Malaria Detection

PROJECT OVERVIEW: Every year thousands of people lose the lives to malaria especially in developing nations its its a cause of grave concern and a major problem to be tackled.In India we have the technology to tackle this issue but not enough hands on deck the number of doctors to patient ratio is not up to the mark.Here my project comes in I have created a malaria detection model which categorises a cell as a malaria infected cell or not.So as in when the blood diagnosis comes in the cell images would be fed to the model and it would give out results skipping the need of a medical practitioner to step in and they can focus on the treatment process.According to mrcindia there are almost 600-1000 deaths per year in India due to malaria and this disease is a curable disease if caught early on so these many deaths per year is a very high number and we have to take steps to prevent them.
Following are some papers I referred for project:
1)https://arxiv.org/pdf/1703.01976.pdf -International Skin Imaging Collaboration (ISIC) 2017 Challenge at the International Symposium on Biomedical Imaging (ISBI)

2)Sabtain thrun cancer detection work was also very useful.

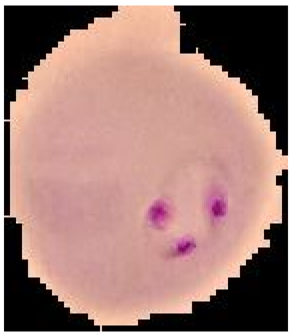
Problem statement: To categorize a blood cell sample as infected with malaria parasite or not on this basis of the cell image.It would be a categorizing problem.

Metrics: Since accuracy is of the highest priority here I propose evaluation metrics to be accuracy based and overall 98% plus accuracy could be considered as outstanding and 95-97% should be very good models with

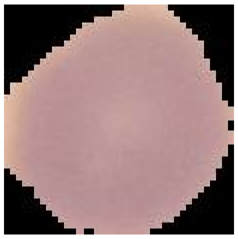
90% plus as acceptable models and below 90%should not be acceptable.I would like to strongly suggest only accuracy as the metric as the data is balanced and would reduce further complications in the project as it has to be as simple as possible at the same time obtaining highest accuracy.

Accuracy=obtained result ÷ original value

Data: The dataset which would be used for this project is obtained from kaggle which contains 2 subfolders of infected and uninfected cell images with a total of 27558 images obtained from the official NIH Website: https://ceb.nlm.nih.gov/repositories/malaria-datasets
These images would be used to teach the machine to find the difference between the 2 categories.the dataset is a balanced as the number of classes are 2 and all the images can be classified easily ny visualizing into 2 categories.The images are in png format and resizing would be required before being fed to the network ideal size would be 224,224,3 so that if required some pretrained models like Resnet could be an open option.



The above pasted image is of a cell infected with malaria if we look closely dark purple spots those are the malaria parasites.



Comparing with a normal healthy cell the purple spots are not present indicating the absence of malaria parasites.So overall the objective here is to teach the model to differentiate the 2 types of cells as it is done by humans.

Algorithms and Techniques: The considered approaches were-
1)Support vector machines:They could have been useful for finding anomalies but the cells and the parasites vary and svms tend to memorize data so it was not chosen for final model.
2)Simple neural networks:If CNNs were not available it would have been the best option.It was dropped because total number of parameters very quickly get too many to be computable
3)CNN'S:Best tool available for image recognition as they are specially designed neural networks for image classification processes.
4)Imagenet's models like resnet 50:They come pretrained out of the box and also contain state of the art CNN architectures.

Finally chosen techniques:Finally the 3rd approach that is creating a whole new CNN architecture for the process specifically reasons being-
1)Large dataset making it easy to create and train a whole CNN based model feasible.
2)Well categorized dataset.
3)High accuracy obtained in first few runs.

Solution statement: To solve this problem I propose the use of CNN's.With the help of CNN the network would learn to read and understand the images and after that a last layer with softmax activation layer 2 cables.Since the number of sample images are in a adequate quantity training CNN is feasible and and best option to obtain highest accuracy possible because in this project accuracy is of the highest priority as wrong diagnosis could be disastrous.

Implementation: Since the images are well separated we just need to assign appropriate labels to the images and perform some basic proproccessing with the help of libraries like numpy,cv2 etc so that the images could be fed to the Convolutional Neural Network with a dense layer on top with softmax activation to help our CNN work as an image classifier.While in this case no data analysis is required as the the categories are well separated.The proposed architecture for CNN is

```
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_7 (Conv2D)            (None, 224, 224, 16)      208       
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 112, 112, 16)      0         
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 112, 112, 32)      2080      
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 56, 56, 32)        0         
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 56, 56, 64)        8256      
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 28, 28, 64)        0         
_________________________________________________________________
dropout_6 (Dropout)          (None, 28, 28, 64)        0         
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 28, 128)       32896     
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 14, 14, 128)       0         
_________________________________________________________________
dropout_7 (Dropout)          (None, 14, 14, 128)       0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 256)       131328    
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 7, 7, 256)         0         
_________________________________________________________________
```

```
dropout_8 (Dropout)          (None, 7, 7, 256)          0
_________________________________________________________________
dropout_9 (Dropout)          (None, 7, 7, 256)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 12544)              0
_________________________________________________________________
dense_3 (Dense)              (None, 500)                6272500
_________________________________________________________________
dropout_10 (Dropout)         (None, 500)                0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                  1002
=================================================================
Total params: 6,448,270
Trainable params: 6,448,270
Non-trainable params: 0
```

Preprocessing: The need of preprocessing is minimal in this project as the the dataset is a very balanced and a clean dataset.The images were stored into 2 different folders.They were brought into the code then given labels accordingly then they were resized to 224,224 because that was the input format the first layer of cnn is expecting and converted into numpy arrays.

Data Sourcing: The data was sourced directly from kaggle with the help of kaggle api.

Work Environment: The work environment used was google's colab notebooks they are the same as jupyter but they come with free GPU/TPU support so they were the reason colab was choosen.

Problems faced: The biggest problem I faced in the project was in the preproccsing.

```
Cells=np.load("Cells.npy")
labels=np.load("labels.npy")
```

Specifically this above mentioned part of code as all of the images when converted into numpy arrays after resizing took a lot of memory to such extent that my pc crashed.Then I turned to cloud options but there also the memory limit would exceed they file cells.npy was exceeding beyond 4 GBs so to tackle this issue I just used a subset of dataset so that it was feasible to train and test the model.Other than this no other major issues were encountered during the implementation.

Refinement: The refinement was not required to a large as the results initially were close to set targets around 90% accuracy but still I added one dense layer with 500 neurons so that the differences could be well distinguished.But overall few refinements were done a few dropout layers were removed from the architecture and that resulted in increased accuracy as it helped the model to retain more.Secondly the initial input layer the kernel size was set to 4 initially and then I reduced it to which also yielded better results.

Benchmark model-
https://www.kaggle.com/ingbiodanielh/malaria-detection-with-fastai-v1
Link to the benchmark model in which 97% accuracy was obtained in the same dataset.Going to the url gives detailed implementation,results and discussion of implementation.

I couldn't obtain same results but got as close as I could.

Results:

Training:

```
Epoch 20/20
9000/9000 [==============================] - 531s 59ms/step - loss: 0.0635 - acc: 0.9774
```

Evaluation:

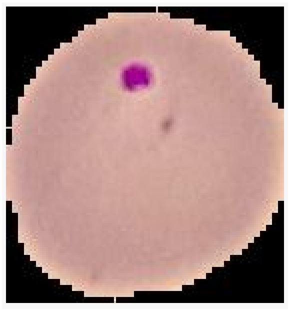
```
accuracy = model.evaluate(x_test, y_test, verbose=1)
```

```
print('\n', 'Test_Accuracy:-', accuracy[1])
```

```
1000/1000 [==============================] - 17s 17ms/step

 Test_Accuracy:- 0.946
```

Example test:`test_model("cell_images/Parasitized/C101P62ThinF_IMG_20150918_151006_cell_80.png")`



```
1/1 [==============================] - 1s 697ms/step
[[1.0000000e+00 5.3829616e-08]]
Malaria detected
```

The model has successfully passed the example test case and comparing to the benchmark model the final accuracy of my model stood at 94.6% and that of benchmark model stood at 97% percent.Therefore both the models could be evaluated on the metrics proposed.

Improvement: The model can be further improved by taking the following steps:
1)Using the whole dataset for training instead of a subset of dataset
2)Pre Trained Imagenet models could be used for the task.
I think using the above mentioned steps would greatly increase the accuracy of the model and it can even surpass the benchmark model.

Conclusion: The project was a success as the desired results were obtained but could be definitely improved if the improvement steps were to be implemented.

Overall this project was a very fun and knowledge imparting project the problem solving part was very interesting as I got to know many could platforms as each of their pros and cons so that next time I can choose the platform according to the task.