Malaria Parastite Detection

ML Project (GEN 511)

November 30, 2019

Crooked Three

Sriram G.	Ravi Kiran	S. Purvaj
(IMT2017018)	(IMT2017034)	(IMT2017039)

Contents

1 Introduction **3**
- 1.1 What is Malaria Parasite detection? . . . 3
- 1.2 What is Image Classification? . . . 3
- 1.3 References . . . 3

2 Data **4**
- 2.1 Exploration of data . . . 4
- 2.2 Data Analysis . . . 4
- 2.3 Data Pre-processing . . . 4
- 2.4 Data augmentation(for Deep Learning) . . . 4

3 Feature Extraction **5**
- 3.1 Variation in color and closeness to dark pink . . . 5
- 3.2 Contours . . . 5

4 Model Selection and Building **7**
- 4.1 **Logistic Regression** . . . 7
  - 4.1.1 Theory . . . 7
  - 4.1.2 Feature Selection . . . 7
  - 4.1.3 Training and Testing . . . 8
- 4.2 **Support Vector Machine** . . . 8
  - 4.2.1 Theory . . . 8
  - 4.2.2 Feature Selection . . . 8
  - 4.2.3 Training and Testing . . . 8
- 4.3 **K Nearest Neighbours** . . . 9
  - 4.3.1 Theory . . . 9
  - 4.3.2 Feature Selection . . . 9
  - 4.3.3 Training and Testing . . . 9
- 4.4 **Convolutional Neural Network** . . . 10
  - 4.4.1 CNN1: . . . 10
  - 4.4.2 CNN2: . . . 11
- 4.5 **Residual Neural Network** . . . 13

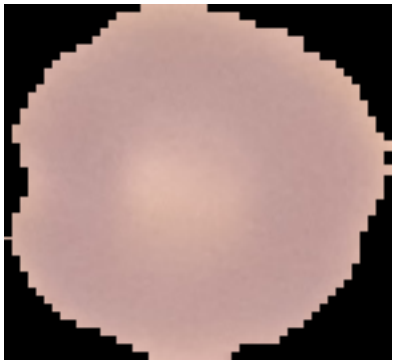
5 Results **16**

1 Introduction

1.1 What is Malaria Parasite detection?

Given images of cells from thin-blood smear, both infected and uninfected, we have to classify the given images into two categories infected(0) or uninfected(1).

(a) Infected



(b) Uninfected

Figure 1.1: Cell images

1.2 What is Image Classification?

Image Classification is the process of assigning a particular *class* to image or probability that an image belongs to the class . Here the number of classes is two and hence it is a **binary classification** problem.

1.3 References

1. Stackoverflow
2. Keras.io
3. Sklearn official documentation
4. Medium
5. OpenCV official documentation

2 Data

The link to the model weights in google drive :
https://drive.google.com/open?id=1azdv9n9fq1uWmyDbZRIrTnI1DzdyK8sC

2.1 Exploration of data

The data consists of two categories infected and uninfected. Each of the category is present in two seperate folders $Uninfected$ and $Parasitized$. Each of the class consists of 11023 images of train data, which sums up the total train data to 22046 images. Test data consists only of images without labels with a count of 1587. These are the images that have to be classified. Each image is in RGB form. Each image is not of the same size and each image has only one cell in it.

2.2 Data Analysis

From the analysis we did regarding the pink shde in some of the images , we figured out that the reason for the abundance of pink in the images was staining of the specimens before putting it under a microscope. Staining helps in penetrating the cell walls and highlighting the cell components. Upon human observation it was found that one of the trivial difference between infected and an uninfected cell image was the presence of a dark spot.

2.3 Data Pre-processing

The first thing we did was to resize all the images to an intermediate size (we resized images that are too small to a bigger size and images that are too large to a smaller size). The range of height varies from 46 to 364 whereas the width varies from 46 to 382.

2.4 Data augmentation(for Deep Learning)

The data augmetation for this problem comprises of rotating the image about 15degrees each, zoom. Also the image was normalized by muliplying by 1/255.0 on each pixel to normalize.

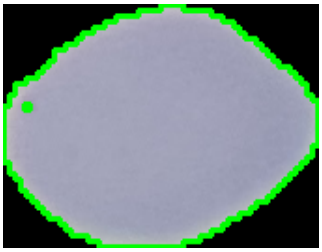
3 Feature Extraction

3.1 Variation in color and closeness to dark pink

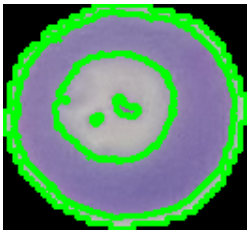
Since we have found out that the main feature of infected cell is the presence of a dark spot we considered two features that would capture the variation in contrast of a pixel from its neighbouring pixels and also the closeness of the pixel to the dark pink color of the spot. Since there were some images with blue spots it was ensured that the variation factor would dominate and both would help to exrtact when there is a oink spot. When there was a varaition from lighter to darker color the closeness to pink color would ensure that there won't be any miss-classification.
First, for each pixel in the image, the *PinkDistance* of the pixel is calculated as the difference in colour of the pixel from dark pink. In other words, the lesser the *PinkDistance*, closer is the colour of the pixel to dark pink. Then for each pixel, we calculated the list of the differences of the *PinkDistance* of this pixel from the *PinkDistance* of it's neighbourhood pixels (eight pixel neighbourhood) and retained the maximum value of these differences as *Variance*. We took the highest of *Variance*'s among all the pixels of an image as a feature of the image.

3.2 Contours

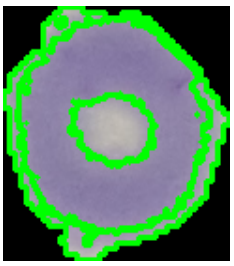
First idea was to see if there are multiple contours in a single image and based on that predict the class. However this did not work because some images had slight variations which the functions in opencv captured and drew contours. For example the below image is a an uninfected image but it has many small contours in it.



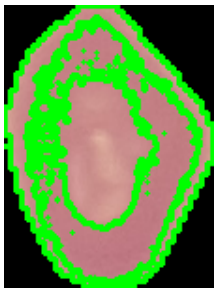
(a) Uninfected



(b) Uninfected



(c) Uninfected



(d) Uninfected

Figure 3.1: Cell images with contours

The next possible feature that could be extracted was the area of highest 2 contours in an image. This would ensure if there is any relation between the area of the cell and area of the possible blob.

4 Model Selection and Building

4.1 Logistic Regression

4.1.1 Theory

Logistic regression is one of the simple binary classifier that is based on a log loss . W is the weight matrix and x is the data matrix. B is the bias.

$$\text{Hypothesis : Z = WX+ b}$$
$$h(\theta(x)) = sigmoid(Z)$$

where sigmoid is

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

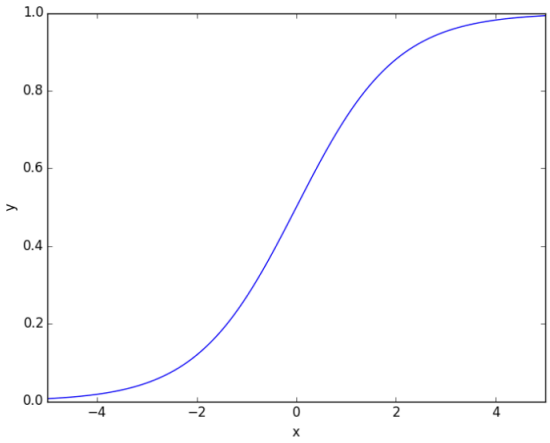


Figure 4.1: Sigmoid function

Loss function for logistic regression is

$$J(h_\theta(x), y)) = -ylog(h_\theta(x)) - (1-y)log(1-h_\theta(x))$$

4.1.2 Feature Selection

For the first model, the variation in color and * closeness to pink were chosen for training. For the second model, the area of 2 maximum contours were chosen as features to train the model. The result of training and testing are given in the following section.

4.1.3 Training and Testing

(Variance)Testing : 89%
(Contours)Testing : 84.036%

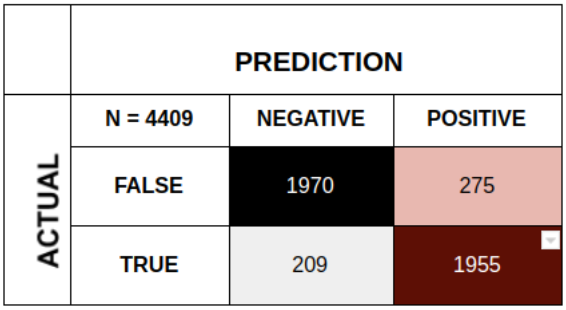


Figure 4.2: Confusion Matrix

4.2 Support Vector Machine

4.2.1 Theory

Support Vector Machine is a simple machine learning model inspired form the kernels. Given data an SVM tries to give the best decision boundary that can fit into the data distribution. It is based on *kernel trick*.

4.2.2 Feature Selection

We reused the same features that were used above to train the two logistic regression to train two SVM models.
The result of training and testing are given in the following section.

4.2.3 Training and Testing

(Variance)Testing : 88.7%
(Contours)Testing : 84.036%

		PREDICTION	
	N = 4409	NEGATIVE	POSITIVE
ACTUAL	FALSE	1983	300
	TRUE	196	1930

Figure 4.3: Confusion Matrix

4.3 K Nearest Neighbours

4.3.1 Theory

The knn algorithm uses the distance of a point from its k nearest neighbours and then tries to see which label it has to be assigned.

4.3.2 Feature Selection

We reused the same features that were used above to train the two logistic regression to train two SVM models.
The result of training and testing are given in the following section.

4.3.3 Training and Testing

(Variance)Testing : 89.15%
(Contours)Testing : 84.036%

		PREDICTION	
	N = 4409	NEGATIVE	POSITIVE
ACTUAL	FALSE	1959	258
	TRUE	220	1972

Figure 4.4: Confusion Matrix

4.4 Convolutional Neural Network

4.4.1 CNN1:

Model description:

This is the first neural network that we trained on this. This neural network is a simple sequential model with four convolution blocks (each convolutional block has a convolutional layer followed by a maxpooling layer) followed by a flattening layer and some dense layers. We added a couple of dropout layers in between dense layers to prevent the model from overfitting to the training data.

We used *relu* as our non-linear activation function in the convolutional layers and in the dense layers except the last dense layer. In the last dense layer, we used *sigmoid* activation since the problem is of binary classification.

Below are the details of the model description and the number of parameters in the model.

Layer (type)	Output Shape	Param #
conv2d_17 (Conv2D)	(None, 198, 198, 32)	896
max_pooling2d_17 (MaxPooling	(None, 99, 99, 32)	0
conv2d_18 (Conv2D)	(None, 95, 95, 128)	102528
max_pooling2d_18 (MaxPooling	(None, 47, 47, 128)	0
conv2d_19 (Conv2D)	(None, 45, 45, 32)	36896
max_pooling2d_19 (MaxPooling	(None, 22, 22, 32)	0
conv2d_20 (Conv2D)	(None, 18, 18, 64)	51264
max_pooling2d_20 (MaxPooling	(None, 9, 9, 64)	0
flatten_5 (Flatten)	(None, 5184)	0
dense_14 (Dense)	(None, 512)	2654720
dropout_6 (Dropout)	(None, 512)	0
dense_15 (Dense)	(None, 256)	131328
dropout_7 (Dropout)	(None, 256)	0
dense_16 (Dense)	(None, 64)	16448
dense_17 (Dense)	(None, 1)	65

Total params: 2,994,145
Trainable params: 2,994,145
Non-trainable params: 0

To compile the model, we used *adam* optimizer and *binary crossentropy* loss function.

Model specific pre-processing:

We did some data augmentation for this model by randomly rotating each image in the range −15deg to +15deg. In addition, we also added a random zoom-in for each image.

Train and test results:

Training Accuracy: 95.57%
Validation Accuracy: 95.1%

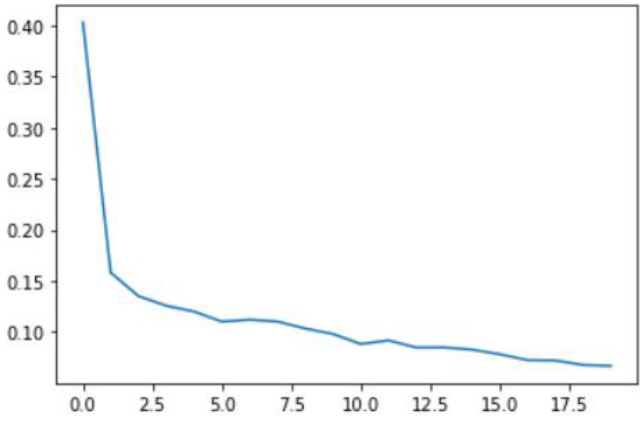


Figure 4.5: Training Loss

4.4.2 CNN2:

Model description:

This is the second neural network that we tried the data on. This neural network is a bit deeper and complex than the previous one. It has three convolution blocks each of which has two convolutional layers followed by a maxpooling layer and batch normalization. Then we added a flattening layer and four dense layers. We added a couple of dropout layers in between dense layers to prevent the model from overfitting to the training data.

This time, we additionally added a dropout layer in between the $2nd$ and the $3rd$ convolutional blocks and a batch normalization between the $2nd$ and the $3rd$ dense layer.

We used *relu* as our non-linear activation function in the convolutional layers and in the dense layers except the last dense layer. In the last dense layer, we used *sigmoid* activation since the problem is of binary classification.

Below are the details of the model description and the number of parameters in the model.

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 198, 198, 64)	1792
conv2d_8 (Conv2D)	(None, 196, 196, 32)	18464
max_pooling2d_4 (MaxPooling2	(None, 98, 98, 32)	0
batch_normalization_5 (Batch	(None, 98, 98, 32)	128
conv2d_9 (Conv2D)	(None, 94, 94, 64)	51264
conv2d_10 (Conv2D)	(None, 90, 90, 32)	51232
max_pooling2d_5 (MaxPooling2	(None, 45, 45, 32)	0
batch_normalization_6 (Batch	(None, 45, 45, 32)	128
dropout_3 (Dropout)	(None, 45, 45, 32)	0
conv2d_11 (Conv2D)	(None, 43, 43, 64)	18496
conv2d_12 (Conv2D)	(None, 41, 41, 64)	36928
max_pooling2d_6 (MaxPooling2	(None, 20, 20, 64)	0
batch_normalization_7 (Batch	(None, 20, 20, 64)	256
flatten_2 (Flatten)	(None, 25600)	0
dense_5 (Dense)	(None, 512)	13107712
dense_6 (Dense)	(None, 256)	131328
batch_normalization_8 (Batch	(None, 256)	1024
dropout_4 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 64)	16448
dense_8 (Dense)	(None, 1)	65

Total params: 13,435,265
Trainable params: 13,434,497
Non-trainable params: 768

The loss function, optimizer we used to compile this model are the same as that of the previous model.

Model specific pre-processing:

We did similar data augmentation of randomly rotating each image in the range -15deg to $+15$deg and random zoom-in for each image.

Train and test results:

Training Accuracy: 96.23%
Validation Accuracy: 95.5%

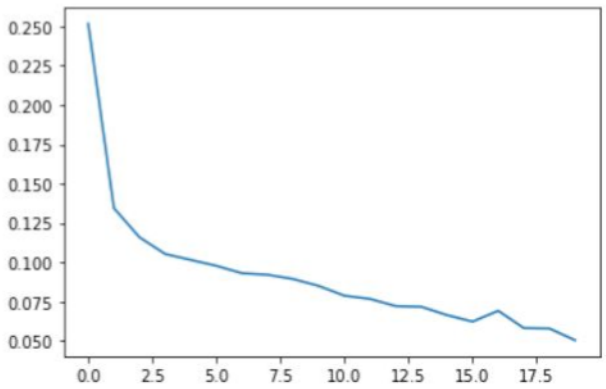


Figure 4.6: Training Loss

4.5 Residual Neural Network

Model description:

The third model we trained the dataset on is a residual neural network. It has two convolution blocks each of which has two convolutional layers followed by a maxpooling layer and batch normalization. We added a third convolutional block which has only one convolutional layer followed by a max pooling block. Then we added a flattening layer and six dense layers. We added a couple of dropout layers in between dense layers to prevent the model from overfitting to the training data.

What makes this model a residual model is the addition of the output of one layer directly to the input of a subsequent layer. In our model, we did this by adding the output of the first dense layer to the input of the third dense layer and the output of the second dense layer to the input of the fourth dense layer.

We used *relu* as our non-linear activation function in the convolutional layers and in the dense layers except the last dense layer. In the last dense layer, we used *sigmoid* activation since the problem is of binary classification.

Below are the details of the model description and the number of parameters in the model.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 200, 200, 3)	0	
conv2d_1 (Conv2D)	(None, 198, 198, 64)	1792	input_1[0][0]
conv2d_2 (Conv2D)	(None, 194, 194, 32)	51232	conv2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 97, 97, 32)	0	conv2d_2[0][0]
batch_normalization_1 (BatchNor	(None, 97, 97, 32)	128	max_pooling2d_1[0][0]
conv2d_3 (Conv2D)	(None, 95, 95, 64)	18496	batch_normalization_1[0][0]
conv2d_4 (Conv2D)	(None, 91, 91, 32)	51232	conv2d_3[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 45, 45, 32)	0	conv2d_4[0][0]
conv2d_6 (Conv2D)	(None, 43, 43, 32)	9248	max_pooling2d_2[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 21, 21, 32)	0	conv2d_6[0][0]
flatten_1 (Flatten)	(None, 14112)	0	max_pooling2d_3[0][0]
dense_1 (Dense)	(None, 512)	7225856	flatten_1[0][0]
dropout_1 (Dropout)	(None, 512)	0	dense_1[0][0]
dense_2 (Dense)	(None, 256)	131328	dropout_1[0][0]
batch_normalization_3 (BatchNor	(None, 256)	1024	dense_2[0][0]
dense_3 (Dense)	(None, 512)	131584	batch_normalization_3[0][0]
add_1 (Add)	(None, 512)	0	dense_1[0][0] dense_3[0][0]
dropout_2 (Dropout)	(None, 512)	0	add_1[0][0]
dense_4 (Dense)	(None, 256)	131328	dropout_2[0][0]
add_2 (Add)	(None, 256)	0	dense_2[0][0] dense_4[0][0]
dense_5 (Dense)	(None, 32)	8224	add_2[0][0]
dense_6 (Dense)	(None, 1)	33	dense_5[0][0]

Total params: 7,761,585
Trainable params: 7,760,929
Non-trainable params: 576

Model specific pre-processing:

We did some data augmentation for this model by randomly rotating each image in the range -15deg to $+15$deg. In addition, we also added a random zoom-in for each image.

Train and test results:

Training Accuracy: 97.5%
Validation Accuracy: 96.008%

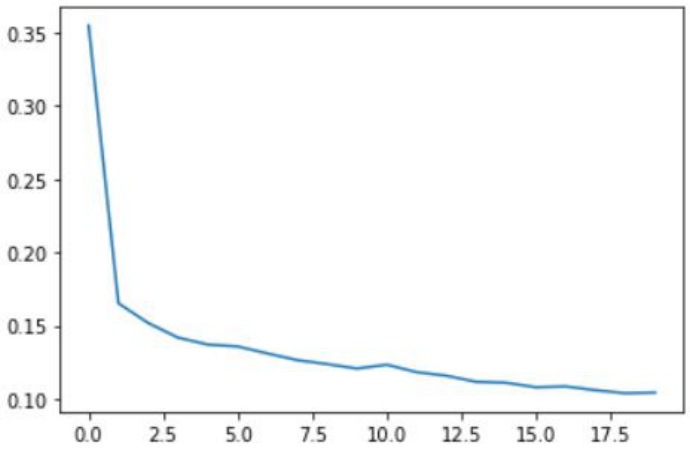


Figure 4.7: Training Loss

5 Results

MODEL	ACCURACY	SUBMISSION ACCURACY
KNN	89%	93%
CNN1	95.57%	93.5%
CNN2	96.23%	95.4%
RNN	97.5%	96.008%