TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Kathmandu Engineering College

Department of Computer Engineering

Project Report

On

Covid / Non-Covid Prediction

By

Prayas Silwal (KAT076BCT055)

Shrutee Gautam (KAT076BCT074)

Shupraj Shiwakoti (KAT076BCT075)

Suyash Bahadur Shrestha (KAT076BCT089)

Kathmandu, Nepal

2079

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Kathmandu Engineering College
Department of Computer Engineering

Minor Project Mid Term Report

On

Covid / Non-Covid Prediction

PROJECT REPORT SUBMITTED TO

THE DEPARTMENT OF COMPUTER ENGINEERING

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR

THE BACHELOR OF ENGINEERING

By

Prayas Silwal (KAT076BCT055)

Shrutee Gautam (KAT076BCT074)

Shupraj Shiwakoti (KAT076BCT075)

Suyash Bahadur Shrestha (KAT076BCT089)

Kathmandu, Nepal

TRIBHUWAN UNIVERSITY
Kathmandu Engineering College
Department of Computer Engineering

ABSTRACT

COVID-19 global pandemic affects health care and lifestyle worldwide, and its early detection is critical to control cases' spreading and mortality. The actual leader diagnosis test is the Reverse transcription Polymerase chain reaction (RT-PCR), result times, availability and cost of these tests are high, so other fast and accessible diagnostic tools are needed. Inspired by recent research that correlates the presence of COVID-19 to findings in Chest X-ray images, this papers' approach uses existing deep learning model ie. convolutional neural networks (CNN) to process these images and classify them as positive or negative for COVID-19.

We used a dataset of X-ray images of both COVID-19 positive and negative cases for training and validation of the model. The data was preprocessed and augmented to increase the diversity of the training data. The model was built using the Keras framework.

The model was evaluated on a validation set and the results showed an accuracy of approximately 98%. These results demonstrate the potential of using deep learning models in detecting COVID-19 from chest X-rays, which can provide a fast, non-invasive and cost-effective alternative to other screening methods.

The trained model has the potential to be integrated into clinical settings to aid healthcare professionals in the rapid and accurate diagnosis of COVID-19.

Table of Contents

Chapter 1: Introduction

1.1. Background theory

The virus called the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) had been discovered in late 2019. The virus which originated in China became a cause of a disease known as Corona Virus Disease 2019 or COVID-19. The World Health Organization (WHO) declared the disease as a pandemic in March 2020. According to the reports issued and updated by global healthcare authorities and state governments, the pandemic affected millions of people globally. The most serious illness caused by COVID-19 is related to the lungs such as pneumonia. The symptoms of the disease can vary and include dyspnea, high fever, runny nose, and cough. These cases can most commonly be diagnosed using chest X-ray imaging analysis for the abnormalities.

X-radiation or X-ray is an electromagnetic form of penetrating radiation. These radiations are passed through the desired human body parts to create images of internal details of the body part. Chest X-ray is used to diagnose the chest-related diseases like pneumonia and other lung diseases, as it provides the image of the thoracic cavity, consisting of the chest and spine bones along with the soft organs including the lungs, blood vessels, and airways. The X-ray imaging technique provides numerous advantages as an alternative diagnosis procedure for COVID-19 over other testing procedures. These benefits include its low cost, the vast availability of X-ray facilities, noninvasiveness, less time consumption, and device affordability. Thus, X-ray imaging may be considered a better candidate for the mass, easy, and quick diagnosis procedure for a pandemic like COVID-19 considering the current global healthcare crisis.

Deep learning and ANNs have endorsed an exponential research focus over the last decade. The deep ANNs have outperformed other conventional models on many essential benchmarks. The advancement of ANNs has massive potential in healthcare applications, specifically in medical data analysis, diagnosis through medical image processing, and analysis. Considering the present pandemic situation, there is an appurtenant relationship between the detection of COVID-19 cases and chest X-ray image analysis and classification.

In this work, an automatic diagnostic system has been developed using CNN which uses chest X-ray analysis results to diagnose whether a person is COVID-19-affected or normal. Preliminary analysis of this study has shown promising results in terms of its accuracy and other performance parameters to diagnose the disease in a cost-effective and time-efficient manner. CNN is a type of DNN (Deep Neural Network), inspired by the visual system of the human brain, and is most commonly used in the analysis of visual imagery. To train the CNN model, first, the dataset has been obtained from Kaggle. Since the dataset obtained for training the model was very small in size and imbalanced, to solve the problem of having very-limited-sized X-ray image dataset, it has been extended using data augmentation techniques to increase its size and also to make the

model training feature rich. Each layer is independently trained to make accurate predictions by gaining knowledge for the discrete role that is considered.

1.2. Objective

The main objective of this project is to:

- Predict covid/non-covid of a patient using Convolution Neural Network algorithm

1.3. Scope and application

The model uses various data from the patient like X-ray and symptoms to predict the presence of COVID. Application of this project can be:

- In medical sector, to predict the presence of COVID in a patient.
- With slight alteration, it can be used to predict the presence of other diseases in a patient

Chapter 2: Methodology

2.1 Model Architecture

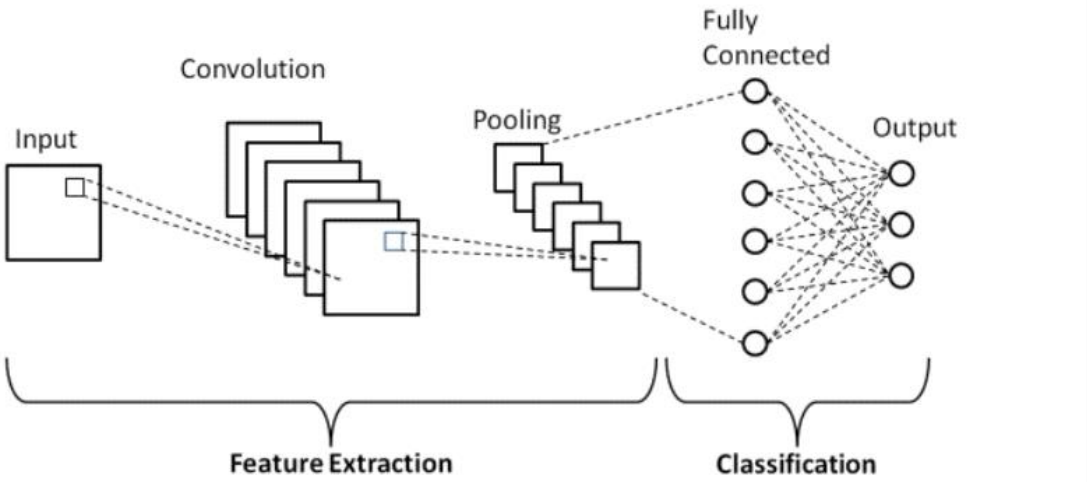


fig: model architecture of Convolution Neural Network(CNN)

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. A CNN typically has three layers:

Convolution Layer: This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field.

Pooling layer: It replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually.

Fully Connected layer: Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN. This is why it can be computed as usual by a matrix multiplication followed by a bias effect. The FC layer helps to map the representation between the input and the output.

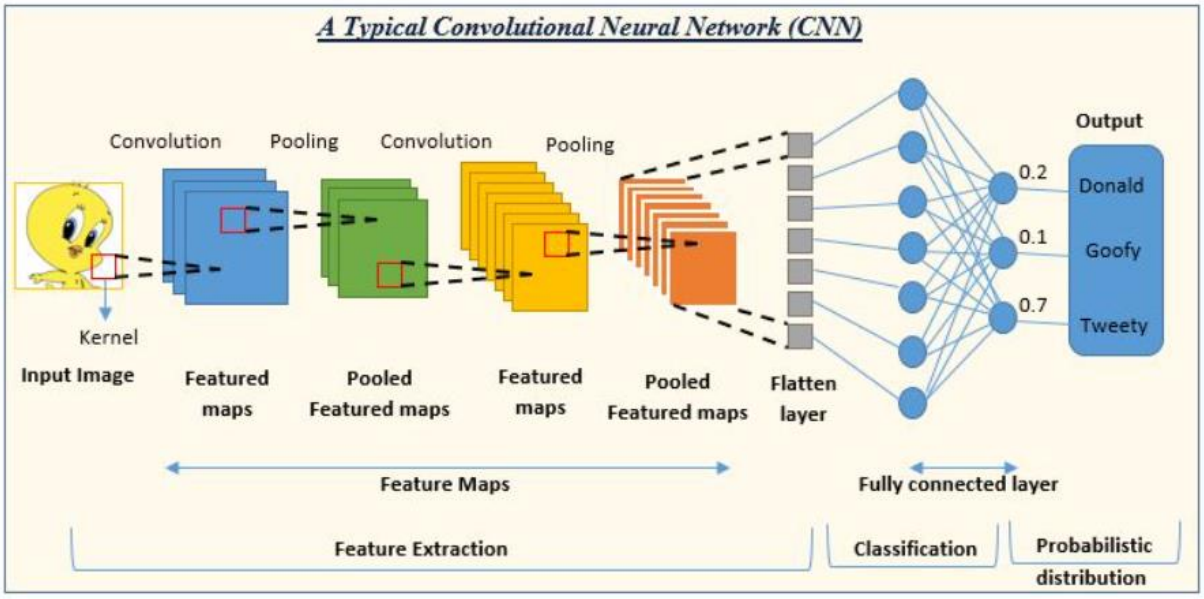


fig: internal architecture of Convolution Neural Network(CNN) with neural representation

2.2 Tools used:

Python:

Python is a high-level, interpreted, general-purpose programming language. Python is developed under an OSI-approved open-source licence, making it freely usable and distributable, even for commercial use.

Google Colab:

Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Kaggle:

Kaggle is an online community platform for data scientists and machine learning enthusiasts. Kaggle allows users to collaborate with other users, find and publish datasets, use GPU integrated notebooks, and compete with other data scientists to solve data science challenges.

NumPy:

NumPy is a library for Python, which adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Pandas:

Pandas is a library written for Python, for data manipulation and analysis. Specifically, it offers data structures and operations for manipulating numerical tables and time series.

ImageDataGenerator:

ImageDataGenerator is used to take the inputs of the original data and then transform it on a random basis, returning the output resultant containing solely the newly changed data. It does not include the data.

TensorFlow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Learning Curves:

Learning Curves are plots used to show a model's performance as the training set size increases. Another way it can be used is to show the model's performance over a defined period of time. We typically used them to diagnose algorithms that learn incrementally from data. Learning curves show the effect of adding more samples during the training process. The effect is depicted by checking the statistical performance of the model in terms of training score and testing score.

2.3 Dataset

The datasets ie. chest X-rays of people suffering from covid, viral pneumonia and normal patients were taken from Kaggle. For training purpose, a total of 111 covid, 70 normal and 70 viral pneumonia datas are used. For testing purpose, 26 covid, 20 normal and 20 viral pneumonia datasare used.

Thus a total of 90 normal cases, 90 viral pneumonia and 137 covid datasets are used in this model.

Chapter 3: DISCUSSION

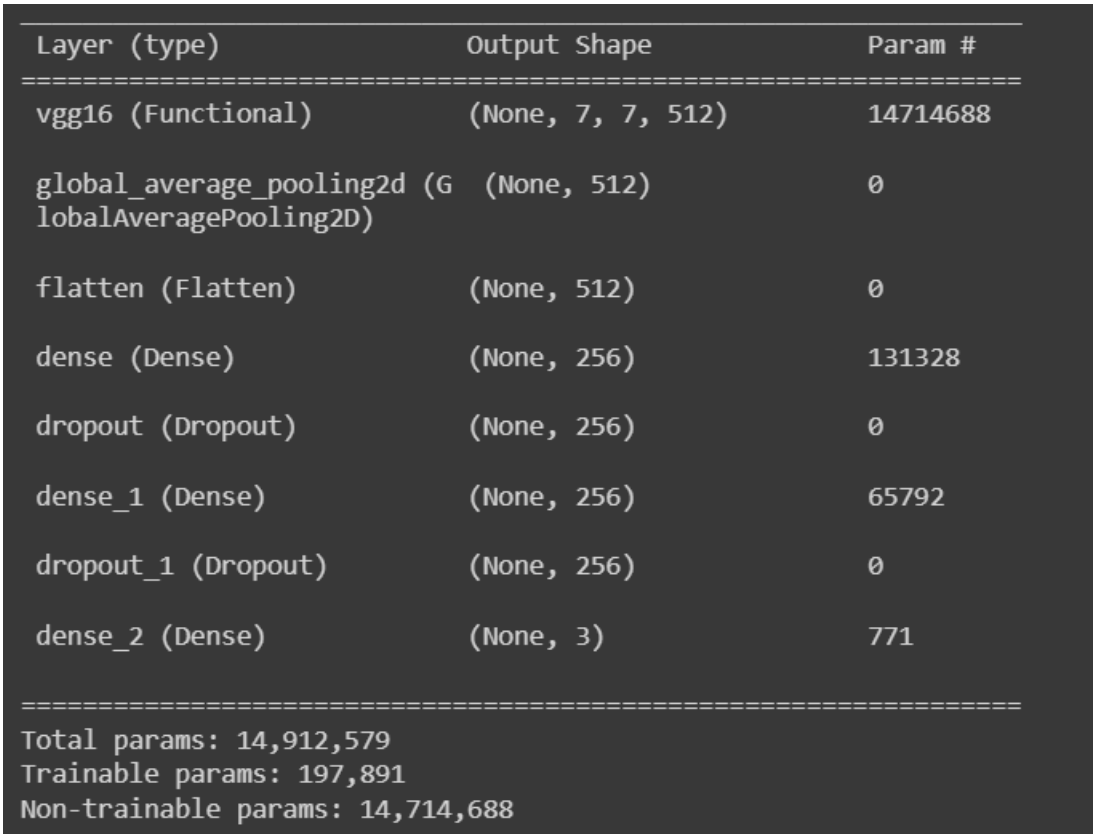
3.1 Data Preprocessing

```
# ImageDataGenerator
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import cv2
from PIL import ImageFilter

height, width = 224, 224
batch_size=64


def generate_data(DIR):
    datagen = ImageDataGenerator(rescale=1./255.)

    generator = datagen.flow_from_directory(
        DIR,
        batch_size=batch_size,
        shuffle=True,
        seed=42,
        class_mode='binary',
        target_size=(height, width),
        classes={'Normal': 0, 'Viral Pneumonia': 1,'Covid': 2}
    )
    return generator

TRAINING_DIR = '/content/Covid19-dataset/train'
TESTING_DIR = '/content/Covid19-dataset/test'

train_generator = generate_data(TRAINING_DIR)
test_generator = generate_data(TESTING_DIR)
```

Here, the training and testing datasets are preprocessed. The code segment prepares the data for the CNN model. It rescales the image pixel values to a range of [0, 1] using ImageDataGenerator, and generates batches of image data from the directories specified by TRAINING_DIR and TESTING_DIR. The generated batches are then used to train and evaluate the CNN model.

3.2 Building the Model

```
from tensorflow.keras.layers.experimental import preprocessing

tf.keras.backend.clear_session()
input_shape = (height, width, 3)
base_model = tf.keras.applications.vgg16.VGG16(
    weights='imagenet',
    include_top=False,
    input_shape=input_shape
)
base_model.trainable = False

model_vgg16 = tf.keras.Sequential()
model_vgg16.add(base_model)
model_vgg16.add(tf.keras.layers.GlobalAveragePooling2D())

model_vgg16.add(tf.keras.layers.Flatten())
model_vgg16.add(tf.keras.layers.Dense(256, activation='relu'))
model_vgg16.add(tf.keras.layers.Dropout(0.5))
model_vgg16.add(tf.keras.layers.Dense(256, activation='relu'))
model_vgg16.add(tf.keras.layers.Dropout(0.5))

model_vgg16.add(tf.keras.layers.Dense(3, activation='softmax'))

model_vgg16.compile(loss='SparseCategoricalCrossentropy',
              optimizer=tf.keras.optimizers.Adam(0.001),
              metrics=['acc'])
model_vgg16.summary()
```



Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 7, 7, 512)	14714688
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 256)	131328
dropout (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 256)	65792
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 3)	771

Total params: 14,912,579
Trainable params: 197,891
Non-trainable params: 14,714,688

Fig: CNN model summary

This code segment defines and compiles a convolutional neural network (CNN) model based on the VGG16 architecture for image classification. The first few lines import the necessary modules and clear the Keras backend to remove any existing models. The input shape of the model is defined as a tuple (height, width, 3) which specifies the dimensions of the input image. The VGG16 model is loaded from the Keras applications module with pre-trained weights from the ImageNet dataset. The include_top argument is set to False to exclude the final classification layer of the VGG16 model, which will be replaced with new layers that are specific to this application. The base model is set to be non-trainable by setting the trainable parameter to False.

A new Keras sequential model, model_vgg16, is defined, which consists of the pre-trained VGG16 model, followed by a GlobalAveragePooling2D layer to reduce the dimensionality of the output, and two fully connected dense layers with dropout layers for regularization. The final layer is a dense layer with 3 output units (corresponding to the three classes) and a softmax activation function.

The model is compiled with the SparseCategoricalCrossentropy loss function, the Adam optimizer with a learning rate of 0.001, and accuracy as the evaluation metric. Finally, the summary() function is called to display a summary of the model architecture.

3.3 Train the model

```
checkpoint = tf.keras.callbacks.ModelCheckpoint('model/vgg16_best.h5', monitor='acc', verbose=1, mode='max',save_best_only=True)
early = tf.keras.callbacks.EarlyStopping(monitor="val_loss", mode="min",restore_best_weights=True, patience=5)

callbacks_list = [checkpoint,early]

history = model_vgg16.fit(
        train_generator,
        validation_data = test_generator,
        #steps_per_epoch=10,
        epochs=50,
        shuffle=False,
        verbose=True,
        callbacks=callbacks_list)
```

This code segment sets up and performs the training of the model_vgg16 model using the fit method. It trains the model on the data generated by the train_generator and evaluates the model on the data generated by the test_generator.

The callbacks argument is used to specify a list of callback functions to be applied during training. In this case, two callbacks are used: ModelCheckpoint and EarlyStopping.

ModelCheckpoint saves the model weights to a file whenever the accuracy of the model improves over the previous best accuracy on the validation set. This is useful for later re-use of the best model found during training.

EarlyStopping stops the training of the model early if the validation loss does not improve for a certain number of epochs, which is set to 5 in this case. This is useful for preventing overfitting and saving training time.

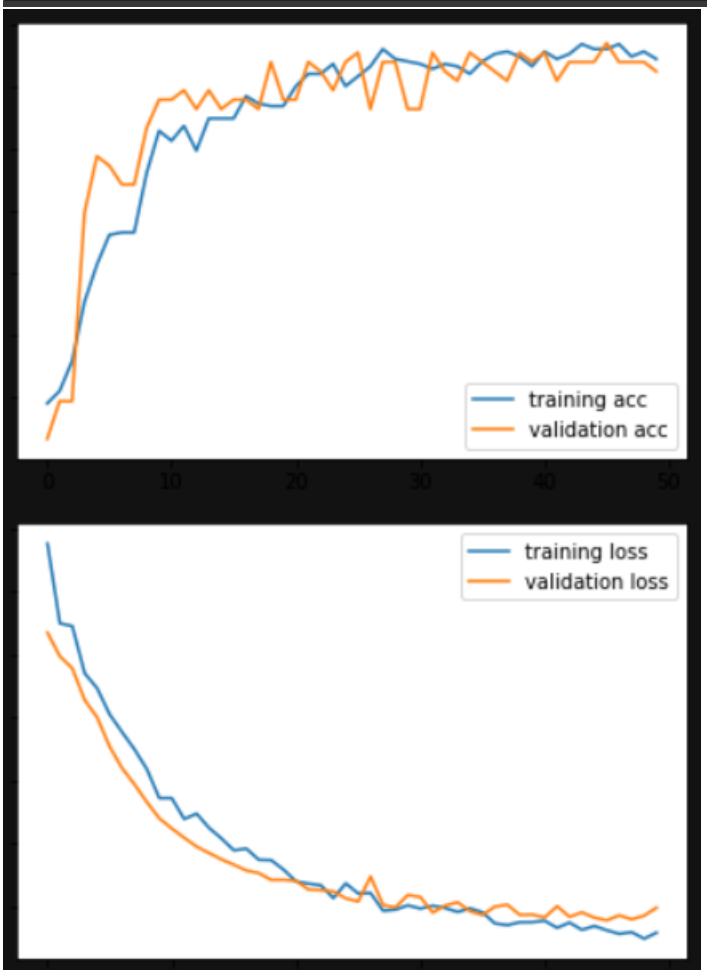
The training progress is stored in the history variable which contains information such as the loss and accuracy of the model during training.

Chapter 4: Conclusion

4.1 Results

4.1.1 Plotting learning curve

```python
# plot learning curve
def plot_learning_curve(history):
    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(len(acc))

    plt.plot(epochs, acc, label='training acc')
    plt.plot(epochs, val_acc, label='validation acc')
    plt.legend();
    plt.figure();

    plt.plot(epochs, loss, label='training loss')
    plt.plot(epochs, val_loss, label='validation loss')
    plt.legend();

plot_learning_curve(history)
```

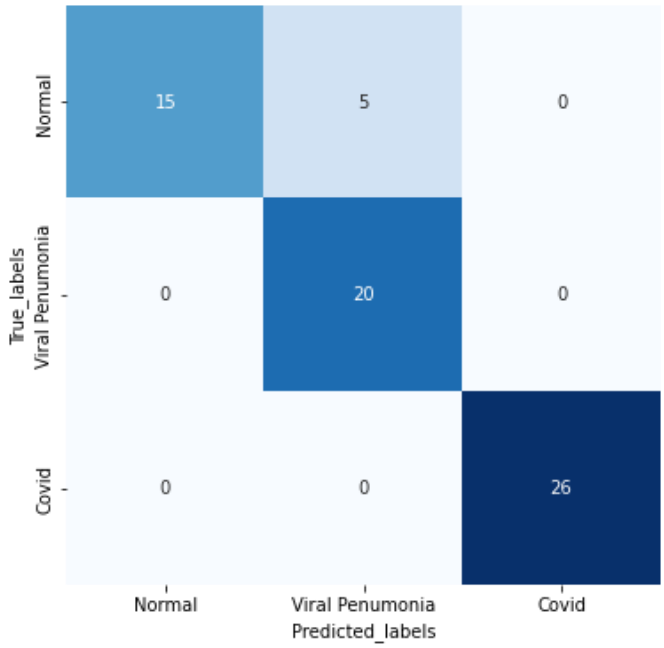


The above code defines a function plot_learning_curve which takes in a history object as input. It then extracts the training and validation accuracy and loss from the history object for each epoch of training. Finally, it uses the matplotlib library to plot the training and validation accuracy and loss over the epochs of training. This function can be used to visualize the learning curve of a deep learning model during training, which can provide insights into how well the model is learning and whether it is overfitting or underfitting.

4.1.2 Model Evaluation

```
ytest = np.array([])
xtest = []

for i in range(math.ceil(len(test_generator.classes)/batch_size)):
    xtest.append(test_generator[i][0])
    ytest= np.concatenate((ytest,test_generator[i][-1]))

xtest = np.concatenate((xtest),axis=0)

ypred_prob =model_vgg16.predict(xtest)
ypred = np.argmax(ypred_prob,axis=1)

plt.figure(figsize=(6, 6))
hm = sns.heatmap(confusion_matrix(ytest,ypred), annot=True, vmin=0, fmt='g', cmap='Blues', cbar=False,
           xticklabels=['Normal','Viral Penumonia','Covid'],yticklabels=['Normal','Viral Penumonia','Covid'])
hm.set(xlabel='Predicted_labels')
hm.set(ylabel='True_labels')
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.75      0.86        20
         1.0       0.80      1.00      0.89        20
         2.0       1.00      1.00      1.00        26

    accuracy                           0.92        66
   macro avg       0.93      0.92      0.92        66
weighted avg       0.94      0.92      0.92        66
```



This segment performs the evaluation of the trained model on the test dataset.First, it initializes empty arrays ytest and xtest. It then loops through the test data generator in batches, and appends the test images to the xtest list, and their corresponding labels to the ytest array.

After the loop completes, it concatenates the test images in xtest to a single array using np.concatenate, which is needed for passing the test data to the predict method of the trained model.Then it calls predict method of the trained model to predict the labels of the test data.

Next, it calculates the confusion matrix of the predicted labels vs true labels, and visualizes the matrix using a heatmap using the seaborn library. It also prints the classification report, which includes precision, recall, f1-score and support for each class.

Overall, this segment evaluates the performance of the trained model on the test data, and provides metrics for its accuracy and classification performance.

Thus the model was successfully able to predict almost all of the testing data for covid as seen on the heat map. However according to the heat map 5 of the normal cases were predicted to be of viral pneumonia which does not exactly fall under the scope of this project. Thus even though it is not fully accurate, the model is successful in it's work.

4.1.3 Accuracy

	Train	Val
Loss	0.061917	0.194338
Acc	0.980080	0.924242

The loss and accuracy of the training and validation datasets were achieved as shown above.

Thus we can say the model is a successful one with an average accuracy of 95%