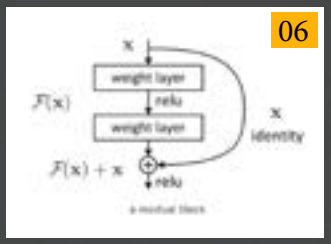
Pepgra

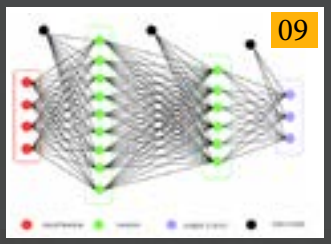
Skin Cancer : Malignant or Benign

HIGHLIGHTS

DATASET

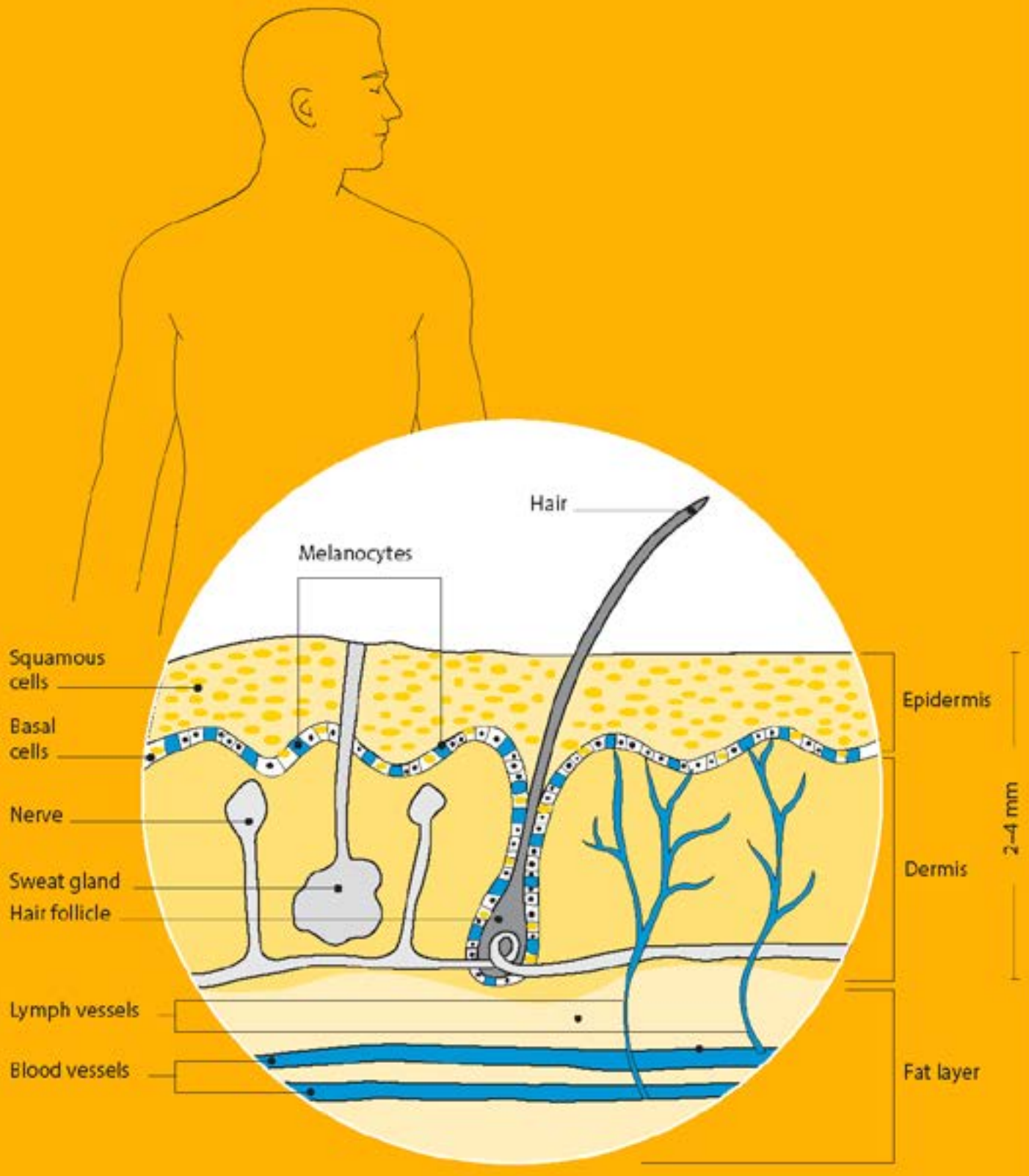


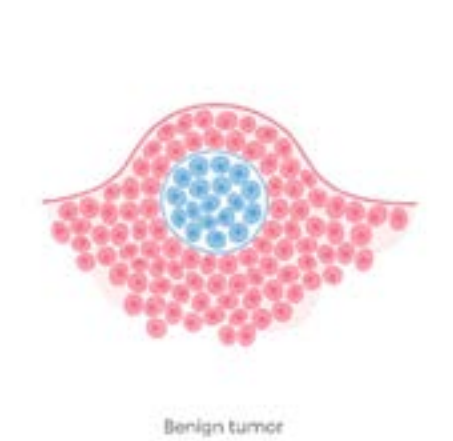
ResNet50

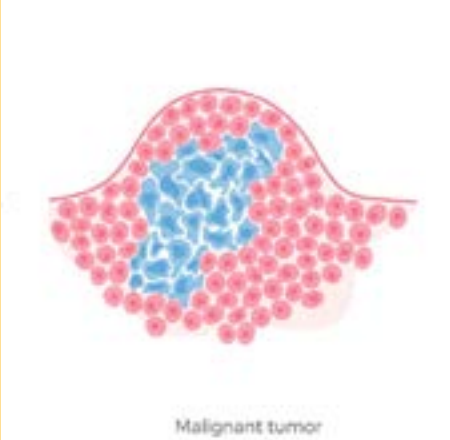


Model Training

THE LAYERS OF THE SKIN







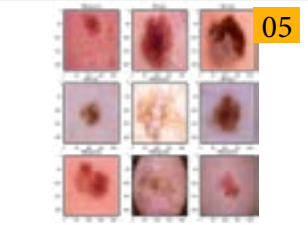
PREFACE

Skin cancers are cancers that arise from the skin. They are due to the development of abnormal cells that have the ability to invade or spread to other parts of the body. There are three main types of skin cancers: basal-cell skin cancer (BCC), squamous-cell skin cancer (SCC) and melanoma. The first two, along with a number of less common skin cancers, are known as nonmelanoma skin cancer (NMSC). Basal-cell cancer grows slowly and can damage the tissue around it but is unlikely to spread to distant areas or result in death. It often appears as a painless raised area of skin, that may be shiny with small blood vessels running over it . Squamous-cell skin cancer is more likely to spread. It usually presents as a hard lump with a scaly top but may also form an ulcer. Melanomas are the most aggressive. Signs include a mole that has changed in size, shape, colour, has irregular edges, has more than one colour, is itchy or bleeds.

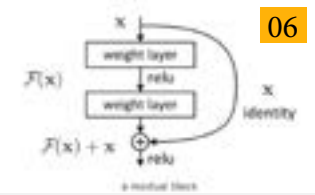
Objective

To predict whether the patients mole is malignant or benign.

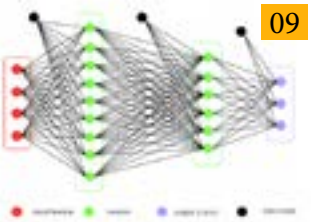
HIGHLIGHTS



DATASET



ResNet50



Model Training

Preface..... 02

Dataset..... 05

ResNet50..... 06

Model..... 07

Hyperparameters..... 08

Model Training..... 09

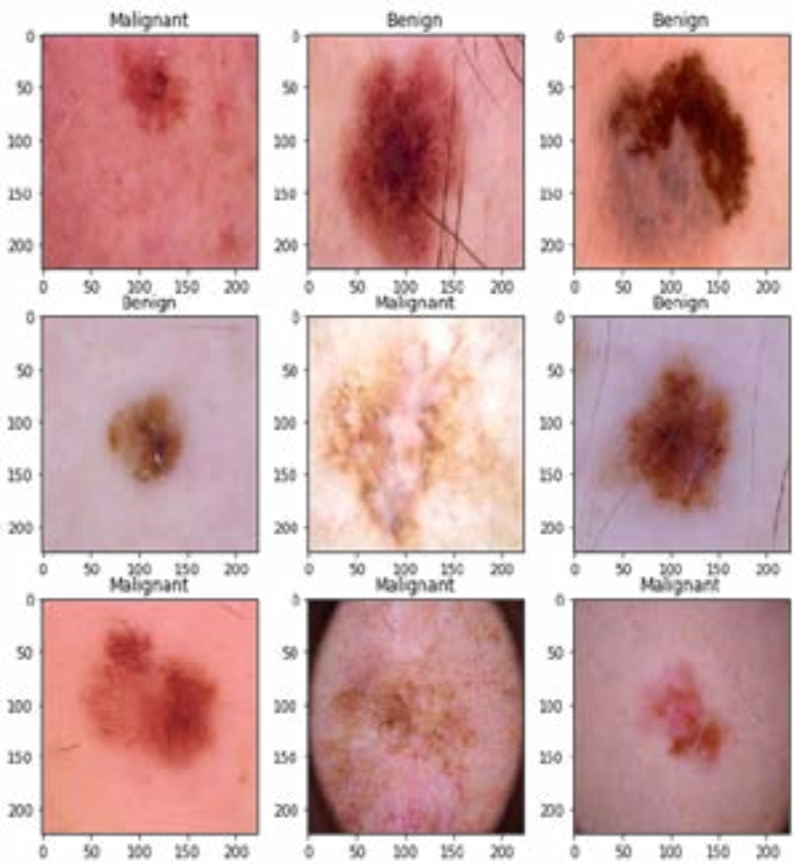
Accuracy..... 10

Dataset

Dataset which we will be using is from ISIC (The International Skin imaging collaboration) Archive. It consists of 1800 pictures of benign moles and 1497 pictures of malignant classified moles.

There are two classes in our dataset, benign and malignant. These two are separated among training set, validation set and testing sets.

Our dataset images looks like this –



> Found - **3,297** images belonging to 2 class.

Next step is to separate our dataset into training set, testing set.

As above image describes, Training set consists of 2637 images and testing set consists of 660 images.

After this we will be rescaling our images and change their shape in 224 * 224 pixels.

```
training_set = train_generator.flow_from_directory('/content/drive/My Drive/skin/train',
                                                   target_size = (224,224),
                                                   batch_size = 64,
                                                   class_mode = 'categorical')

test_set = test_generator.flow_from_directory('/content/drive/My Drive/skin/test',
                                              target_size = (224, 224),
                                              batch_size = 64,
                                              class_mode = 'categorical',
                                              shuffle=False)
```

In this module we will be using ResNet50 model for classifying this problem.

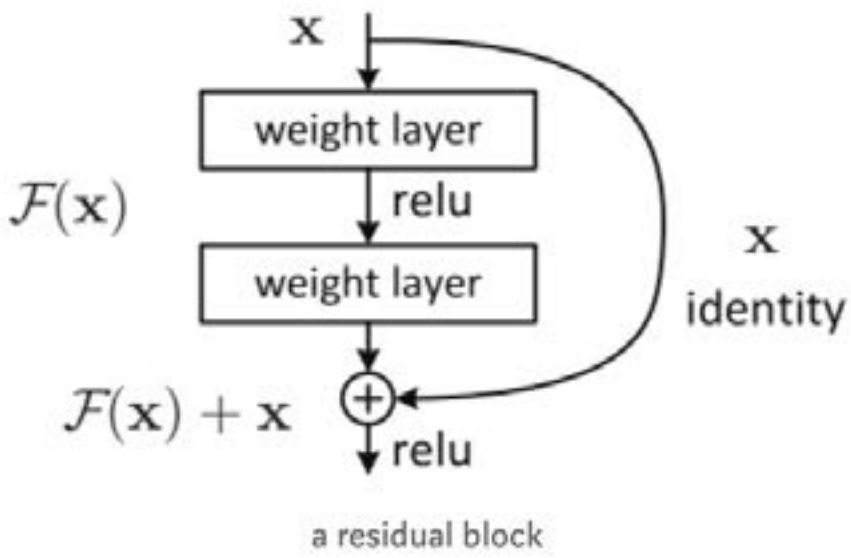
ResNet50

> **ResNet** is instead a form of exotic architecture that relies on micro-architecture modules.

The term micro-architecture refers to the set of "building blocks" used to construct the network. A collection of micro-architecture building blocks (along with your standard CONV, POOL, etc. layers) leads to the macro-architecture.

The main innovation of ResNet is the skip connection. As you know, without adjustments, deep networks often suffer from vanishing gradients, ie: as the model backpropagates, the gradient gets smaller and smaller. Tiny gradients can make learning intractable.

The skip connection in the diagram below is labeled "identity." It allows the network to learn the identity function, which allows it pass the input through the block without passing through the other weight layers.



a residual block

This allows you to stack additional layers and build a deeper network, offsetting the vanishing gradient by allowing your network to skip through layers of it feels they are less relevant in training.
To build a model, what we will do is to have a base model and normal layers after it. In our case Base model Is ResNet50 and follow up with some convolutional layers, batchnormalizarion, dropouts and maxpooling layers.
Lets start with

- Sequential model
- Then we will add resnet50 as our base model.
- Convolutional layer(Conv2D)
- Maxpooling layer
- Dropout layer
- Repeating the convolutional , maxpooling and dropout once again.
- Then we flatten out the layer
- The we will have dense layer with 2 output and softmax as activation func tion.

> ResNet model input shape - **(224,224,3)**

```
base_model= ResNet50(include_top=False, weights='imagenet', input_shape=(224,224,3))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/te
```

Above image gives us a glimpse of ResNet50's parameters –weights are from the ImageNet classifications and it takes the input as (224,224,3).

MODEL

Below image is what we have coded:

```
model= Sequential()
model.add(base_model)
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.40))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2, 2)))
model.add(Dropout(0.40))
model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(2, activation='softmax'))
```

Model summary looks like this –

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 7, 7, 2048)        23587712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 5, 5, 64)          1179712
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 3, 3, 64)          36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 1, 1, 64)          0
_________________________________________________________________
dropout_6 (Dropout)          (None, 1, 1, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               33280
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 1026
=================================================================
Total params: 24,838,658
Trainable params: 24,785,538
Non-trainable params: 53,120
_________________________________________________________________
```

> Optimizer used here is -
> **'ADAM'**

HYPERPARAMETERS:

Filter size(3,3)
Number of filters – 32
Maxpooling size (2,2)
We will have same for the next few layers too as before, then at last sigmoid function is used .
Loss – binary_crossentropy
Metrics – 'Accuracy'
Activation –

- Relu for cnn layers
- Sigmoid for final layer(As our model is binary class output).

MODEL TRAINING

CNN Fit Generator

Cnn model we will be fitting with the training_set and eposhs will be 30 , then for the validation set validation generator is used. We will be using Adam optimizer with learning rate of 0.0001. Loss is categorical cross-entropy and the metrics is accuracy.

> Learning rate - **0.0001**

After this the model starts training for 30 epochs-

```
Epoch 1/30
41/41 [==============================] - 71s 2s/step - loss: 0.4182 - acc: 0.7934 - val_loss: 0.3599 - val_acc: 0.8156
Epoch 2/30
41/41 [==============================] - 69s 2s/step - loss: 0.3223 - acc: 0.8503 - val_loss: 0.3427 - val_acc: 0.8440
Epoch 3/30
41/41 [==============================] - 69s 2s/step - loss: 0.2575 - acc: 0.8868 - val_loss: 0.3064 - val_acc: 0.8188
Epoch 4/30
41/41 [==============================] - 69s 2s/step - loss: 0.2037 - acc: 0.9063 - val_loss: 0.3267 - val_acc: 0.8574
Epoch 5/30
41/41 [==============================] - 69s 2s/step - loss: 0.1718 - acc: 0.9264 - val_loss: 0.2391 - val_acc: 0.9010
Epoch 6/30
41/41 [==============================] - 69s 2s/step - loss: 0.1421 - acc: 0.9394 - val_loss: 0.3981 - val_acc: 0.8658
Epoch 7/30
41/41 [==============================] - 69s 2s/step - loss: 0.1419 - acc: 0.9462 - val_loss: 0.4327 - val_acc: 0.8591
Epoch 8/30
41/41 [==============================] - 69s 2s/step - loss: 0.1125 - acc: 0.9539 - val_loss: 0.4365 - val_acc: 0.8691
Epoch 9/30
41/41 [==============================] - 69s 2s/step - loss: 0.0699 - acc: 0.9737 - val_loss: 0.5394 - val_acc: 0.8456
Epoch 10/30
41/41 [==============================] - 69s 2s/step - loss: 0.0807 - acc: 0.9741 - val_loss: 0.3852 - val_acc: 0.8826
Epoch 24/30
41/41 [==============================] - 70s 2s/step - loss: 0.0148 - acc: 0.9954 - val_loss: 0.4995 - val_acc: 0.8859
Epoch 25/30
41/41 [==============================] - 69s 2s/step - loss: 0.0555 - acc: 0.9795 - val_loss: 0.5732 - val_acc: 0.8826
Epoch 26/30
41/41 [==============================] - 68s 2s/step - loss: 0.0377 - acc: 0.9882 - val_loss: 0.9824 - val_acc: 0.8305
Epoch 27/30
41/41 [==============================] - 70s 2s/step - loss: 0.0334 - acc: 0.9893 - val_loss: 0.4246 - val_acc: 0.9027
Epoch 28/30
41/41 [==============================] - 68s 2s/step - loss: 0.0135 - acc: 0.9950 - val_loss: 0.5380 - val_acc: 0.8943
Epoch 29/30
41/41 [==============================] - 70s 2s/step - loss: 0.0081 - acc: 0.9973 - val_loss: 0.5613 - val_acc: 0.8792
Epoch 30/30
41/41 [==============================] - 69s 2s/step - loss: 0.0295 - acc: 0.9924 - val_loss: 0.4881 - val_acc: 0.8591
<keras.callbacks.History at 0x7fc6fed270f0>
```

After training we will then predict our test variables using model.predict_generator. The Output of the predicted results looks like this –

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 1
0 1 1 1 0 0 0 0 1 0 1 1 0 1 0 0 1 1 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 0
1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 0 0
0 0 0 1 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 0 1 1 0 1 1 1
```

Zero's are malignant and one's are benign.

ACCURACY

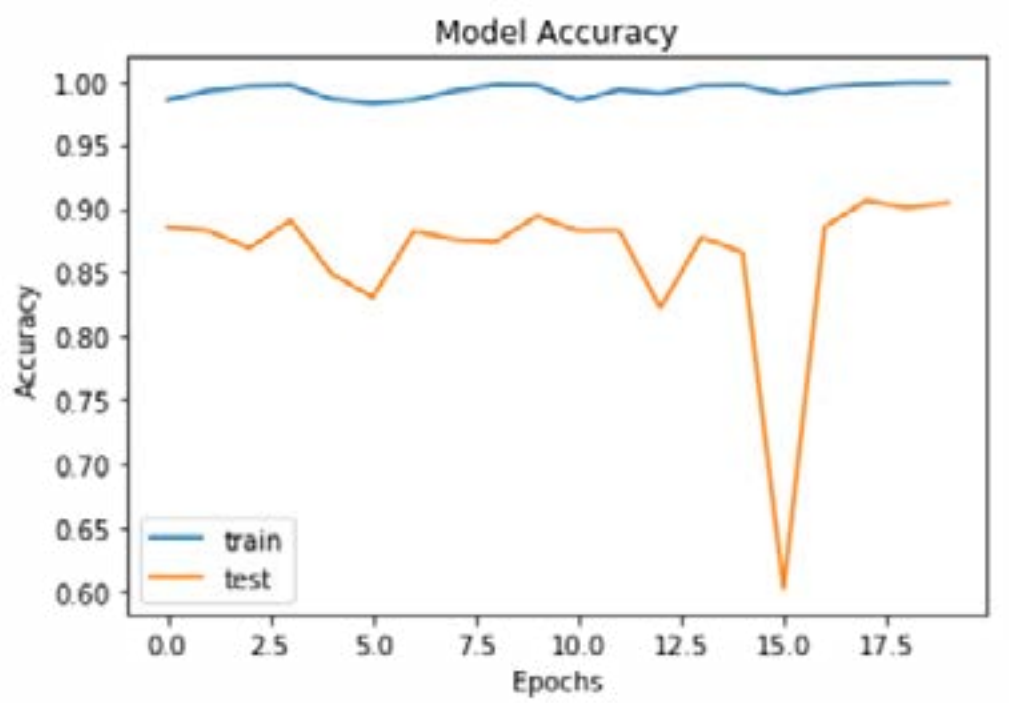
We will then move on to Accuracy and confusion matrix –

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
print(accuracy)
```

```
0.8696969696969697
```

> ACCURACY - 87%

Training accuracy and testing accuracy graph of our Model is below –



Other models tried:

- With more CNN layers and added dropout gave some more accuracy and did not overfit the data.
- We used different Optimizer such as ADAM and ADAGRAD, to check we can achieve better accuracy.
- We have also tried tweaking each parameter of different layers, tried dropout methods again, used different Epochs and batch normalization.

ABOUT US

Pepgra is a leading global contract research outsourcing organization provider of scientific, knowledge-based services to bio-pharmaceutical, generic pharmaceutical, biotech, medical device companies and healthcare companies in the areas of clinical trial monitoring, regulatory writing, post-market surveillance, biostatistics and statistical programming services. Our mission is to become a strategic partner to global life science companies providing high quality knowledge-based expertise across the product lifecycle with the ultimate objective of improving quality of healthcare for patients worldwide. Our corporate headquarter is located in India with operations in USA, and the Philippines

Format type: E-Book

UK: 10 Park Place,
Manchester M4 4EY.
UK: +44-1143520021
Email:info@pepgra.com
Web:www.pepgra.com