Malaria Cells Classifier

Capstone Project

Machine Learning Engineer Nanodegree

Carlos Mertens

April 28th, 2019

I. Definition

Project Overview

The Health-care field of research is one of the fastest growing domain for Machine Learning and Artificial Intelligence owing to the vast collection of data in the sector. Only in the US, the Health-care system generates approximately one trillion gigabytes of data a year.[1]

Morris Panner wrote an article for Forbes where he stated that data is the key to the future of the medical revolution. He also wrote that, the most data we collected and learned from it, the closer we would be coming to discover new treatments and cures.[2]

We also use Machine Learning to prevent or to detect early stages of a number of diseases. For example, researchers from Stanford University have successfully created a Machine Learning Classifier to detect cancer skin. They have slightly surpassed the accuracy of certified dermatologist by training a Convolutional Neural Network Classifier with a dataset of 129,450 clinical images of skin lesions.[3]

By taking advantage of the knowledge and techniques used by the researches from Stanford University, we have also created a classifier that could detect blood cells infected with Malaria parasites.

According to the World Health Organization[4], in 2017 there were 435,000 death due to malaria infections around the world and more than 80% of all deaths occurred in developing nations. A malaria cell classifier we have built, would greatly help doctors from these developing nations by instantly detecting Malaria in the blood cells tested from patients. Therefore by early detection, many affected people can start having the proper treatment avoiding death or possible organ damages due to the infection.

By growing up in the Amazons area of Bolivia, I was constantly exposed to the bites of the mosquito infected with Malaria. Although I have not experienced any death cause by Malaria, I have many friends that were treated late and they have been left with liver complications.

The dataset we have acquired for this project contains 27.558 images of blood cells infected with the malaria parasite and cells that are not infected. They were divided equally in two folders named Parasitized and Uninfected and the data was almost ready to be used to train our classifier by the time we acquired it from the US National Library of Medicine[5]. The data was originally obtained from patients that were diagnosed with malaria and patients that were not diagnosed with it in Bangladesh.

As an input to the classifier, we require the user end to provide an image of a blood cell that they would like to classify it as infected or not.

Problem Statement

The malaria infections could take from 7 to 18 days and even in some cases it has taken a year to develop and to show the symptoms. The initial symptoms could easily be confused by a flu and therefore it could be difficult for doctors to identify the malaria infection without a blood and lab tests. The disease is not fatal instantly, instead most complications are due to the lack of proper treatment on time. For example, in my experience with friends having the disease, the blood test had to be sent to another city in order to be tested for malaria in the lab. By the time the malaria tests came back from the other city, my friends were exposed to the disease for too long which caused them to life with some organs failures.

The machine learning classifier we created, would be very helpful in developing countries to diagnose the patients fast enough in order to avoid further organs complications or death.

We created a command line application to classify an image of a blood cell if it is infected with the malaria parasite or not. The image is provided by the user through a command line such as Command Prompt for Windows operating system or Terminal for Mac or Linux operating system. The application is able to measure the percentage on the blood cell if it is infected or not in order to help a doctor to make a swift decision on applying the proper treatment. Therefore we can be helping hospitals and care centers in developing countries to avoid the delay in the decision to start treatments for malaria in patients. Furthermore, the malaria parasite is adaptable to the environment[6], therefore we created an application that could be portable and the training can be improved or retrained with additional data in order to classify different aspect of the malaria parasites in the future.

Metrics

We analyzed and kept track of the accuracy of the neural network model we trained. Accuracy is an acceptable metric evaluation for machine learning problems, however we wanted to focus on a more complex evaluation metric and one that it is more justify for the problem we were intended to solve.

We evaluated F-beta score which uses a harmonic mean to measure and to keep track of the precision and recall values of a model trained. By using the concept of Confusion Matrix where we store four values (True Positives, True Negatives, False Positives, False Negatives), we could say that we have a problem of a Type Error 1 (Error of the first kind, or False Positive) which In the medical terms, this would be when we misdiagnosed a healthy patient as sick and sent him for more tests. It means that for our problem, it will be okay to classify a healthy cell as infected but not okay to classify an infected cell as healthy.

Confusion Matrix

	Classified Infected	**Classified Healthy**
Infected Cell	True Positives (Correctly Classified)	False Negatives (Type Error2)
Healthy Cell	False Negatives (Type Error 1)	True Negatives (Correctly Classified)

Therefore, we used F-beta score and we set the value of beta (β) closer to 1 in order to get a high recall value which in medical terms would be high sensitivity value.

F-beta Score Formula

$$F_\beta = (1 + \beta^2) * \frac{Precision * Recall}{\beta * Precision + Recall}$$

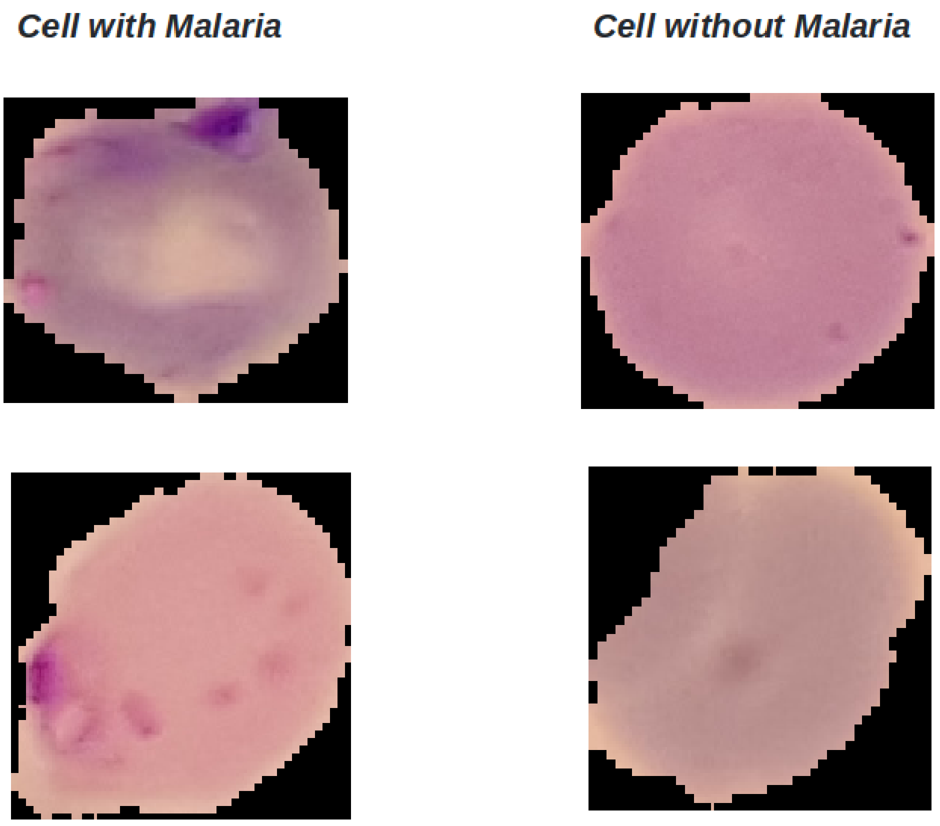
II. Analysis

Data Exploration

As we mentioned before, the dataset of blood cell images infected and not infected with malaria parasites were acquired from the US National Library of Medicine (NIH) website. The images were capture by a built-in camera of an Android phone attached to a conventional light microscope and carefully annotated by experts[5].

The dataset has been previously cleaned and labeled by separating the infected and the healthy examples in different folders; Parasitized folder for the infected examples and Uninfected folder for the healthy examples. Each folder contained 13,779 images of their respective label and are saved in a .png format (Portable Network Graphics).

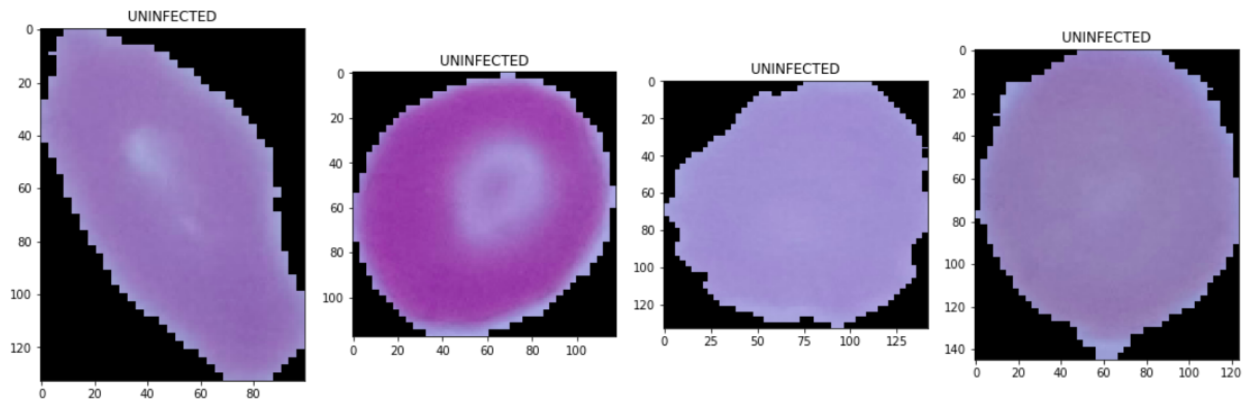
As an input, we require the user-end to provide the path of the image that they would like to pass through our classifier in order to detect if the image contains an infected cell or not. As we mention before, the image has to be provided through the command line as we intended solution is a command line application. The image does not need to be preprocessed by the user and the format can be any image format.



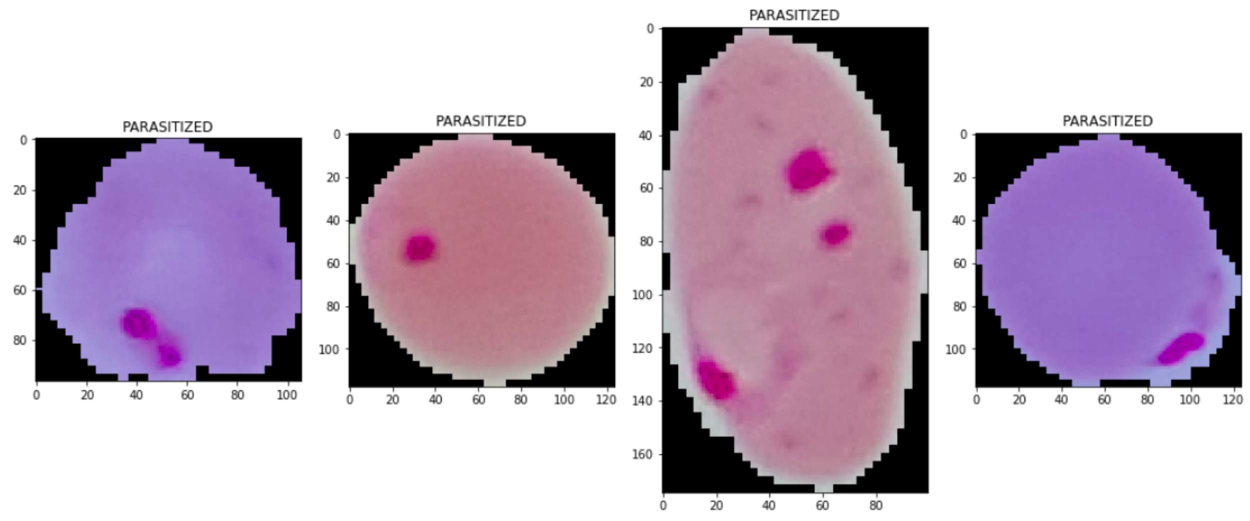
Exploratory Visualization

We used a Jupyter Notebook environment to work on the technical part of the project and we created a visual content to have a better idea of the images in the dataset we were going to use:

```
Original shape: (133, 100, 3)
Original shape: (118, 118, 3)
Original shape: (133, 142, 3)
Original shape: (145, 124, 3)
```



```
Original shape: (97, 106, 3)
Original shape: (118, 124, 3)
Original shape: (175, 100, 3)
Original shape: (118, 124, 3)
```



Once again, we have to thank to the US National Library of Medicine (NIH) website for making the dataset available for research and academy use. The images are already labeled by separating them into different groups (folders) and this separation was conducted by experts at the Mahidol-Oxford Tropical Medicine Research Unit in Bangkok[5].

However, we can notice that the size of each image is different from one another and therefore we had to resize them all in order to have a uniform size of images in the dataset.

We created the visualization because we wanted to also have a look into some of difference between the images of infected cells and not infected cells. The human eye could possible identify the pigmentation in the cells blood and give a estimate guess of the infected cells blood. However, there is some discoloration and shadows in some of the healthy cell images that could easily be confused by the human eye. Therefore, the neural network would be able to break down the pixels and find a pattern on those discoloration or shadows and make a more accurate estimation of which blood cells are actually infected.

Algorithms and Techniques

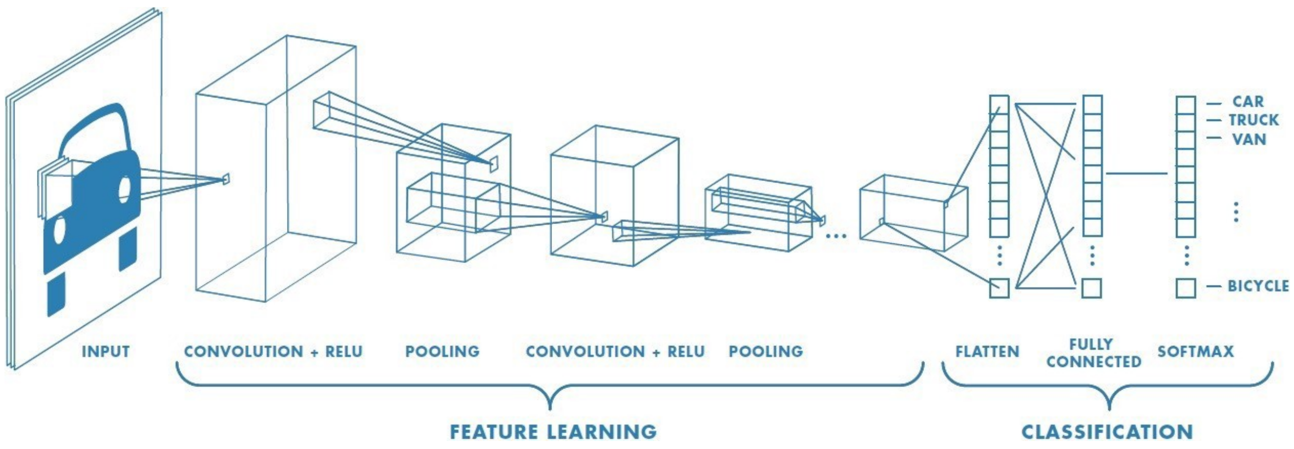
We used a deep learning algorithm named *convolutional neural network (CNN)* to build our classifier. This algorithm works perfectly with images converted into matrixes using the pixels as values.

To understand how CNN algorithms work, we need to first explain the idea of *local connectivity*. Locally connected layers algorithms use far fewer parameters than a densely connected layer algorithms and therefore they save us a lot of time and computational power. The idea is that local connected algorithms convert the image in a matrix and divide it in different areas. Every area is connected to its specific *node* and the node only learns about that area designated. Then the nodes report to the *output layer* where the information learned is combined. The number of nodes can be expanded creating a *collection of hidden layers* where every collection will be responsible for learning a specific area of the image.

By using the idea of the locally connected layer, *convolutional layer algorithms* were created. In a CNN algorithm, we select the width and height of the convolutional window which are the hyper-parameters represented on a grid called a *filter*, and then the window is slided vertically and horizontally on the matrix. At each position, the window defines a small section of pixels in the matrix and connects it to a specific single hidden node. Then the collection of these nodes formed a *convolutional layer*. We control the behavior of the convolutional layers by specifying the number of filters and the size of convolutional windows or *kernels*. If we want to find more patterns in the images, we can use more filter to learn better the pixels in the images. Every filter will be a *collection of nodes* with different weights and bias. This collection of nodes are called *feature maps* or *activation maps*.

Color images like the one we have in the dataset and the image we require as an input, are interpreted as 3D array with height, width and depth (RGN), and the depth value is 3. It is considered as a stack of 3 two-dimensional matrices, hence the filter or the convolutional window needs to be also a 3 dimensional grid.

We also have the *hyper-parameters* of the *stride* of the kernel which is the amount by which the filter slides over the image. A stride value of 1 makes the convolutional window the same width and height as the input image. But if we set the value to 2, when we do not have too many details in the image, then the convolutional layer will differ from the input images. However the filters will fall outside the image input, so we then need to also set *padding* to create another dummy column in the image input to have the same size of the convolutional window.
We have obtain a nice image to illustrate a convolutional neural network[7].



Benchmark

As we mentioned on the project proposal, initially we thought about using as a benchmark the classifier from the Stanford University on the skin cancer detection. They have obtained an overall accuracy of 72% ± while professional dermatologists had an accuracy of 65% ± by looking at the same infected blood cell images. However, we have to consider that the project from Stanford University is classifying several type of skin cancer and not just one the simple type of classification we needed to solve our problem.

Nevertheless, we have found some Kaggke kernels- “a cloud computational environment that enables reproducible and collaborative analysis” [8] – that could reach as high as 97%

accuracy on their models. We have chosen as our benchmark a simple but well structured kernel create by Rohit Pawar[9] where he had reached an accuracy of around 95%.

However as we mentioned before, the accuracy was just part of our metrics evaluation because we wanted to exclusively focus on the F-beta score. We looked again into the project from Stanford University on detecting skin cancer detection. They scored 91% by using a comparison metric between *sensitivity* which is *recall* and *specificity* which is a little similar, but not the same as *precision*[3]. In any case, we have set our goal to have the best possible score in sensitivity or recall in order to make sure that we would not miss or send home patients that are actually sick. Therefore we have chosen F-beta metric with a value to obtain the best approximation to high recall or sensitivity.

III. Methodology

Data Preprocessing

The dataset was acquired already separated into two folders with the images of healthy and sick blood cells. However, further preparation was needed in order to have it ready for training the classifier and this are the steps taken:

1. We downloaded the dataset and placed both folders (Uninfected and Parasitized) into a folder named *Data* located in the project directory.
2. We created in the Data directory another folder named Input-Data with two sub-folders named Healthy and Infected. We have chosen five random images from the Uninfected folder and placed them into the Healthy folder. We did the same with the infected images from Parasitized folder. We took five random images and then we placed them into the Infected folder. We did this in order to have some images as input for testing the finished command line application with images that the network had not seen before.
3. As we saw in the data visualization, all the images came in different sizes therefore it would not be possible to feed them to the convolutional neural network. We decided to standardized and resized all images to *224 x 224 x 3*. We have chosen a high resolution pixels thinking into the future in case that we want to re-train the algorithm with more modern images or more classification options.
4. So we looped into the Uninfected and Parasitized folders and resized each image and saved them into an empty *list called features* while in parallel we saved their

respective label into another empty *list called labels*. We had denominated the labels as *binary representation* of 0s and 1s; for the uninfected or healthy cells 0s and for the parasitized or infected cells 1s.

5. All the images in the features list were converted into arrays or matrix and we did the same for all the labels in the labels list.
6. Finally, as it is standard practice in machine learning, we separated a percentage of all the images processed and their respective labels for training and testing the model. We have chosen *25% of all images* to be used for testing. Therefore when the testing was done, the classifier was not bias with the never seen data. The labels for the training and testing were also converted into *binary category*.

Implementation

Once the data was preprocessed and ready for training and testing a convolutional neural network classifier, we proceed to build the architecture of the classifier. As we have mentioned many time before, we wanted to build a classifier thinking that in the future the application could be improved by training the classifier with more sophisticated images or even adding more type of classification (parasites, viruses, etc). Therefore we built a sequential model with 4 *feature learning connected areas or batch* before reaching to the *classification area*.

Feature Learning Layers (batch 1):

- Convolutional layer with 16 filters, a kernel size of 2 x 2, padding for sliding kernel, activation ReLu (Rectified Linear Unit) which is the most used and popular activation available, and the input size of 224 x 224 x 3 which is the size of the processed images.
- MaxPooling layer to reduce the dimensionality of the image by half the size. Hence we set the pool size equal to 2.
- BatchNormalization layer to normalize and activate the reduced image from the previous layer. This layer helps to reduce the sensitivity and to improve the stability of the network.
- Dropout layer to prevent overfitting (memorization of the data) of the learning by randomly removing a percentage of the data to be learned. We have chosen 20% of the data to be dropped.

Feature Learning Layers (batch 2):

- Convolutional layer with 32 filters which is the standard practice to double it from the previous batch, the kernel size does not change, padding for the kernel and activation ReLu. This time we did not need to set the input size since it is connected to the output of the previous layer.
- Same MaxPooling layer as the previous batch
- Same Dropout layer as the previous batch

Feature Learning Layers (batch 3):

- Similar layers as the previous batch with the exception of doubling the filters to 64.

Feature Learning Layers (batch 4):

- Similar layers as the previous batch with the exception of doubling the filters to 128.

Please note that if we need to train the classifier with more detailed images, we could always add another batch of layers similar to the previous but doubling the filters.

Classification Layers:

- GlobalAveragePooling layer to minimized overfitting and to reduce the dimension of the image processed on the previous feature learning layers by averaging the feature maps and flattening the data.
- Dense layer to perform the classification of the features maps or activation maps. We have chosen the size of 1024 units or dimensionality of the output space. We also have chosen an activation ReLu for this layer.
- Dense layer to final classification of the images. The size unit is set to 2 in order to classify the image as 0 (healthy) or 1 (infected). The activation used was *softmax* which is the most used and the standard for two classes classification.

Optimizer: We used "*adam*" as optimizer since it has been proven[10] to be a more stable, faster learner, and does not loses mayor decreases in accuracy.

Loss: We used "categorical_crossentropy" to measure the performance of the model classifier since the desire output would be a probability value between 0 and 1 which will be interpreted as 0% and 100% accuracy.

Refinement

In order to get to the architecture described above, we have done many tests with different parameters:

- We have tested the network with less and more feature learning batches where we had gotten less accurate models or a little better accurate model but extremely long training time and computational power consumption.
- We have tried with an initial number of filters to 8 and doubling them as standard practice but we found it to be a little faster but less accurate. We needed to perform more training round or epochs in order to improve the accuracy and F-beta score. Since the images of cell blood do not have so many features to describe we thought that we would do better with 16 filter to start the training in case we have better data in the future.
- The first Dense layer was tested with 400, 500, 512, 600, 750 and 1024 units. As the units increase, the training time and computational power increase. However, as the units increase, we also see the increase in the accuracy and F-beta score. Since we built this model with the future implementation in mind, we decided to go with high number of units (1024) for more features detection if we have better images and more classifications.
- Dropout percentage was tested on 0.15 or 15% of the data to be dropped in each layer. However, since the dropout is to reduce overfitting, by experience, I have set my personal standard to .20 or 20% and since the 0.15 value did not make much difference in general, I decide to stay with value of 0.20

IV. Results

Model Evaluation, Validation and Justification

In the training and testing section of the project, we have performed many tests with different architecture and parameters and many epochs (running cycle tests) as high as 50 which took a lot of time and computational power. At the end, we were satisfy with the result of the final model trained in 20 to 25 epochs. On those training and testing, the accuracy was compared to the benchmark we had set for the project and the F-beta score was also compare to sensitivity vs specificity score by the Stanford University project to detect skin cancer.

The final classifier we trained was able to perform at an accuracy of 94% to 96% which is approximately about the same accuracy obtain by our benchmark. Although we know some other projects in Kaggle had reached 97% to 98 %, we were satisfied and confident on the accuracy we reached.

The F-beta score that the classifier obtained was between 0.93 and 0.95 while the skin cancer classifier obtained a 0.91 in their sensitivity vs specificity metric test. Since the F-beta score was our most important metric to follow, we are very confident with the result and the comparison against the skin cancer classifier. We have set the value of beta to 0.90 because we wanted to have a high sensitivity (recall) score and we believe we have reached a very respectable score (0.93 – 0.95).

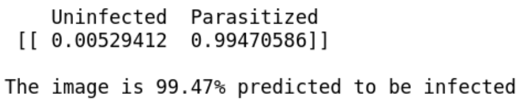
In resume, we are confident in the metric evaluation compare to the benchmark. At the same time the results obtained by our classifier are exactly on the personal goals we were trying to reach.

We have also tested the final classifier with the images we separated in the Input-Data folder. The classifier gave us a remarkable results on predicting the input images. Same of the results were at a high accuracy such as 95% to 99% prediction to be infected on actual infected samples and 1% to 10% prediction to be infected on images of healthy samples.

Therefore, the final architecture and parameters for the classifier are aligned with the result we were expecting. After many tests with different architecture and parameters as mention above, we are confident that we have built a great classifier, especially after seeing how it performed with images never seen before.

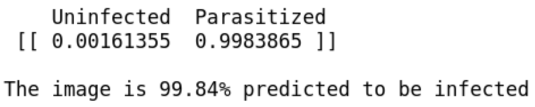
Actual infected images tested

Test 1



```
    Uninfected  Parasitized
[[ 0.00529412  0.99470586]]

The image is 99.47% predicted to be infected
```

Test 2



```
    Uninfected  Parasitized
[[ 0.00161355  0.9983865 ]]

The image is 99.84% predicted to be infected
```

Actual Non-infected images tested

Test 3

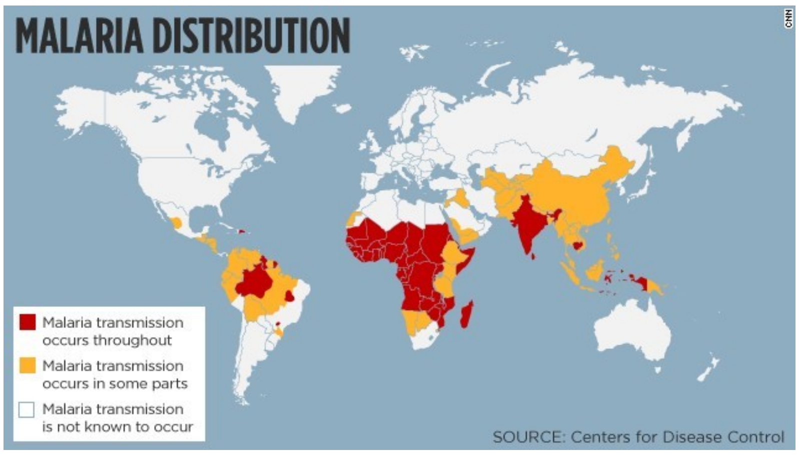
```
   Uninfected  Parasitized
[[  9.99866843e-01   1.33153109e-04]]

The image is 0.01% predicted to be infected
```

Test 4

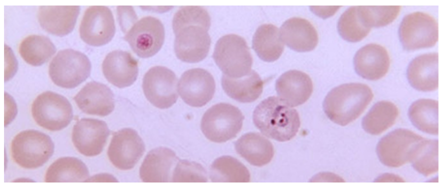
```
   Uninfected  Parasitized
[[ 0.99875891  0.0012411 ]]

The image is 0.12% predicted to be infected
```

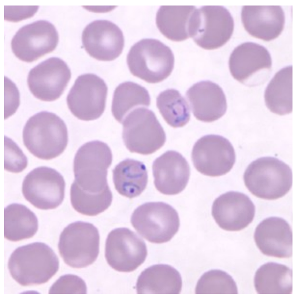
V. Conclusion

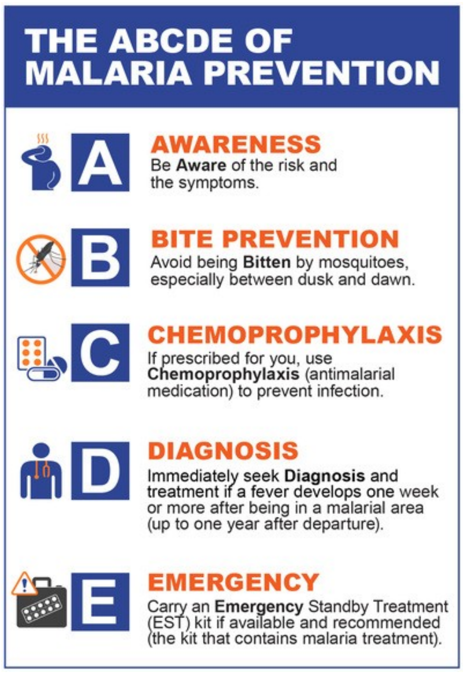
Free-Form Visualization


MALARIA DISTRIBUTION
CNN
Malaria transmission occurs throughout
Malaria transmission occurs in some parts
Malaria transmission is not known to occur
SOURCE: Centers for Disease Control

As we can see in the map above, the most affected areas in the world is in developing nations. The infection is transmitted by mosquito bites and it can be detected in the blood cells.







THE ABCDE OF MALARIA PREVENTION

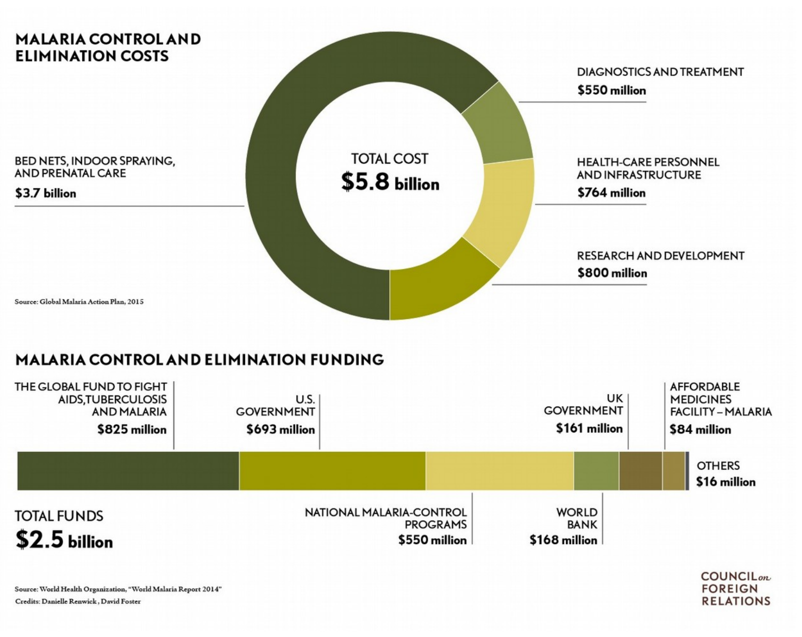
A **AWARENESS** Be **Aware** of the risk and the symptoms.

B **BITE PREVENTION** Avoid being **Bitten** by mosquitoes, especially between dusk and dawn.

C **CHEMOPROPHYLAXIS** If prescribed for you, use **Chemoprophylaxis** (antimalarial medication) to prevent infection.

D **DIAGNOSIS** Immediately seek **Diagnosis** and treatment if a fever develops one week or more after being in a malarial area (up to one year after departure).

E **EMERGENCY** Carry an **Emergency** Standby Treatment (EST) kit if available and recommended (the kit that contains malaria treatment).



Some of the prevention methods and cost to try eradicating or controlling the infection. In my opinion, it is almost impossible to completely eradicate malaria. Malaria has been found even in thousands years old mummies. The best way to try to eradicate would be by early detection and early treatment to avoid the infection to be spread again. However, as we mention before, developing nations affected by the disease can not have the resources to perform the lab test to all their patients and probably would take them days to get the results.

Reflection

After many tests with different architecture and parameters, we are confident that we have created a great classifier application, especially when we think that this classifier could be retrained in the future with better data. We believe we have a blue print for a classifier that could learn from blood cells images in order to detect patterns of many other diseases and viruses. We trained the classifier with a high sensitivity because we wanted the application to be a tool for doctors with scarce resources but not the final say in patients. We want the doctors to take the decisions with the help of the classifier’s prediction. We have conducted a laborious research in order to create what it looks like a simple application but at the same time a wonderful tool to help doctors.

We had a great boosting from the beginning by acquiring professional labeled images. We just needed to make a small analysis on the data, mostly to familiarize ourself with the images we were going to work with. We did perform some preparation on the data such as resizing all of them to a uniform size, giving them a binary label and splitting the data for training, validating, testing and samples to use as input. We converted them into tensors to be able to pass them through the neural network classifier.

We built an architecture with parameters that gave us great satisfaction. We tested the model with many different architectures and parameters until we came up with the one that preformed great in the data we have but also with the idea that in the future we could retrained the classifier with better data or with more classifications of diseases.

We validated and tested the classifier trained with - what we believe - the better metric evaluation for a health care domain in machine learning, F-beta score set to high precision. We obtained highly satisfying results in comparison with the benchmarks we proposed. The F-beta score was better result than the benchmark and the accuracy scores around the same as the benchmark. We tested the final model with the data we had separated to be used as input samples and we obtained extraordinary accuracy on the predictions.

Finally, we created two Python files - train and predict - in order to be used as an command line application. Therefore our project is ready to be exported and to be adapted into a web application or a phone application. We left the possibility open to retrained the classifier with better data or with more classification diseases such as viruses, parasites.

Overall, we are very confident and very please with the result of this project. We did not encounter any mayor difficulties and the most interesting aspect of the project was to see the wonder of convolutional neural network algorithm.

Improvement

As we mentioned constantly through this report, we build this project with the idea that in the future could be improved with new and better data or it could even be modified to learn and classify more blood diseases.

However as satisfying the results in this project has been, we could have used some other techniques in order to further improve it, for example:

- We could implement a grid-search to find the best parameters for the architecture we used.
- We could have used complexity graph in order to visualize the learning curve to show when the data was overfitting and be able to choose the best number of epochs.
- We could have used transfer learning and load a pre-trained model in order to save time and computational power.
- We could have applied an augmented image generator to the data in order to avoid further overfitting.

In general, there is always room for improvement in developing applications since there are constantly new techniques to help us build a better world with Machine Learning.

Note

For a technical report and visualization of the code, open the Jupyter Notebook “malaria.ipynb” or the web file “malaria.html” in the project directory.

To run the application use the python file “predict.py” in the command line by typing “python predict.py –input <path to blood cell image>”

Reference

1. David Champagne, Sastry Chilukuri, Martha Imprialou, Saif Rathore, and Jordan VanLare (2018), “Machine Learning and Therapeutics 2.0: Avoiding hype, realizing potential”. *McKinsey & Company* [Online], Available at: https://www.mckinsey.com/industries/pharmaceuticals-and-medical-products/our-

insights/machine-learning-and-therapeutics-2-0-avoiding-hype-realizing-potential [Accessed on 1 April, 2019]

2. Morris Panner (2018), “Health data meets Artificial Intelligence and Machine Learning”. Forbes [Online], Available at: https://www.forbes.com/sites/forbestechcouncil/2018/11/21/health-data-meets-artificial-intelligence-and-machine-learning/ [Accessed on 1 April 2019]

3. Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, Sebastian Thrun, “Skin cancer classification with Deep Learning”. nature [Online], Available at: https://cs.stanford.edu/people/esteva/nature/ [Accessed on 2 April 2019]

4. World Health Organization [Online] “Global Health Observatory (GHO) Data”, Available at: https://www.who.int/gho/malaria/epidemic/deaths/en/ [Accessed on 2 April 2019]

5. U.S. National Library of Medicine [Online], Available at: https://ceb.nlm.nih.gov/repositories/malaria-datasets/ [Accessed on 3 April 2019]

6. Martin Rono (2017), “How the malaria parasites adapts when faced with different opportunities for transmission in its natural enviroment”. Kemri – Welcome Trust [Online], Available at: http://kemri-wellcome.org/blog-post/how-the-malaria-parasites-adapts-when-faced-with-different-opportunities-for-transmission-in-its-natural-environment/ [Accessed 3 April 2019]

7. Sumit Saha, “A comprehensive guide to Convolutional Neural Networks – the ELI5 way”. Towards Data Science [Online]. Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [Accessed on 27 April 2019]

8. Documentation, “How to use Kaggle”. Kaggle [Online], Available at: https://www.kaggle.com/docs/kernels [Accessed 4 April 2019]

9. Rohit Pawar, “Malaria cells classification through Keras”. Kaggle [Online], Available at: https://www.kaggle.com/sharp1/malaria-cells-classification-through-keras [Accessed on 4 April 2019]

10. David Mack, “How to pick the best learning rate for your machine learning project”, Medium [Online blog]. Available at: https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2 [Accessed on April 14 2019]