CNN for detecting Pneumonia from X-rays

Reeti Bhagat

Sep 25,2020

Thanks to Springboard mentor

Ash Yousefi

Contents

Problem Statement

- Globally,450 million get infected by pneumonia in a year
- 4million people die from the disease in the world
- 50 thousand people die from the disease in the united states of America
- In this project, we are going to explore X-ray images as doctors frequently use X- rays and CT scans to diagnose pneumonia from the given x-rays.
- Artificial intelligence-based solutions can provide support for experts in the medical domain in performing time-consuming works

What is Pneumonia?

Pneumonia is a lung inflammation caused by a viral or bacterial infection that can range from mild to severe cases.

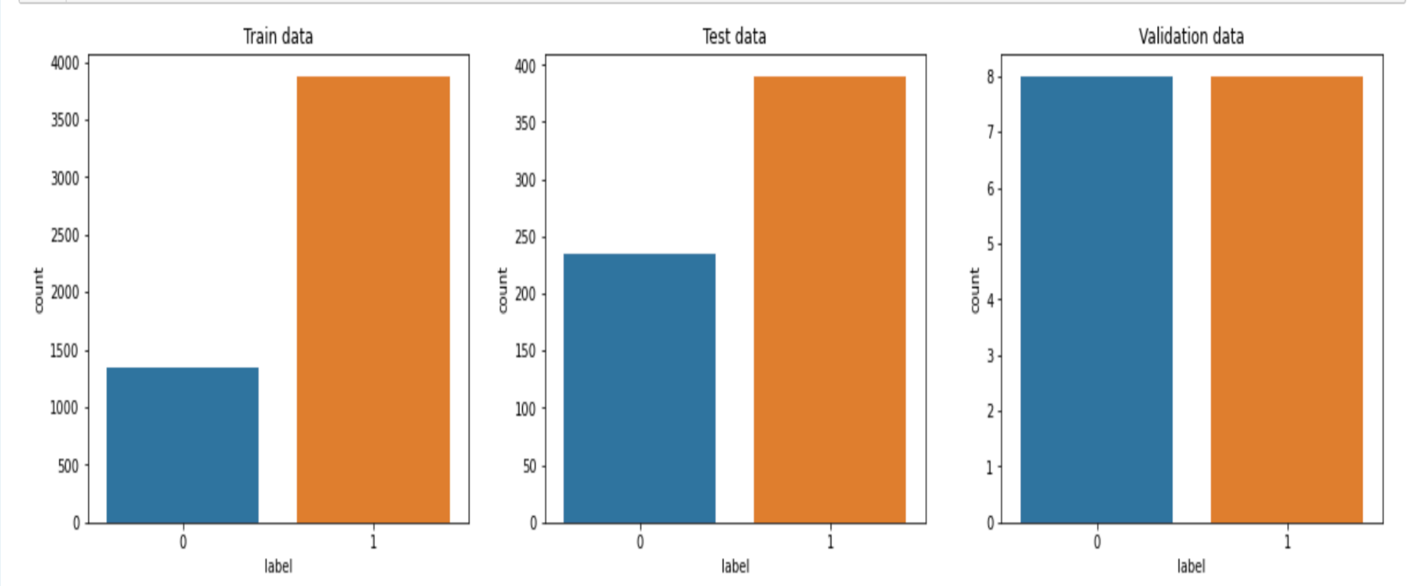
This inflammation makes the patient unable to breathe enough oxygen to reach the bloodstream.

It happens when an infection makes the air sacs (alveoli) in the lungs fill with fluid or pus that might affect either one or both lungs.

If your doctor thinks you might have pneumonia, a chest X-ray will be performed to find the infection in the patient's lungs and how far it’s spread.

Dataset

- https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
- Organized into 3 folders(train,test,val)
- 5,863 X-ray images(jpeg) and 2 categories(Normal/Pneumonia)



Number of images in train_normal_dir: 1342
Number of images in train_pneumonia_dir: 3876
Number of images in val_normal_dir: 9
Number of images in val_pneumonia_dir: 9
Number of images in test_normal_dir: 234
Number of images in test_pneumonia_dir: 390

Fig1. Number of images in each set

Visualization of images

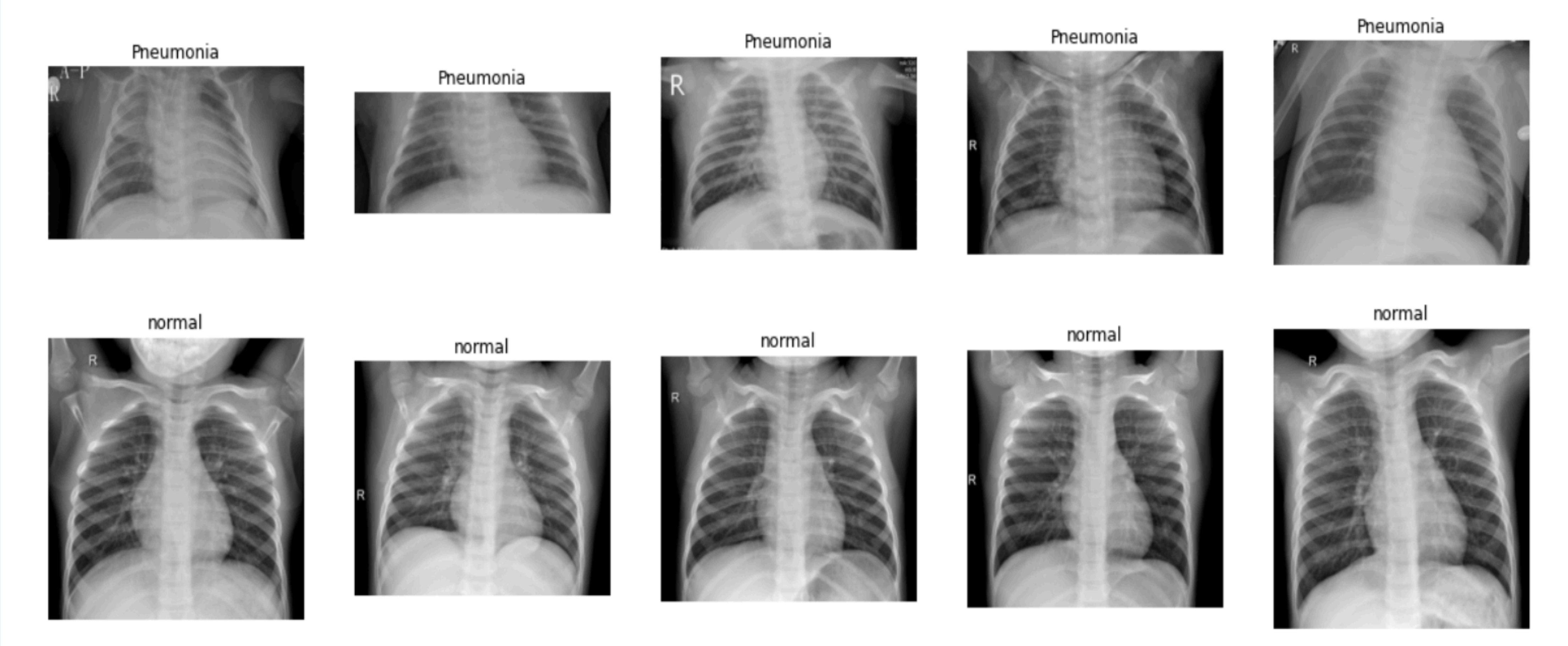


Fig2. Images of chest x-rays

Data Acquisition Using Colab

- Keras
  - High Level library for building deep learning models
  - TensorFlow framework backend
  - Leverage backend engine to compute on GPU
- Google Colab
  - Jupyter notebook environment running Python in Cloud
  - Access to GPU resources
  - Free

Data Preprocessing

- ImageDataGenerator
  - reads images from disk and convert them to float32 tensors and feed them to the network
  - Sets up generators that are capable of loading the required amount of data directly from the source folder, converting them into training data
  - rescales the images
  - Performs data augmentation
    Zoom range=0.1and horizontal_flip=True

- Flow_from_directory
  - load images from the disk, applies rescaling and resizes the images into the required dimensions.

Model Development

tf.keras.layers.Conv2D(): The convolution layer which improves image recognition by isolate images features

tf.keras.layers.MaxPooling2D(): a layer to reduce the information in an image while maintaining features

tf.keras.layers.BatchNormalization(): Normalize and scale inputs or activations

tf.keras.layers.Dropout():Applies Dropout to the input

tf.keras.layers.Flatten(): flatten the result into 1-dimensional array

tf.keras.layers.Dense(): add densely connected layer

A five-layer convolution network will be built where Conv2D() , MaxPooling2D(),Batch Normalization() and Dropout() are stack together as one layer. Then, the output of the final convolutional layer will be flattened and fit to fully connected neurons.

Compiling the Model

```
model=Sequential()
model.add(Conv2D(32 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' , input_shape = (150,150,1)))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.1))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(64 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(128 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
model.add(Conv2D(256 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
model.add(MaxPooling2D((2,2) , strides = 2 , padding = 'same'))
model.add(Flatten())
model.add(Dense(units = 128 , activation = 'relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 1 , activation = 'sigmoid'))
model.compile(optimizer = "Adam" , loss = 'binary_crossentropy' , metrics = ['accuracy'])
model.summary()
```

Compiling the Model

loss: pneumonia detection is using sigmoid activation in the final step, which resulted in either 0 or 1 (normal or pneumonia). Therefore, **binarycrossentropy** is the most suitable loss function

optimizer: Adam will be used

metrics: accuracy is the measurement metric to obtain the prediction accuracy rate on every epoch

Training the Model

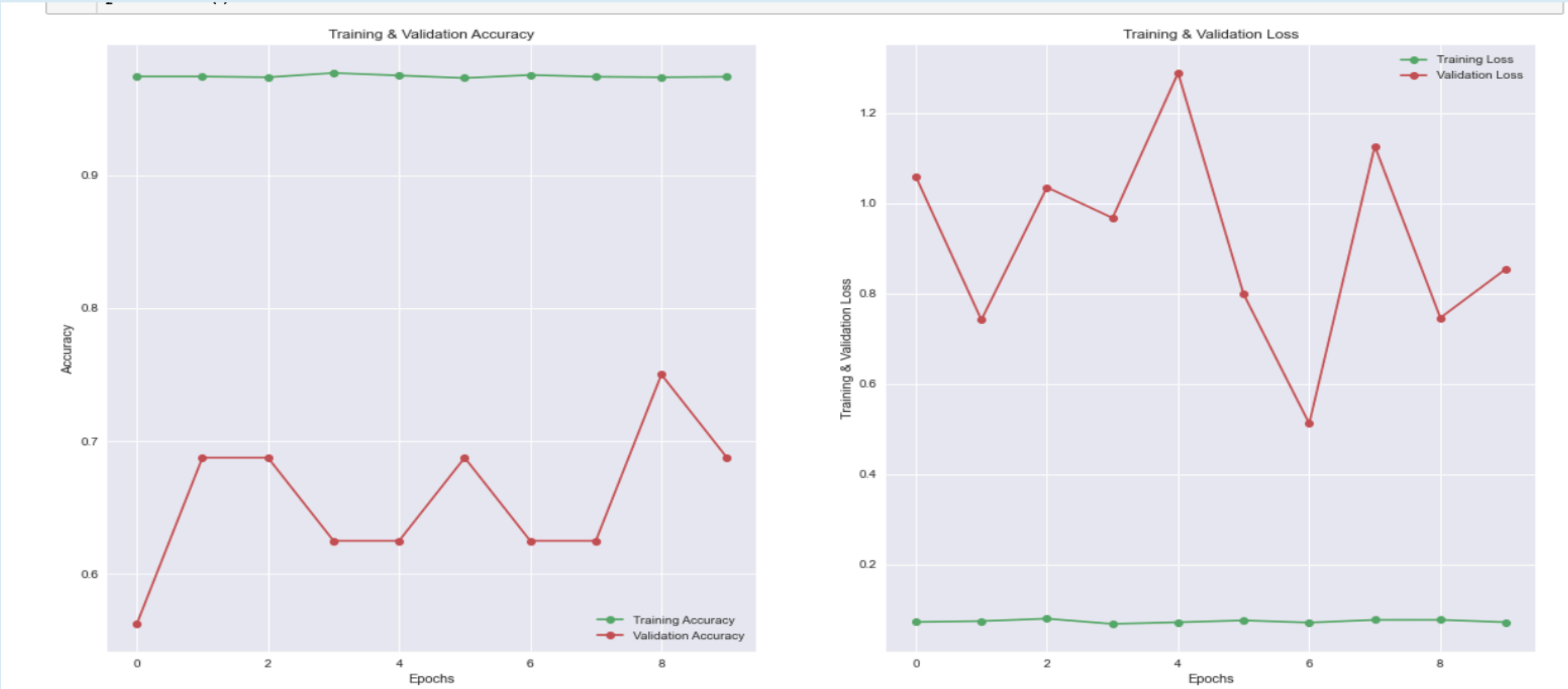
- Model Checkpoint: Saves model or weights at some intervals so the model or weights can be loaded later to training
- ReduceLROnPlateau: Monitors a quantity and if no improvement is seen for a ‘patience’ number of epochs, the learning rate is reduced
- EarlyStopping: Prevents overtraining of the model by terminating process if it’s not really learning anything.

```
]:  learning_rate_reduction = ReduceLROnPlateau(monitor='val_loss', patience = 2, verbose=2,factor=0.3,mode='max')
    checkpoint=ModelCheckpoint(filepath="best_weights.hdf5",monitor='val_loss',save_best_only=True,save_weights_only=Tr
    early_stop=EarlyStopping(monitor='val_loss',min_delta=0.1,patience=1,mode="max")
```

```
]:  history = model.fit(
        datagen.flow(X_train,y_train, batch_size =32) ,
        epochs = 10 ,
        verbose=1,
        validation_data = datagen.flow(X_val, y_val) ,
        callbacks = [learning_rate_reduction,checkpoint,early_stop])
```

Evaluation

Additionally, the following is the analysis of model accuracy and loss rate within 10 epochs after model training between train and validation dataset:



Evaluating the Model

Once the model is completely trained, the test dataset will be used to ensure unbiased prediction of a final model.

This model evaluation step is useful to measure how well the trained model predicts unseen data.

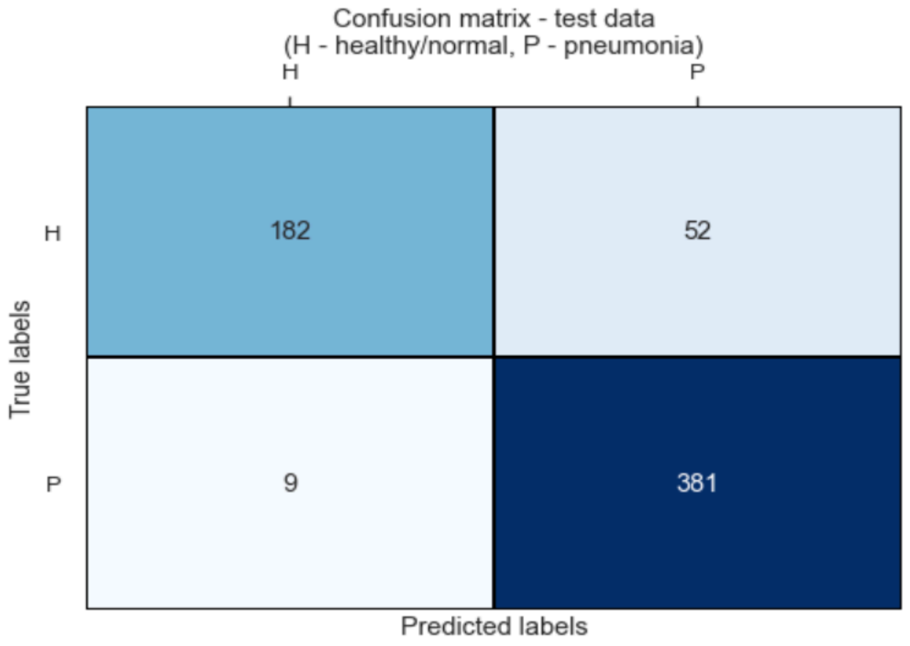
```
print("Loss of the model is - " , model.evaluate(X_test,y_test)[0])
print("Accuracy of the model is - " , model.evaluate(X_test,y_test)[1]*100 , "%")
```

```
20/20 [==============================] - 3s 140ms/step - loss: 0.3413 - accuracy: 0.9022
Loss of the model is -  0.34133675694465637
20/20 [==============================] - 3s 143ms/step - loss: 0.3413 - accuracy: 0.9022
Accuracy of the model is -  90.22436141967773 %
```

The accuracy of model reaches 90.22% with loss of 0.34.

Model Performance

	precision	recall	f1-score	support
Normal (Class 0)	0.95	0.78	0.86	234
pneumonia (Class 1)	0.88	0.98	0.93	390
accuracy			0.90	624
macro avg	0.92	0.88	0.89	624
weighted avg	0.91	0.90	0.90	624

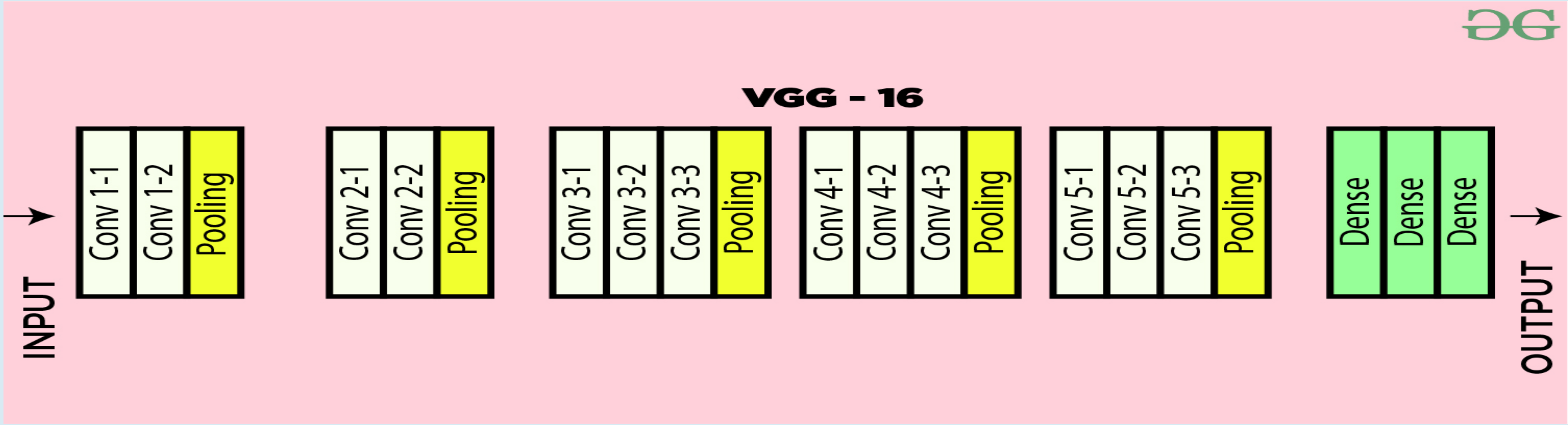


Recall is 97% while precision in 87% so it can be considered as good model while the accuray is 90%.

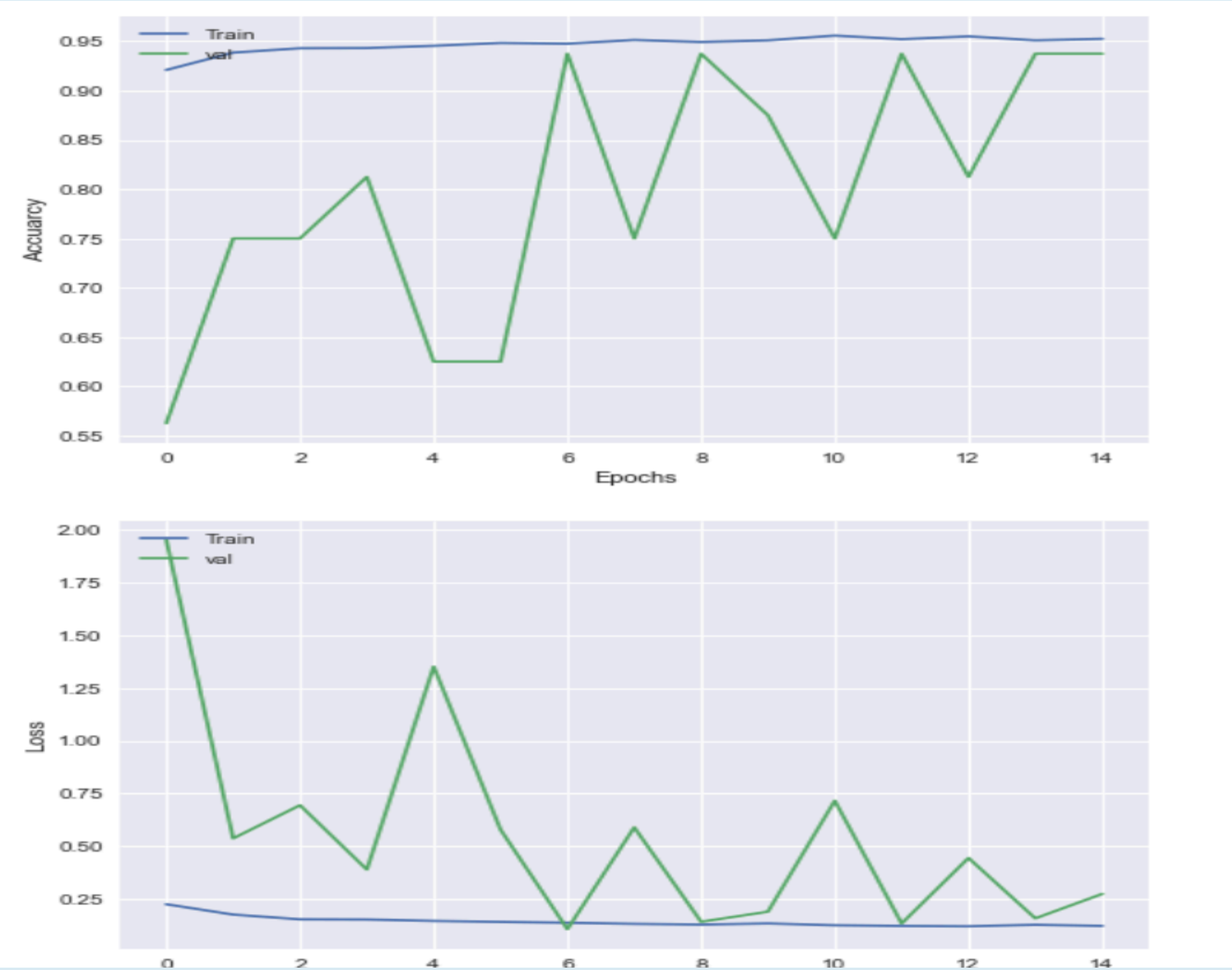
Fig3. Classification Report and Confusion Matrix

Transfer Learning with VGG16

- Model Summary



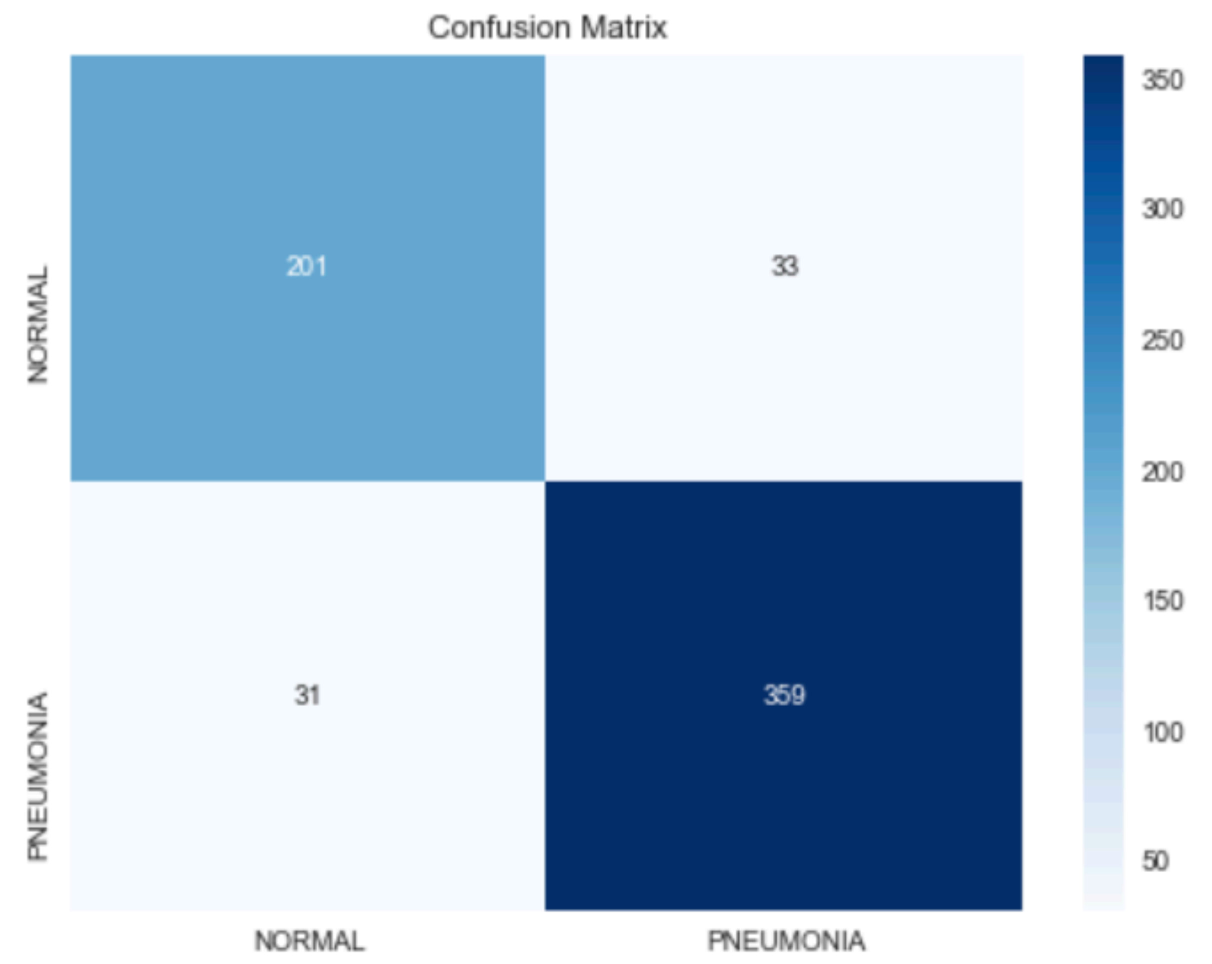
VGG16 TRAINING PERFORMANCE



VGG16 TESTING PERFORMANCE

- Classification Report

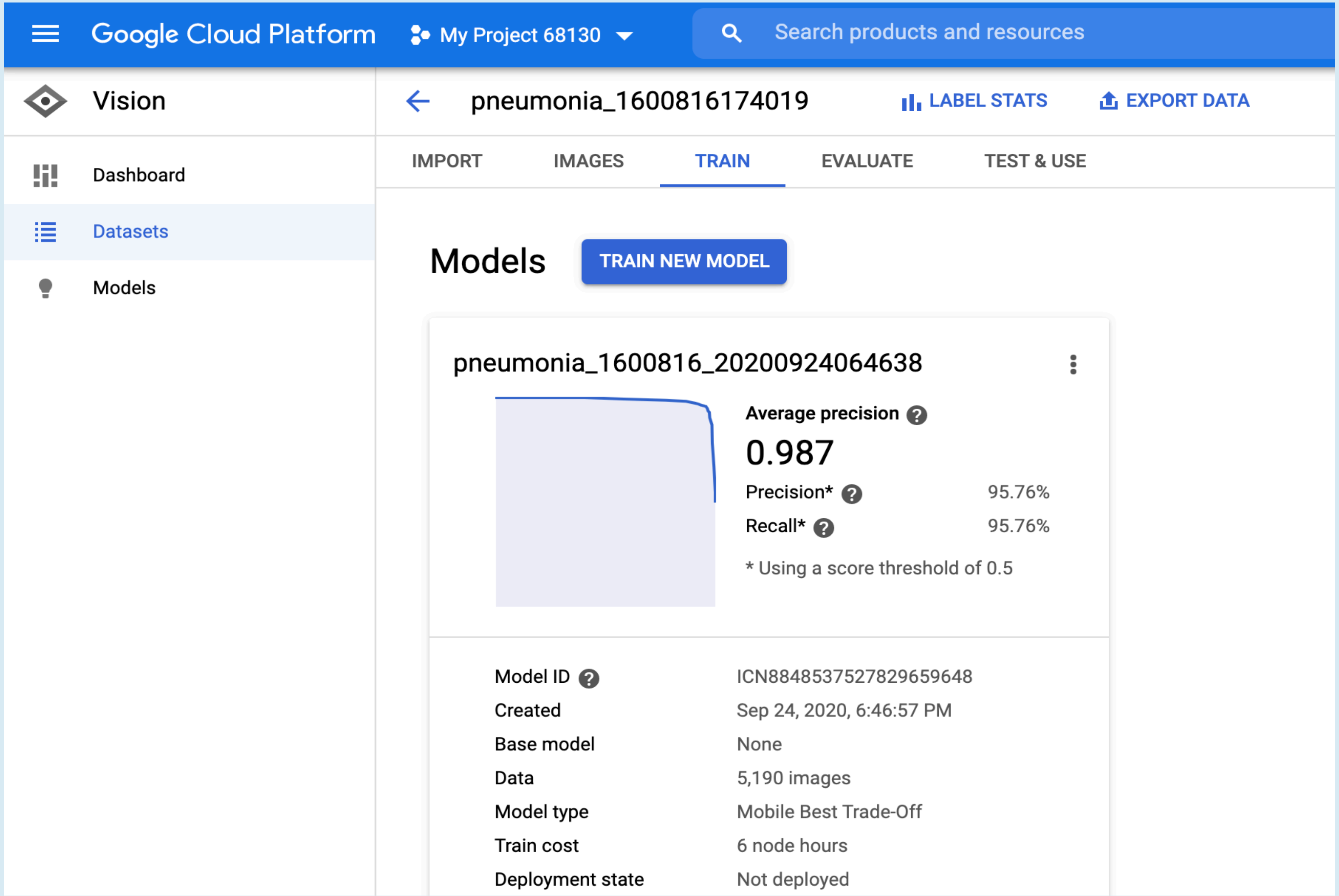
	precision	recall	f1-score	support
0	0.87	0.86	0.86	234
1	0.92	0.92	0.92	390
accuracy			0.90	624
macro avg	0.89	0.89	0.89	624
weighted avg	0.90	0.90	0.90	624



Google Cloud AutoML Vision for Medical Image Classification

- Enable AutoML Cloud Vision on GCP
- Download the Dataset to Google Cloud Storage
- Preparing the Dataset for Modeling
- Modeling with Cloud AutoML Vision
- Testing the Model
- Deploying pneumonia detection web app with Tensorflow.json and AWS ec2 instance

Modeling with Cloud AutoML Vision



SUMMARY OF RESULTS

- Used google collab to apply neural architecture(using Keras)
- Applied Deep Learning with CNN to classify chest x-ray images
- Applied transfer learning VGG16 model to classify X-ray images
- Import, train, test and use in GCP AutoML Vision and deployed pneumonia detection web app with Tensorflow.json .
- VGG16 approach has 88% accuracy while other approaches has 90% accuracy.

FOLLOW – ON WORK

- Improve baseline model accuracy with different approaches
  - Increase the number of convolutional layers
  - Change the predetermined values(epochs, batch , size, callbacks…)

- Increase the number of data
  - Using data augmentation techniques
  - Different DLGAN models

- Train the data using different Transfer Learning models.

THANK YOU !

REETI BHAGAT
Email: bhagat.reeti@gmail.com
LinkedIn: https://www.linkedin.com/in/reeti-bhagat/
Project @ :https://github.com/reetibhagat/Capstone-project2