Capstone Project

RAJESWARI G

sunday, February 17, 2019

I. Definition

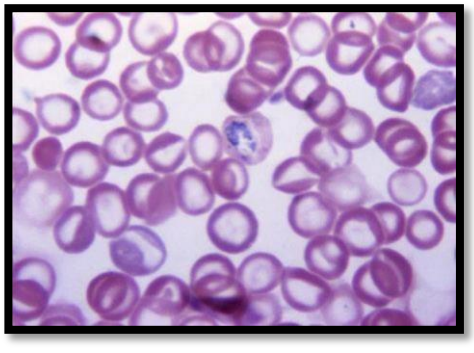
Project Overview

Malaria is a life-threatening disease caused by parasites that are transmitted to people through the bites of infected female Anopheles mosquitoes. It is preventable and curable.

- In 2017, there were an estimated 219 million cases of malaria in 90 countries.
- Malaria deaths reached 435 000 in 2017.
- The WHO African Region carries a disproportionately high share of the global malaria burden. In 2017, the region was home to 92% of malaria cases and 93% of malaria deaths.

Malaria is caused by Plasmodium parasites. The parasites are spread to people through the bites of infected female Anopheles mosquitoes, called "malaria vectors." There are 5 parasite species that cause malaria in humans, and 2 of these species – P. falciparum and P. vivax – pose the greatest threat.

Diagnosis of malaria can be difficult



- Where malaria is not endemic any more (such as in the United States), healthcare providers may not be familiar with the disease. Clinicians seeing a malaria patient may forget to consider malaria among the potential diagnoses and not order the needed diagnostic tests.

Laboratories may lack experience with malaria and fail to detect parasites when examining blood smears under the microscope.

- Malaria is an acute febrile illness. In a non-immune individual, symptoms usually appear 10–15 days after the infective mosquito bite. The first symptoms – fever, headache, and chills – may be mild and difficult to recognize as malaria. If not treated within 24 hours, P. falciparum malaria can progress to severe illness, often leading to death.

Microscopic Diagnosis

Malaria parasites can be identified by examining under the microscope a drop of the patient's blood, spread out as a "blood smear" on a microscope slide. Prior to examination, the specimen is stained to give the parasites a distinctive appearance. This technique remains the gold standard for laboratory confirmation of malaria. However, it depends on the quality of the reagents, of the microscope, and on the experience of the laboratories.

Identifying the Malaria detection by using computer Vision architecture is not much accuracy while finding the malaria cells in the human body. So to overcome this problem then Deep learning came into the picture by using it we can identify the uninfected image cell.

Convolutional neural networks have the ability to automatically extract features and learn filters. In previous machine learning solutions, features had to be *manually* programmed in—for example, size, color, the morphology of the cells. Utilizing Convolutional neural networks (CNN) will greatly speed up prediction time while mirroring (or even exceeding) the accuracy of clinicians.

Reference Link:

https://towardsdatascience.com/detecting-malaria-using-deep-learning-fd4fdcee1f5a

Applications

- Save humans by detecting and deploying Image Cells that contain Malaria or not!

- We can extend this project for the several cell detecting diseases to identify with good accuracy.
- Can be used for research purpose on different disease to identify easily.
- We get around 95% of accuracy in this project,

- In this project we are considering dataset that contain with two folders called

    1. Parasitized (Infected)
    2. Uninfected

  And a total of **27,558** image of Dimension **148 x 148**

Problem Statement

The Aim of this project is to detect the Malaria by using the Cell Image Dataset. In this project I am going to use keras and for improving the Accuracy for detecting the cell Image.

Metrics

As the data I am using is balanced, I want to use Accuracy as a evaluation metric which will be given by the (correct images/total number of images), and loss metric as 'binary_crossentropy' and adam optimizer since our model has two classes for both validation and testing set.

Accuracy = 100 * correct / Total

II. Analysis

Data Exploration

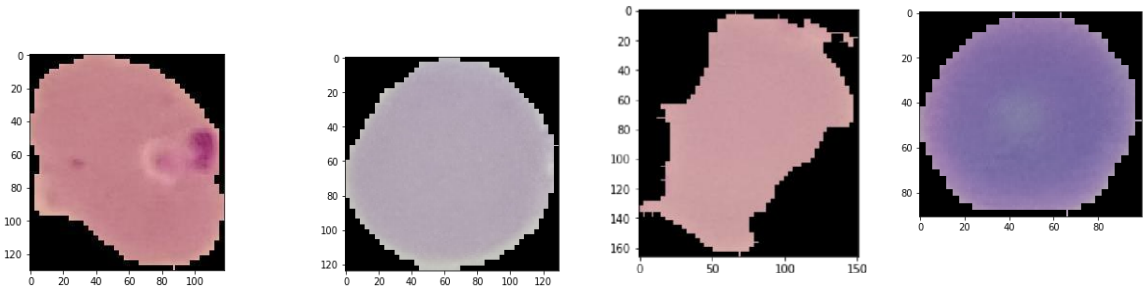
The dataset contains 2 folders:

3. Infected
4. Uninfected

And a total of **27,558** image

- **Dimension 148 x 148**

Reference Link:
https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria



Parasitized Uninfected Uninfected Uninfected

- The dataset contains a total of 27,558 cell images with equal instances of parasitized and uninfected cells. An instance of how the patient-ID is encoded into the cell name is shown herewith: “P1” denotes the patient-ID for the cell labeled “C33P1thinF_IMG_20150619_114756a_cell_179.png”.

The data is a balanced data.

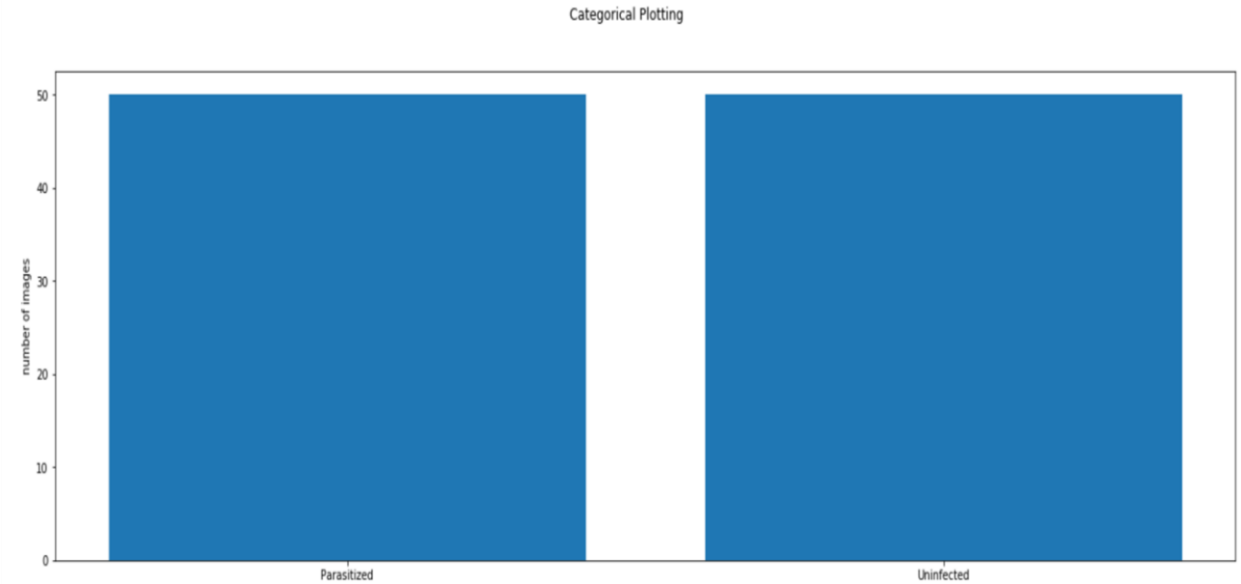
- I made data and labels list where data will be image to array implementation which contains RGB values of each image and label will be class of cells.
- I want to set up the data transformations for each set of data. In general, we want to have the same types of transformations on the validation and test sets of data. However, with the training data, we can create a more robust model by training it on rotated, flipped, and cropped images

Exploratory Visualization

There are 27,558 total images

Since it is a balanced data, dataset is divided into equal parts parasitized and uninfected. The distribution will be as shown above

The data is divided into



Algorithms and Techniques

Deep learning is a subfield of machine learning with algorithms inspired by the working of the human brain. These algorithms are referred to as artificial

neural networks. Examples of these neural networks include Convolutional Neural Networks that are used for image classification, Artificial Neural Networks and Recurrent Neural Networks.

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are everywhere. It is arguably the most popular deep learning architecture. The recent surge of interest in deep learning is due to the immense popularity and effectiveness of convnets. The interest in CNN started with AlexNet in 2012 and it has grown exponentially ever since. In just three years, researchers progressed from 8 layer AlexNet to 152 layer ResNet.

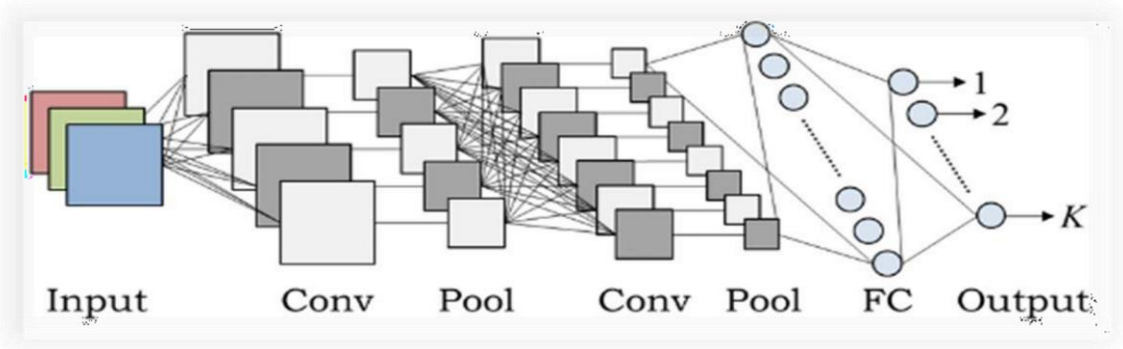
CNN is now the go-to model on every image related problem. In terms of accuracy they blow competition out of the water. It is also successfully applied to recommender systems, natural language processing and more. The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself.

CNN is also computationally efficient. It uses special convolution and pooling operations and performs parameter sharing. This enables CNN models to run on any device, making them universally attractive.

All in all this sounds like pure magic. We are dealing with a very powerful and efficient model which performs automatic feature extraction to achieve superhuman accuracy (yes CNN models now do image classification better than humans). Hopefully ,CNN trains the data efficiently with less training time.

Architecture

There is an input image that we're working with. We perform a series convolution + pooling operations, followed by a number of fully connected layers. If we are performing multiclass classification the output is softmax. We will now dive into each component.



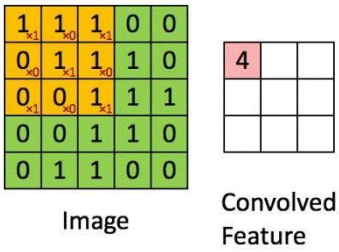
Convolution

The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. In our case the convolution is applied on the input data using a *convolution filter* to produce a *feature map*. There are a lot of terms being used so let's visualize them one by one.

On the left side is the input to the convolution layer, for example the input image. On the right is the convolution *filter*, also called the *kernel*, we will use these terms interchangeably. This is called a *3x3 convolution* due to the shape of the filter.

We perform the convolution operation by sliding this filter over the input. At every location, we do element-wise matrix multiplication and sum the result. This sum goes into the feature map. The green area where the convolution operation takes

place is called the *receptive field*. Due to the size of the filter the receptive field is also 3x3.



Here the filter is at the top left, the output of the convolution operation "4" is shown in the resulting feature map. We then slide the filter to the right and perform the same operation, adding that result to the feature map as well.

Pooling

After a convolution operation we usually perform *pooling* to reduce the dimensionality. This enables us to reduce the number of parameters, which both shortens the training time and combats over fitting. Pooling layers down sample each feature map independently, reducing the height and width, keeping the depth intact.

The most common type of pooling is *max pooling* which just takes the max value in the pooling window. Contrary to the convolution operation, pooling has no parameters. It slides a window over its input, and simply takes the max value in the window. Similar to a convolution, we specify the window size and stride.

In CNN architectures, pooling is typically performed with 2x2 windows, stride 2 and no padding. While convolution is done with 3x3 windows, stride 1 and with padding.

Hyperparameters

Let's now only consider a convolution layer and pooling, and go over the hyperparameter choices we need to make. We have 4 important hyperparameters to decide on:

- **Filter size**: we typically use 3x3 filters, but 5x5 or 7x7 are also used depending on the application. There are also 1x1 filters which we will explore in another article, at first sight it might look strange but they have interesting applications. Remember that these filters are 3D and have a depth dimension as well, but since the depth of a filter at a given layer is equal to the depth of its input, we omit that.
- **Filter count**: this is the most variable parameter; it's a power of two anywhere between 32 and 1024. Using more filters results in a more powerful model, but we risk overfitting due to increased parameter count. Usually we start with a small number of filters at the initial layers, and progressively increase the count as we go deeper into the network.

- **Stride**: we keep it at the default value 1.

- **Padding**: we usually use padding.

Implementation

Structurally the code looks similar to the ANN we have been working on. There are 4 new methods we haven't seen before:

- ***Conv2D***: this method creates a convolutional layer. The first parameter is the filter count, and the second one is the filter size. For example in the first convolution layer we create 32 filters of size 3x3. We use *relu*non-linearity as activation. We also enable padding. In Keras there are two options for padding: *same* or *valid*. Same means we pad with the number on the edge and valid means no padding. Stride is 1 for convolution layers by default so we don't change that.

- ***MaxPooling***: creates a maxpooling layer, the only argument is the window size. We use a 2x2 window as it's the most common. By default stride length is equal to the window size, which is 2 in our case, so we don't change that.

- ***Flatten*:** After the convolution + pooling layers we flatten their output to feed into the fully connected layers

- ***Dropout***: Dropout is used to prevent overfitting and the idea is very simple. During training time, at each iteration, a neuron is temporarily "dropped" or disabled with probability *p*. This means all the inputs and outputs to this neuron will be disabled at the current iteration.

Introduction to keras

It is a Python library for artificial neural network ML models which provides high level fronted to various deep learning frameworks with Tensorflow being the default one.

Why I preferred keras to other Python deep learning libraries

There is an important reason why I prefered keras to other deep learning libraries and I used tensorflow in the backend:

1.Keras along with convultion neural networks takes very less time in training the data and works very efficiently.

2.As , I believed it gave a best accuracy of 95% compared to its benchmark model, where I considered performance of only single CNN layer .

3. Keras has strong multi-GPU support and distributed training support.

4. It is one of the best python libraries for the image classification problems. Which gives best accuracy.

Benchmark

- In this malaria cell detection, I kept benchmark model by adding only one convolution Layer and Max pooling layer and dense layers with 2 units. The accuracy when single layer is added can be seen below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_66 (Conv2D)           (None, 50, 50, 16)        208
_________________________________________________________________
max_pooling2d_65 (MaxPooling (None, 25, 25, 16)        0
_________________________________________________________________
dropout_62 (Dropout)         (None, 25, 25, 16)        0
_________________________________________________________________
flatten_28 (Flatten)         (None, 10000)             0
_________________________________________________________________
dense_54 (Dense)             (None, 500)               5000500
_________________________________________________________________
dropout_63 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_55 (Dense)             (None, 2)                 1002
=================================================================
Total params: 5,001,710
Trainable params: 5,001,710
```

```
2755/2755 [==============================] - 2s 633us/step

 Test_Accuracy:- 0.9328493648021061
```

- Next, I improved the benchmark model accuracy by adding sufficient layers to the model as shown below.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_67 (Conv2D)           (None, 50, 50, 16)        208
_________________________________________________________________
max_pooling2d_66 (MaxPooling (None, 25, 25, 16)        0
_________________________________________________________________
conv2d_68 (Conv2D)           (None, 25, 25, 32)        2080
_________________________________________________________________
max_pooling2d_67 (MaxPooling (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_69 (Conv2D)           (None, 12, 12, 64)        8256
_________________________________________________________________
max_pooling2d_68 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
dropout_64 (Dropout)         (None, 6, 6, 64)          0
_________________________________________________________________
flatten_29 (Flatten)         (None, 2304)              0
_________________________________________________________________
dense_56 (Dense)             (None, 500)               1152500
_________________________________________________________________
dropout_65 (Dropout)         (None, 500)               0
_________________________________________________________________
dense_57 (Dense)             (None, 2)                 1002
```

```
2755/2755 [==============================] - 1s 509us/step

 Test_Accuracy:- 0.9560798548202549
```

As we can observe here, when the model is trained with more number of layers the accuracy of the model has increased which is our ultimate aim.

III. Methodology

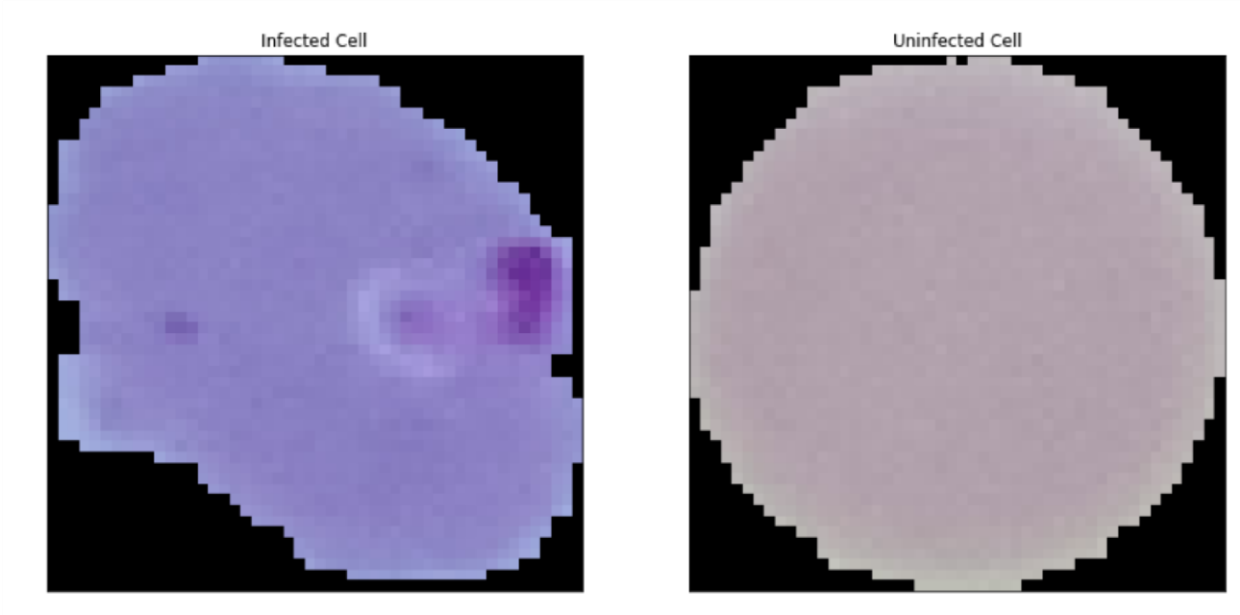
Data Preprocessing

- I made data and labels list where data will be image to array implementation which contains RGB values of each image and label will be class of cells.

- With the training data, we can create a more robust model by training it on rotated, flipped the images with different angles, and cropped images. So, in the data preparation data is resized and flipped in different angles for the better performance of the model to get the best accuracy. The data is divided into training_ set and testing_set.

Implementation

- I used one hot encoding for each class in order to classify them into binary format.
- Next I created a sequential model adding convultion layers and applying CNN techniques such as maxpooling to create a maxpool layer with reduced window size, flatten to pass the output to the fully connected layers and dropout to reduce the overfitting.
- Visualization of cell that is infected and uninfected:



Model Training

Model training is done on both benchmark model and our target model with batch size of 50 and 20 epochs. Once the training starts, it will be as follows

```
24803/24803 [==============================] - 9s 358us/step - loss: 0.6099 - acc: 0.6852
Epoch 2/20
24803/24803 [==============================] - 7s 270us/step - loss: 0.4575 - acc: 0.7939
Epoch 3/20
24803/24803 [==============================] - 7s 268us/step - loss: 0.3199 - acc: 0.8709
Epoch 4/20
24803/24803 [==============================] - 7s 267us/step - loss: 0.2473 - acc: 0.9038
Epoch 5/20
24803/24803 [==============================] - 7s 266us/step - loss: 0.2152 - acc: 0.9198
Epoch 6/20
24803/24803 [==============================] - 7s 266us/step - loss: 0.1880 - acc: 0.9305
Epoch 7/20
24803/24803 [==============================] - 7s 267us/step - loss: 0.1613 - acc: 0.9423
Epoch 8/20
24803/24803 [==============================] - 7s 267us/step - loss: 0.1425 - acc: 0.9490
Epoch 9/20
24803/24803 [==============================] - 7s 267us/step - loss: 0.1199 - acc: 0.9573
Epoch 10/20
24803/24803 [==============================] - 7s 267us/step - loss: 0.1030 - acc: 0.9639
Epoch 11/20
24803/24803 [==============================] - 7s 266us/step - loss: 0.0864 - acc: 0.9713
Epoch 12/20
24803/24803 [==============================] - 7s 267us/step - loss: 0.0716 - acc: 0.9764
Epoch 13/20
24803/24803 [==============================] - 7s 267us/step - loss: 0.0632 - acc: 0.9788
Epoch 14/20
24803/24803 [==============================] - 7s 266us/step - loss: 0.0536 - acc: 0.9822
```

As we can observe here during training loss has decreased gradually and accuracy of our system has increased.

Refinement

• At first model gets an accuracy of 92%,so to improve the accuracy I have changed few parameters such as adding more number of convultion l layers to the model, maxpooling, flattening the data to pass the output o fully connected layers and I applied dropout to reduce the overfitting. Finally I got the accuracy upto 95% for my model which satisfied me.

IV. Results

Model Evaluation and Validation

- In Final model, the accuracy of training samples have nearly 98% accuracy while for validation it is somewhat reduced this is due to there are less number of validation images in given dataset .Any ways the testing accuracy is 95% which can be noted as good model for performance .
- Consider the Loss for training and testing where testing images has low loss that means it is performing good predictions at testing .

Justification

When compared with my benchmark model, my model was found stronger by giving training accuracy up to 98% whereas my testing accuracy is around 95%. The results obtained from my model above satisfactory .

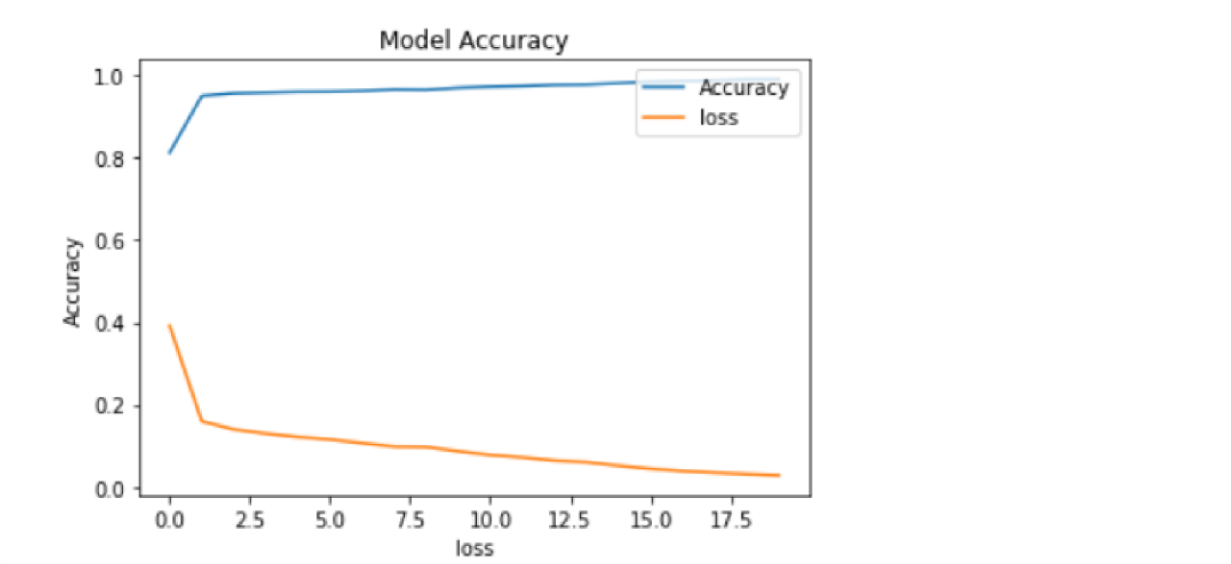
In deep learning we cannot justify that no model can be good at some certain point of view, It just all about improving the performance by add-on new techniques and changing the hyper parameters which will be generate a good outcome that gives some satisfactory to my work.

Now I can confidently say that my still we can improve the solution significant to solve the problem by adding much more cell image data.

V. Conclusion

Free-Form Visualization

You can clearly see the increase in the accuracy of our target model upto 95% and decrease in the loss.



Improvement

Our implementation,TensorFlow has a much larger ecosystem, with important supporting tools like the excellent Keras, a simplified API that makes TensorFlow much more consumable, and TensorBoard, an impressive visualization tool that makes it easy to plot a wide variety of model metrics.

The PyTorch ecosystem isn’t standing still though.

Improving my model better than this may be done by using new techniques such as FastAI Which gets more accuracy by training the model, that’s sounds good to hear when I heard about FastAI. I will be going to learn the FastAI methods how will be going to use it and will improve considering this project accuracy as benchmark and I will implement the FastAI technique for this problem statement.