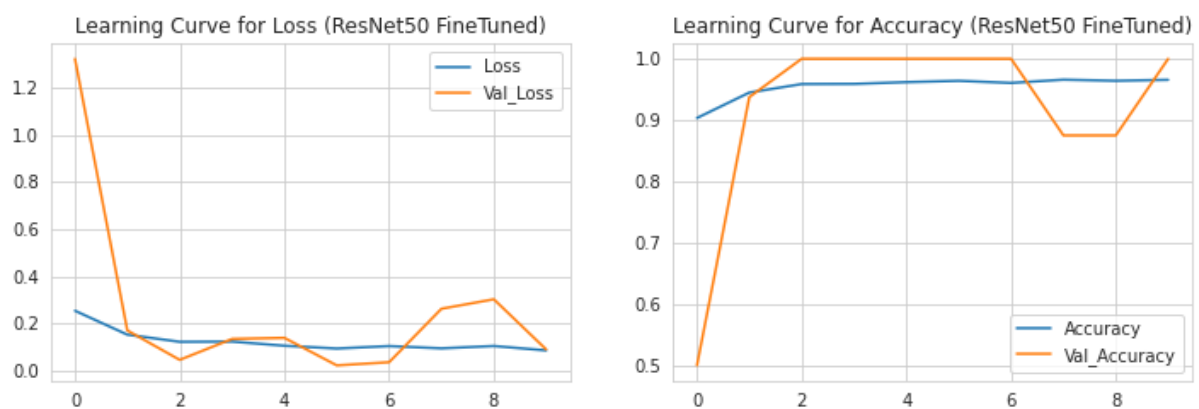


Fig 19 Output of the Finetuned ResNet50 model

```
Evaluating the Fine Tune model

[ ] # Evaluating the model on train and test
    score4_train = model3.evaluate(train)
    print("Train Loss: ", score4_train[0])
    print("Train Accuracy: ", score4_train[1])

    score4_test = model3.evaluate(test)
    print("\nTest loss: ", score4_test[0])
    print("Test Accuracy: ", score4_test[1])

164/164 [=====] - 205s 1s/step - loss: 0.4136 - accuracy: 0.8990
Train Loss: 0.41361093521118164
Train Accuracy: 0.8989667296409607
20/20 [=====] - 14s 673ms/step - loss: 0.5895 - accuracy: 0.8782

Test loss: 0.5894671082496643
Test Accuracy: 0.8782051205635071
```

Fig 20 Accuracy Score of Finetuned ResNet50 model

CONCLUSION

In this project, we tried to analyze and try to predict if a person is suffering pneumonia by classifying Chest X-Ray Image Dataset. First, we tried to construct our own basic CNN model which got us a train accuracy of about 92%. Then we explored Deep Neural Network models like VGG-19 and ResNet50. Further, we fine-tuned the ResNet50 model to achieve our best train accuracy of 95% and handled the major issue of overfitting. Working on this project was a great learning experience as well as helped us in working with Neural Networks concepts in depth.

Contribution –

Abhishek Amberkar – EDA, Custom CNN

Pranavi Vashishtha - VGG 19, Data processing

Shreeya Kokate – ResNet 50, Finetuning

REFERENCES

- [1] Chouhan, V., Singh, S. K., Khamparia, A., Gupta, D., Tiwari, P., Moreira, C., ... & De Albuquerque, V. H. C. (2020). A novel transfer learning based approach for pneumonia detection in chest X-ray images. *Applied Sciences*, 10(2), 559.
- [2] Rahman, T., Chowdhury, M. E., Khandakar, A., Islam, K. R., Islam, K. F., Mahbub, Z. B., ... & Kashem, S. (2020). Transfer learning with deep convolutional neural network (CNN) for pneumonia detection using chest X-ray. *Applied Sciences*, 10(9), 3233.
- [3] Jain, R., Nagrath, P., Kataria, G., Kaushik, V. S., & Hemanth, D. J. (2020). Pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning. *Measurement*, 165, 108046.
- [4] Hashmi, M. F., Katiyar, S., Keskar, A. G., Bokde, N. D., & Geem, Z. W. (2020). Efficient pneumonia detection in chest xray images using deep transfer learning. *Diagnostics*, 10(6), 417.
- [5]. Cha, S. M., Lee, S. S., & Ko, B. (2021). Attention-Based transfer learning for efficient pneumonia detection in chest X-ray images. *Applied Sciences*, 11(3), 1242.
- [6] Chhikara, P., Singh, P., Gupta, P., & Bhatia, T. (2020). Deep convolutional neural network with transfer learning for detecting pneumonia on chest X-rays. In *Advances in Bioinformatics, Multimedia, and Electronics Circuits and Signals* (pp. 155-168). Springer, Singapore.
- [7] Ansari, N., Faizabadi, A., Motakabber, S., & Ibrahimy, M. (2020). Effective Pneumonia Detection using Res Net based Transfer Learning. *Test Engineering and Management*, 82, 15146-15153.
- [8] Jain, R., Nagrath, P., Kataria, G., Kaushik, V. S., & Hemanth, D. J. (2020). Pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning. *Measurement*, 165, 108046.
- [9] Ayan, E., & Ünver, H. M. (2019, April). Diagnosis of pneumonia from chest X-ray images using deep learning. In *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)* (pp. 1-5). Ieee.
- [10] Eid, M. M., & Elawady, Y. H. (2021). Efficient pneumonia detection for chest radiography using ResNet-based SVM. *European Journal of Electrical Engineering and Computer Science*, 5(1), 1-8.
- [11] Bharati, S., Podder, P., Mondal, M., & Prasath, V. B. (2021). CO-ResNet: Optimized ResNet model for COVID-19 diagnosis from X-ray images. *International Journal of Hybrid Intelligent Systems*, (Preprint), 1-15.