CS6005

Deep Learning

Mini-Project Assignment - 1

Image Classification of Malaria Cells using CNN

Steven F. Gilbert

2018103071
'P' Batch

16.04.20

Introduction

This Mini project aims to implement image classification on a suitable dataset using convolutional neural networks. CNN Here we create a CNN network with a combination of convoluted layers and fully connected layers. The chosen dataset is divided into train and test datasets and the model is trained with the former and the latter is later used for validation.

Problem Statement

The problem specified is to predict image classification on a dataset consisting of cell images from infected malaria patients and uninfected samples. The dataset contains about 27,558 cell images which need to be classified. To solve the problem at hand, a suitable CNN model is to be created and validation is to be followed to test the accuracy and efficacy of the proposed model.

Dataset Details

The dataset has been obtained from the National Library of Medicine. The cell images of the dataset have been obtained from a repository of segmented cells from thin blood smear slide images from the Malaria Screener research activity. The samples were extracted from a group of 150 infected people and about 50 uninfected people. This dataset contains a total of 27,558 cell images with half being infected cells and the other half being uninfected cells. The cells images are 128 x 128 in resolution and are configured in 3 color channels.

The dataset is also part of a study conducted by a group of researchers

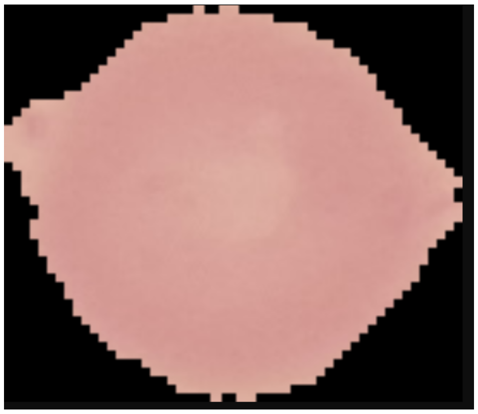
Published in 2018 [1].

Link :

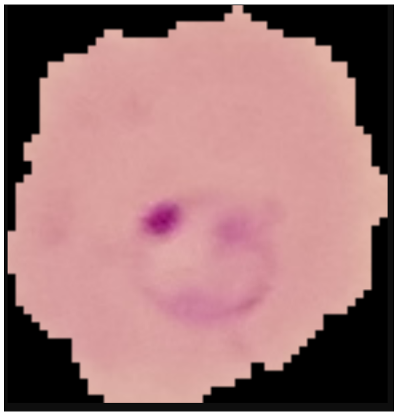
https://lhncbc.nlm.nih.gov/LHC-downloads/downloads.html#malaria-datasets

Sample Data:

Uninfected Cell - 13,779 images



Infected Cell - 13,779 images



Modules

The project code can be split into multiple modules, each with a specific function.

1) Reading the dataset

The dataset is split into two folders one containing the infected cell images and the other having uninfected cell images. Here cv2 is used to read the data into separate variables for each type. The images here are at a resolution of 128 x 128.

2) Preprocessing data

We then ImageDataGenerator from Tensorflow rescale the images at a value of 1/255. We then use the associated functions to split the train and validation sets. Validation split is set 0.2 and batch size at 16. The class mode has also been modified to categorical.

3) Model Creation

The CNN model is created using a sequential model. First 4 convolution layers are added each with a Max Pooling layer and a dropout layer to prevent overfitting. The filters of each layer are 16, 32, 64, 64 respectively. The dropout values are set to 0.3 for each. The activation function used is ‘relu’.

Next is a flatten layer followed by the dense layers. There are two dense layers having 64 and 32 nodes respectively. Both use the ‘relu’ activation function and both have a dropout of 0.2. The output layer has 2 nodes for the classes and an activation function used here is ‘softmax’.

Libraries

The main reason why python is very popular in the field of deep learning is that it boasts a multitude of libraries to perform several important and painstaking modules and functions.

Some of the libraries used in this project are as follows:

1) Tensorflow - TensorFlow is an end-to-end open source platform for machine learning .

2) Keras - It offers consistent & simple APIs, it minimizes the number of user actions required for common use cases .

3) Numpy - Package which provides data computational functions .

4) Matplotlib - Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python .

5) Cv2 - a library of Python bindings designed to solve computer vision problems .

CNN Module Summary

To solve the image classification problem we construct a CNN model to classify the images. The CNN model consists of multiple Convoluted layers, Max Pooling layers, dense and fully connected layers, a flatten layer at the end of the convoluted layers and a dropout layer at each layer to help in regularization.

List on CNN architecture components :

1) **Convolution Layer :**
    a) The convolution layer allows the network to learn filters which maximally respond to a local region of the input.
    b) A pixel increases in correlation to nearby pixels than distant pixels.
    c) It uses a filter or kernel to produce a feature map.

2) **Max Pooling Layer :**
    a) Pooling layers reduce the dimensions of the feature map,thus no.of parameters to learn are also reduced.
    b) This layer summarizes the features of a region in a feature map from a convoluted layer
    c) A Max Pooling layer extracts the maximum value of the area it convolves.

3) **Dropout Layer :**
    a) Dropout is a technique used to prevent the model from overfitting.

   b) It works by randomly setting the outgoing edges of hidden units to 0 at each update of the phase.

4) **Dense Layer :**
   a) The dense layer performs functions on the input and produces an output vector.
   b) It represents the layer which performs matrix vector multiplication.

Model Architecture :

Parameter	**Value**
Epochs	20
Optimizer	Adam
Learning Rate	0.001
Loss function	Categorical Cross Entropy

Convolution Layers :

a) Filters = 16, 32, 64, 64
b) Dropout = 0.3
c) Kernel size = 3*3
d) Activation function = relu

Fully Connected Layers:

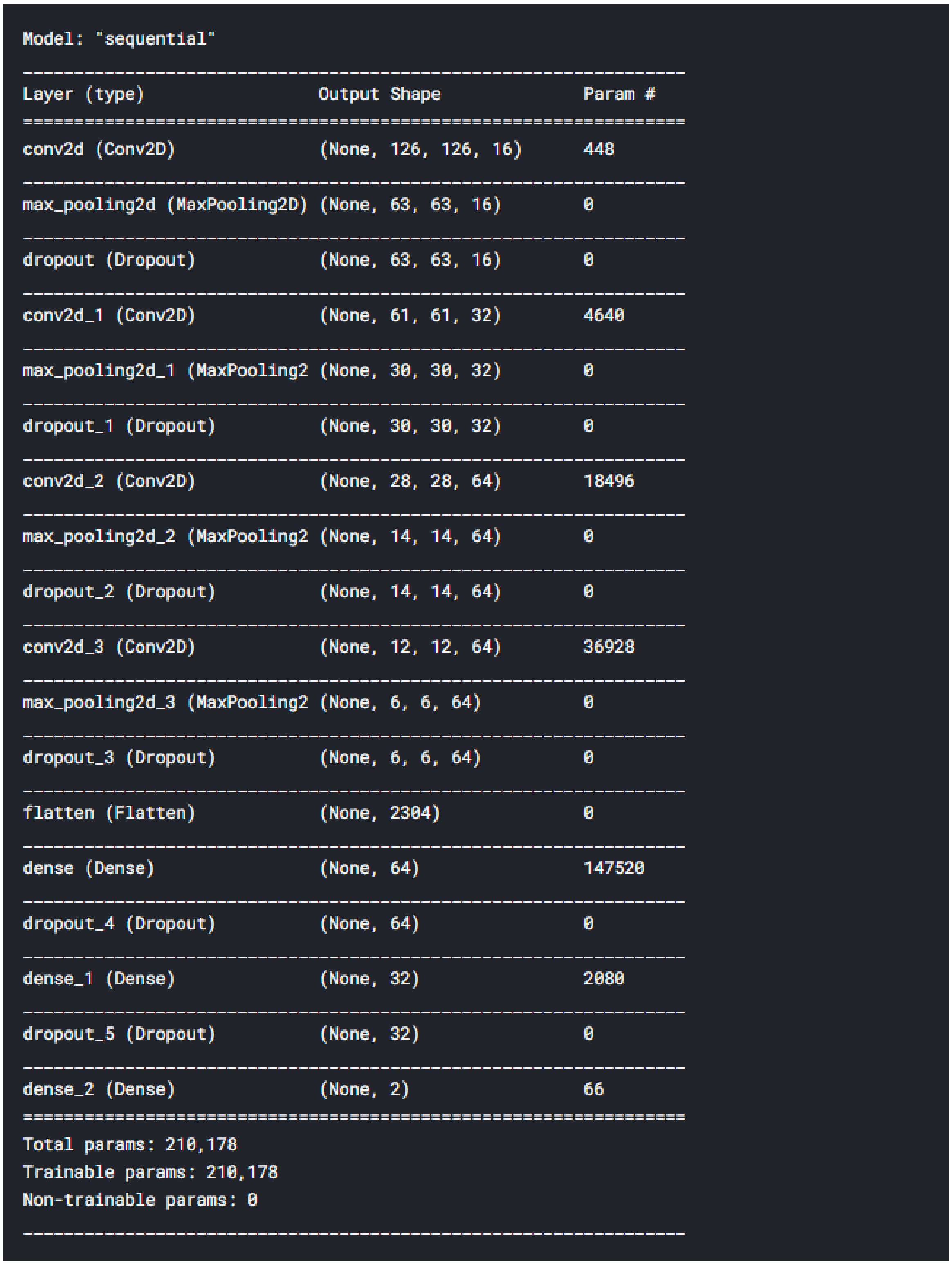
a) Nodes = 64, 32
b) Dropout = 0.2
c) Activation function = relu

Output Layer:

a) Nodes = 2
b) Activation function = softmax

(P.T.O)

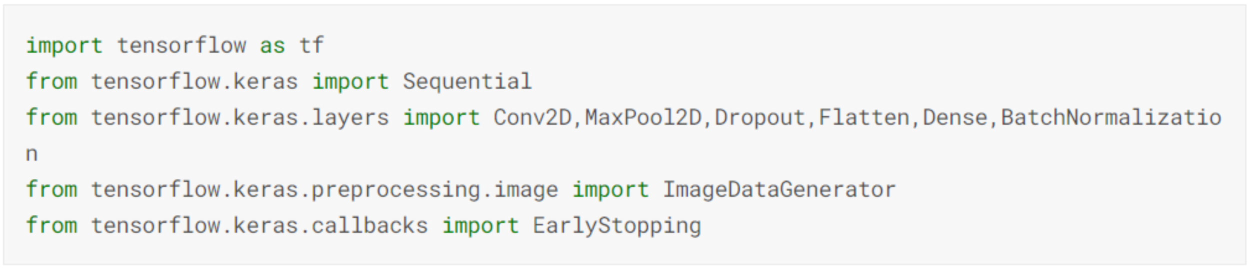
CNN Code Summary



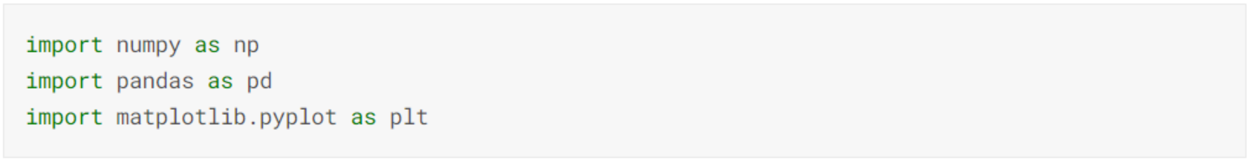
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 63, 63, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 61, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_3 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0
_________________________________________________________________
dense (Dense)                (None, 64)                147520
_________________________________________________________________
dropout_4 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 66
=================================================================
Total params: 210,178
Trainable params: 210,178
Non-trainable params: 0
_________________________________________________________________
```

Coding Snapshots

1) Importing Packages

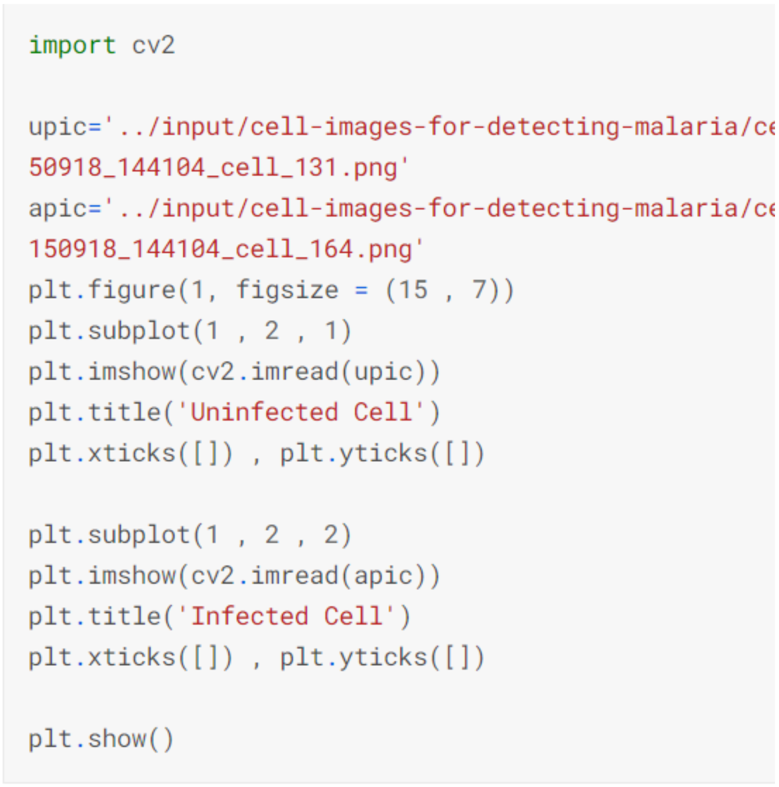


```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D,MaxPool2D,Dropout,Flatten,Dense,BatchNormalizatio
n
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
```



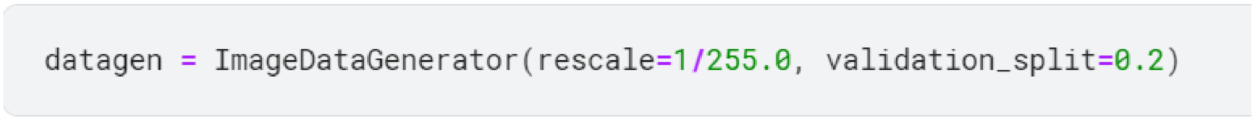
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

2) Reading images from dataset using cv2



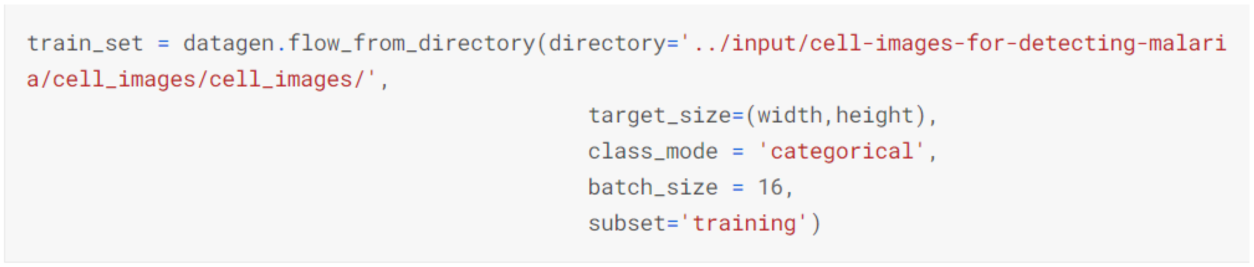
```
import cv2

upic='../input/cell-images-for-detecting-malaria/ce
50918_144104_cell_131.png'
apic='../input/cell-images-for-detecting-malaria/ce
150918_144104_cell_164.png'
plt.figure(1, figsize = (15 , 7))
plt.subplot(1 , 2 , 1)
plt.imshow(cv2.imread(upic))
plt.title('Uninfected Cell')
plt.xticks([]) , plt.yticks([])

plt.subplot(1 , 2 , 2)
plt.imshow(cv2.imread(apic))
plt.title('Infected Cell')
plt.xticks([]) , plt.yticks([])

plt.show()
```

3) Using Tensorflow ImageDataGenerator for rescaling

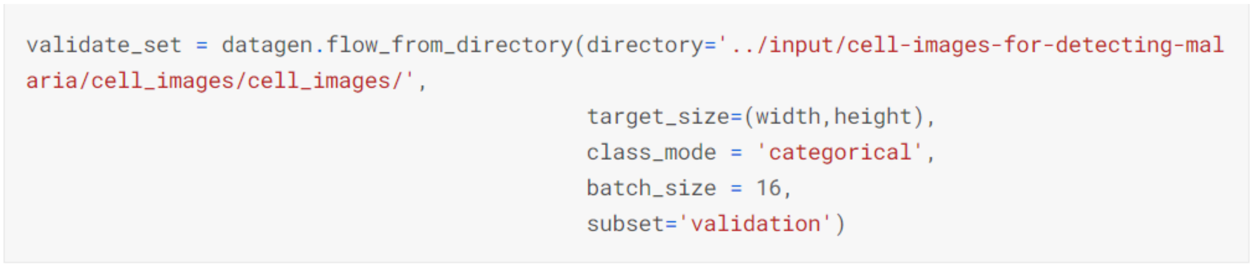


```
datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.2)
```

4) Using Datagen to split train and test data splits and encoding categorical



```
train_set = datagen.flow_from_directory(directory='../input/cell-images-for-detecting-malari
a/cell_images/cell_images/',
                                         target_size=(width,height),
                                         class_mode = 'categorical',
                                         batch_size = 16,
                                         subset='training')
```



```
validate_set = datagen.flow_from_directory(directory='../input/cell-images-for-detecting-mal
aria/cell_images/cell_images/',
                                           target_size=(width,height),
                                           class_mode = 'categorical',
                                           batch_size = 16,
                                           subset='validation')
```

5) Creating Model

```
model = Sequential()
model.add(Conv2D(16,(3,3),activation='relu',input_shape=(128,128,3)))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))

model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.3))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.3))

model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(32,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(2,activation='softmax'))
```

6) Model compile

```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

7) Model training

```
history = model.fit_generator(generator = train_set,
                             steps_per_epoch = len(train_set),
                              epochs =20,
                              validation_data = validate_set,
                              validation_steps=len(validate_set))
```

8) Plotting curves

```
def plotLearningCurve(history,epochs):
  epochRange = range(1,epochs+1)
  plt.plot(epochRange,history.history['accuracy'])
  plt.plot(epochRange,history.history['val_accuracy'])
  plt.title('Model Accuracy')
  plt.xlabel('Epoch')
  plt.ylabel('Accuracy')
  plt.legend(['Train','Validation'],loc='upper left')
  plt.show()

  plt.plot(epochRange,history.history['loss'])
  plt.plot(epochRange,history.history['val_loss'])
  plt.title('Model Loss')
  plt.xlabel('Epoch')
  plt.ylabel('Loss')
  plt.legend(['Train','Validation'],loc='upper left')
  plt.show()
```

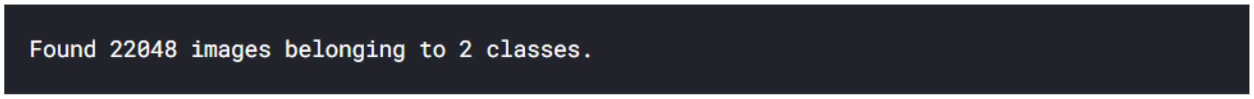
```
plotLearningCurve(history,20)
```

9) Model Evaluation

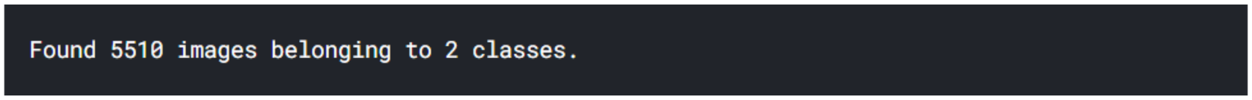
```
scoreSeg = model.evaluate_generator(validate_set, len(validate_set))
```

Results

1) Splitting Train and Test data

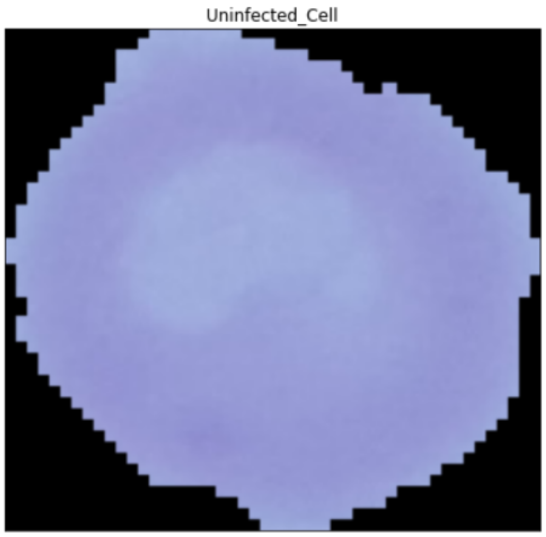


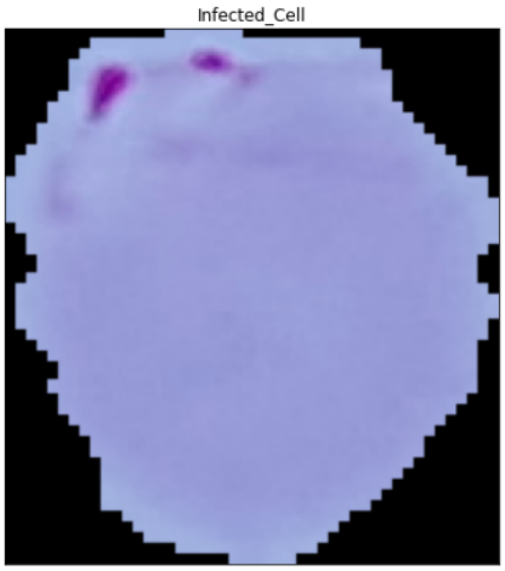
```
Found 22048 images belonging to 2 classes.
```



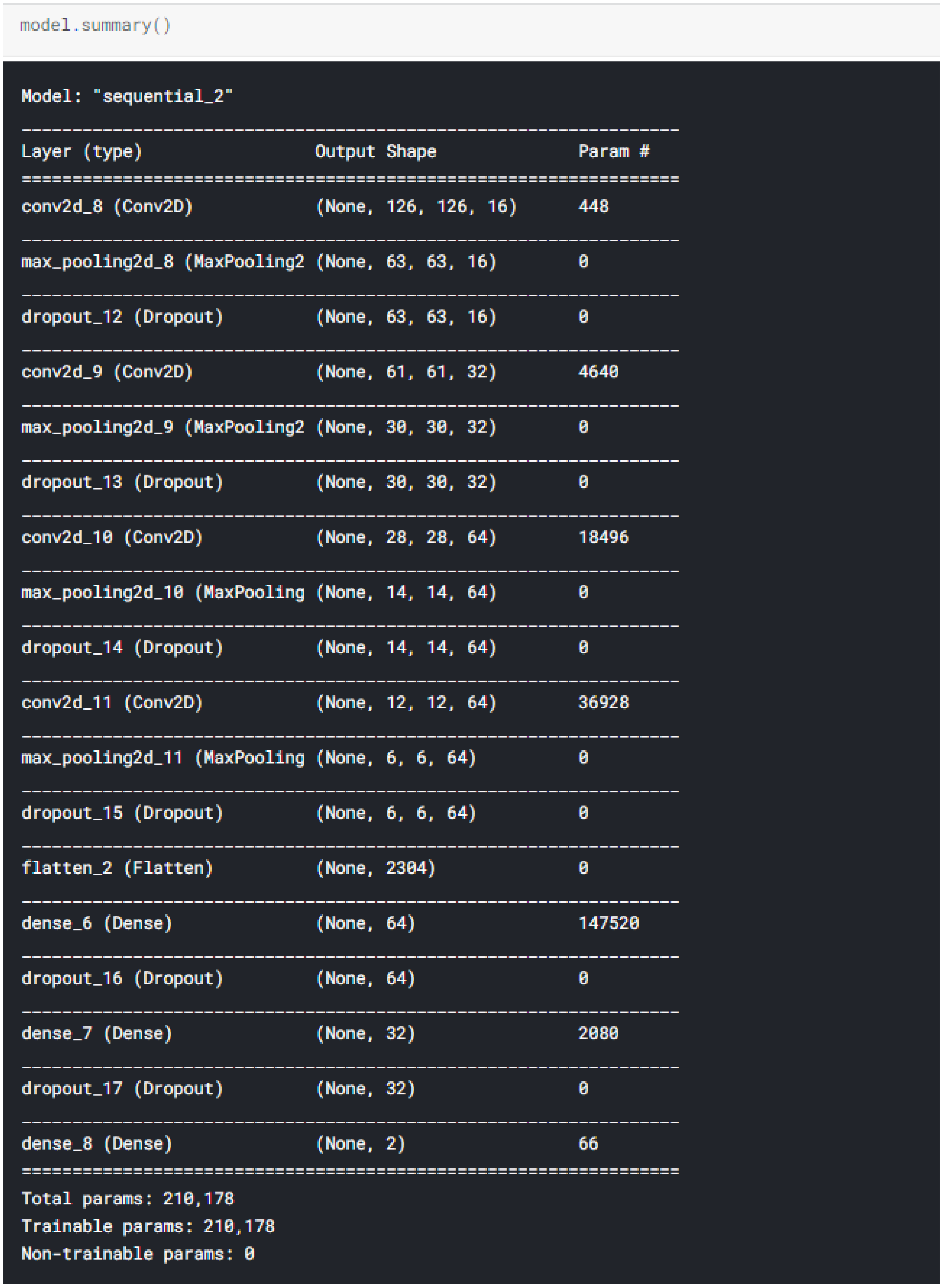
```
Found 5510 images belonging to 2 classes.
```

2) Sample data output





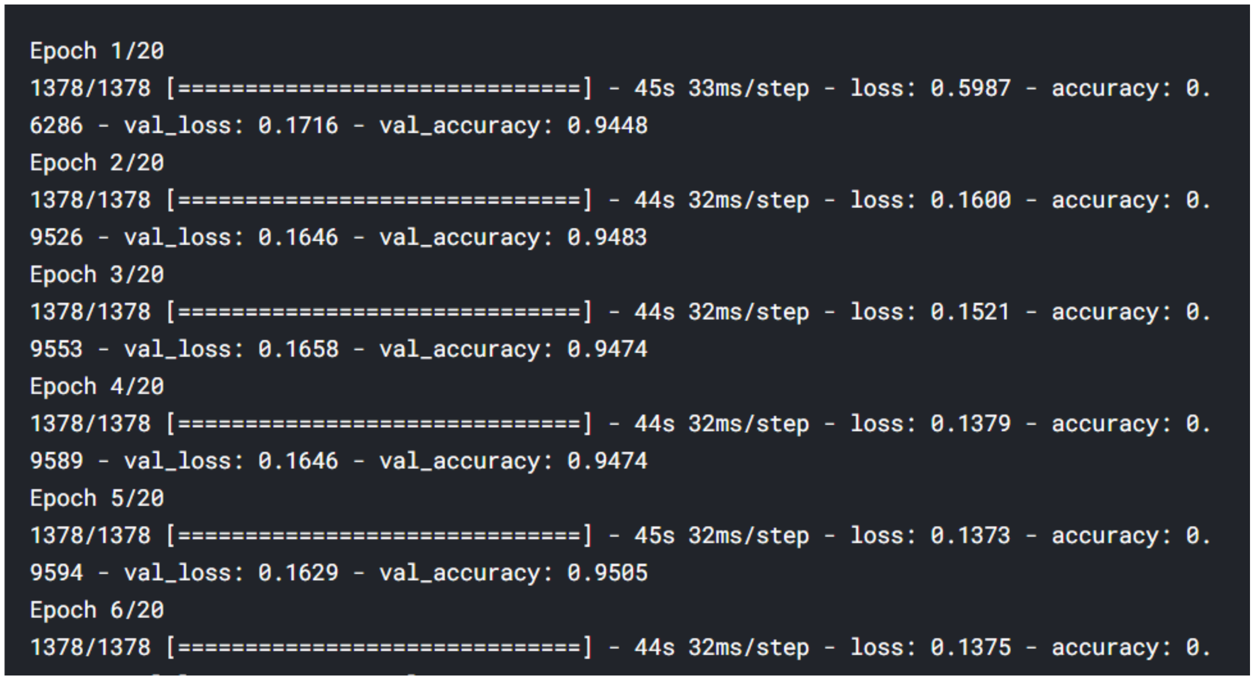
3) Model Creation



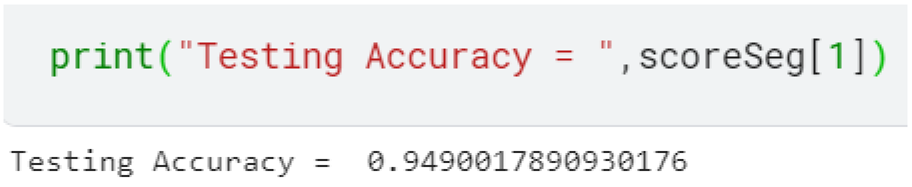
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_8 (Conv2D)            (None, 126, 126, 16)      448       
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 63, 63, 16)        0         
_________________________________________________________________
dropout_12 (Dropout)         (None, 63, 63, 16)        0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 61, 61, 32)        4640      
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 30, 30, 32)        0         
_________________________________________________________________
dropout_13 (Dropout)         (None, 30, 30, 32)        0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 28, 64)        18496     
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 14, 14, 64)        0         
_________________________________________________________________
dropout_14 (Dropout)         (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 12, 12, 64)        36928     
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 6, 6, 64)          0         
_________________________________________________________________
dropout_15 (Dropout)         (None, 6, 6, 64)          0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 2304)              0         
_________________________________________________________________
dense_6 (Dense)              (None, 64)                147520    
_________________________________________________________________
dropout_16 (Dropout)         (None, 64)                0         
_________________________________________________________________
dense_7 (Dense)              (None, 32)                2080      
_________________________________________________________________
dropout_17 (Dropout)         (None, 32)                0         
_________________________________________________________________
dense_8 (Dense)              (None, 2)                 66        
=================================================================
Total params: 210,178
Trainable params: 210,178
Non-trainable params: 0
```

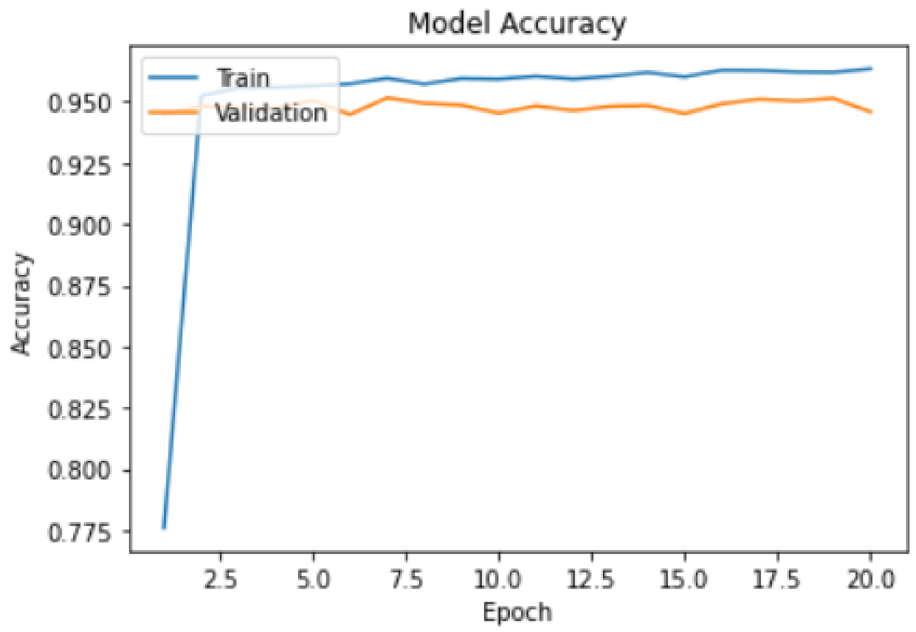
4) Training Model


```
Epoch 1/20
1378/1378 [==============================] - 45s 33ms/step - loss: 0.5987 - accuracy: 0.
6286 - val_loss: 0.1716 - val_accuracy: 0.9448
Epoch 2/20
1378/1378 [==============================] - 44s 32ms/step - loss: 0.1600 - accuracy: 0.
9526 - val_loss: 0.1646 - val_accuracy: 0.9483
Epoch 3/20
1378/1378 [==============================] - 44s 32ms/step - loss: 0.1521 - accuracy: 0.
9553 - val_loss: 0.1658 - val_accuracy: 0.9474
Epoch 4/20
1378/1378 [==============================] - 44s 32ms/step - loss: 0.1379 - accuracy: 0.
9589 - val_loss: 0.1646 - val_accuracy: 0.9474
Epoch 5/20
1378/1378 [==============================] - 45s 32ms/step - loss: 0.1373 - accuracy: 0.
9594 - val_loss: 0.1629 - val_accuracy: 0.9505
Epoch 6/20
1378/1378 [==============================] - 44s 32ms/step - loss: 0.1375 - accuracy: 0.
```

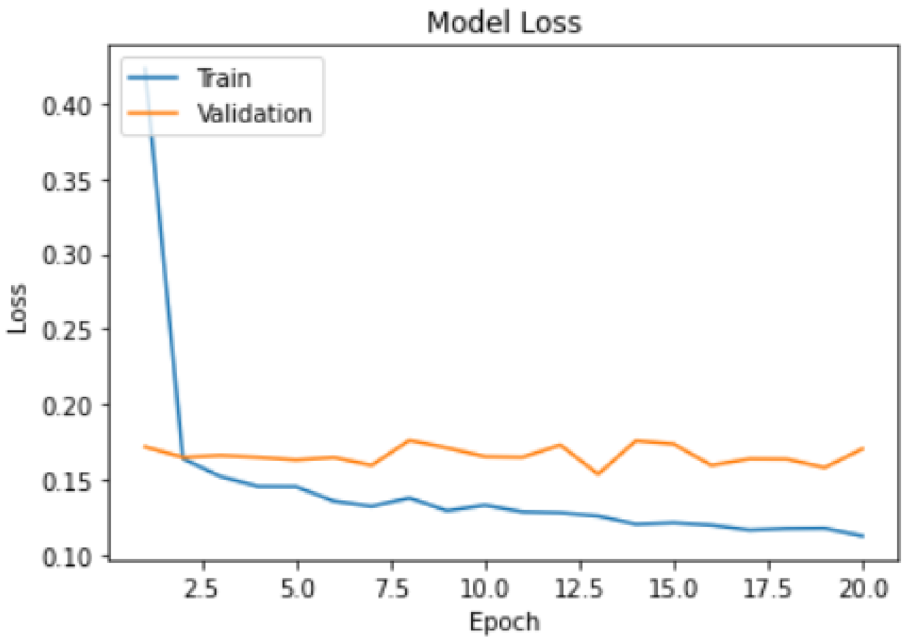
5) Final Validation Accuracy


```
print("Testing Accuracy = ",scoreSeg[1])
```
```
Testing Accuracy =  0.9490017890930176
```

6) Learning Curve



7) Loss curve



Conclusion

As seen from the results, the CNN architecture has been modeled to classify the cell images of the dataset with good performance of **94.9%** Testing Accuracy. Different aspects of the CNN model have been explained and its features have been understood.

References

[1]Rajaraman S, Antani SK, Poostchi M, Silamut K, Hossain MA, Maude RJ, Jaeger S, Thoma GR. 2018. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images.