Detecting Malaria Using Deep Learning

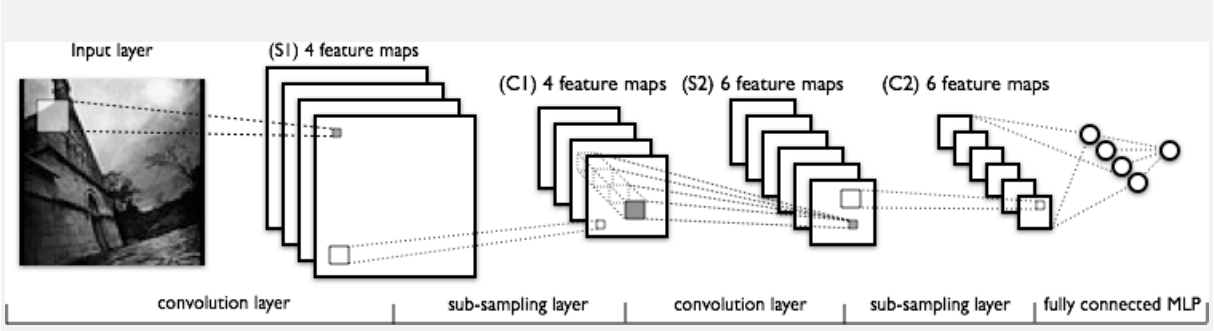
-By Abhishek Birajdar

Malaria is a deadly, infectious mosquito-borne disease caused by Plasmodium parasites. These parasites are transmitted by the bites of infected female Anopheles mosquitoes. The motivation for this project is however based on the nature and fatality of this disease. Initially if an infected mosquito bites you, parasites carried by the mosquito will get in your blood and start destroying oxygen-carrying RBCs (red blood cells). Typically, the first symptoms of malaria are similar to the flu or a virus when you usually start feeling sick within a few days or weeks after the mosquito bite. However, these deadly parasites can live in your body for over a year without any problems! Thus, a delay in the right treatment can lead to complications and even death. Hence early and effective testing and detection of malaria can save lives.

Deep Learning for Malaria Detection

With regular manual diagnosis of blood smears, it is an intensive manual process requiring proper expertise in classifying and counting the parasitized and uninfected cells. Typically, this may not scale well and might cause problems if we do not have the right expertise in specific regions around the world. Some advancements have been made in leveraging state-of-the-art (SOTA) image processing and analysis techniques to extract hand-engineered features and build machine learning based classification models. However, these models are not scalable with more data being available for training and given the fact that hand-engineered features take a lot of time.

Deep Learning models, or to be more specific, Convolutional Neural Networks (CNNs) have proven to be really effective in a wide variety of computer vision tasks. Briefly, the key layers in a CNN model include convolution and pooling layers as depicted in the following figure.



A typical CNN architecture (Source: deeplearning.net)

Convolution layers learn spatial hierarchical patterns from the data, which are also translation invariant. Thus, they are able to learn different aspects of images. For example, the first convolution layer will learn small and local patterns such as edges and corners, a second convolution layer will learn larger patterns based on the features from the first layers, and so on. This allows CNNs to automate feature engineering and learn effective features which generalize well on new data points. Pooling layers help with down sampling and dimension reduction.

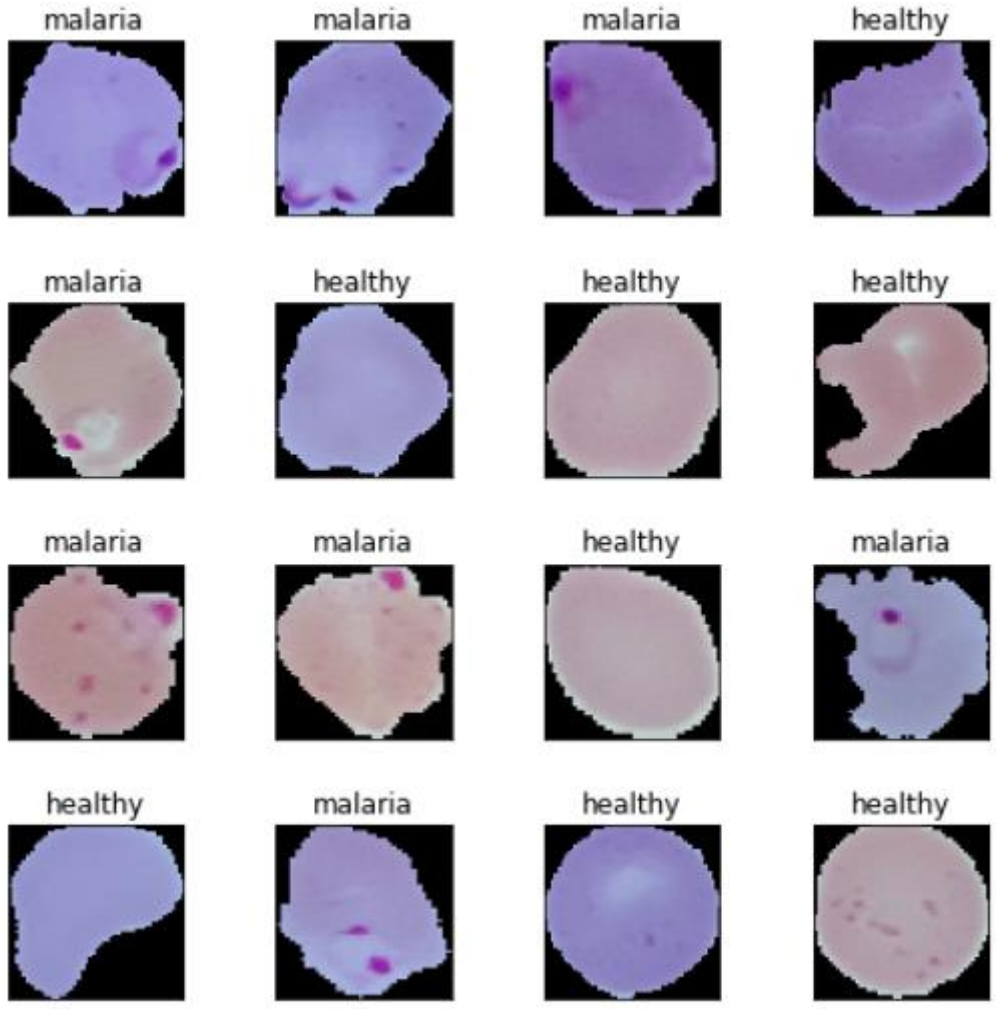
Thus, CNNs help us with automated and scalable feature engineering. Also, plugging in dense layers at the end of our model enables us to perform tasks like image classification. Automated malaria detection using deep learning models like CNNs could be very effective, cheap and scalable especially with the advent of transfer learning and pre-trained models which work quite well even with constraints like less data.

Dataset

The dataset contains 2 folders

- Infected
- Uninfected

And a total of 27,558 images.



Approach to the task

The images are divided into training and validation sets using TensorFlow ImageDataGenerator

• Resizing of images into a 3D tensor with shape of width = 68, height = 68, channels = 3 shaped arrays, so that every image will have the same dimensions as each other.

When performing image classification using CNN, it is recommended to rescale/normalize the images for less computational complexity. When using the image as it is and passing through a Deep Neural Network, the computation of high numeric values may become more complex. To reduce this, we can normalize the values to range from 0 to 1 by dividing it with 255 which is the highest pixel of an image.

Activation Functions Used

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

- Rectified Linear Unit (ReLU):

The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

- Sigmoid:

The sigmoid function is a special form of the logistic function and is usually denoted by σ(x) or sig(x). It is given by: $\sigma(x) = 1/(1+\exp(-x))$ When the activation function for a neuron is a sigmoid function, it is a guarantee that the output of this unit will always be between 0 and 1. Also, as the sigmoid is a non-linear function, the output of this unit would be a non-linear function of the weighted sum of inputs. Such a neuron that employs a sigmoid function as an activation function is termed as a sigmoid unit.

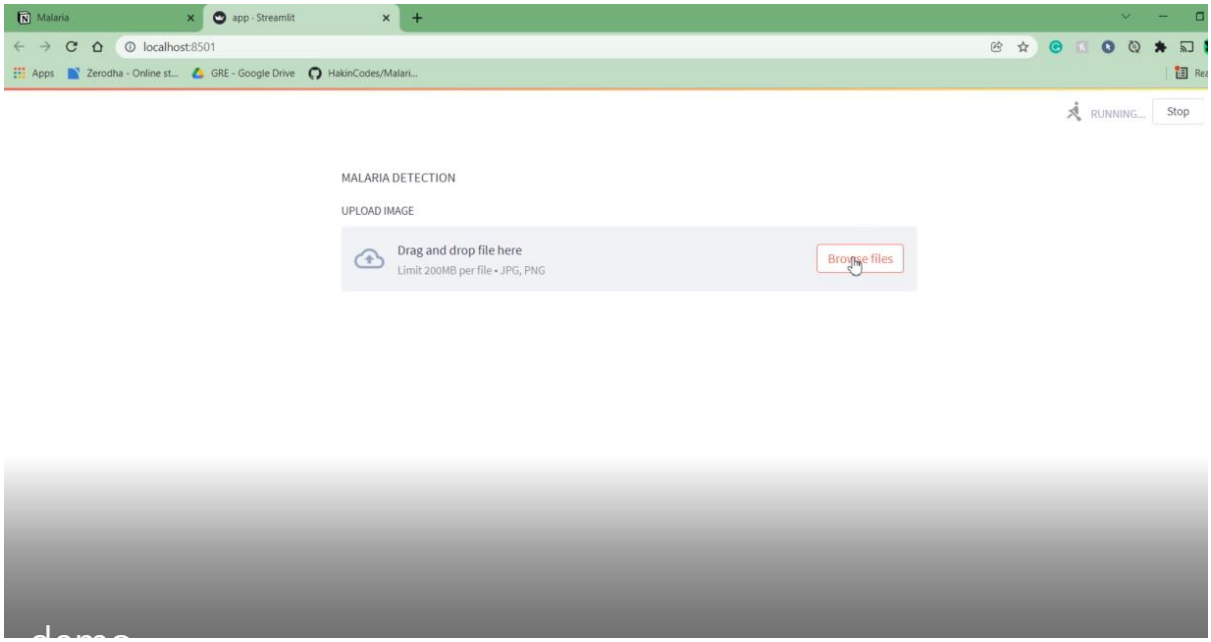
Results

```
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1378/1378 [==============================] - 134s 96ms/step - loss: 0.6303 - accuracy: 0.6429 - val_loss: 0.4596 - val_accuracy: 0.8494
Epoch 2/6
1378/1378 [==============================] - 122s 88ms/step - loss: 0.3297 - accuracy: 0.8975 - val_loss: 0.2384 - val_accuracy: 0.9287
Epoch 3/6
1378/1378 [==============================] - 127s 92ms/step - loss: 0.2186 - accuracy: 0.9301 - val_loss: 0.2152 - val_accuracy: 0.9343
Epoch 4/6
1378/1378 [==============================] - 112s 81ms/step - loss: 0.1885 - accuracy: 0.9348 - val_loss: 0.1778 - val_accuracy: 0.9356
Epoch 5/6
1378/1378 [==============================] - 112s 81ms/step - loss: 0.1798 - accuracy: 0.9371 - val_loss: 0.1751 - val_accuracy: 0.9374A:
52s - loss: 0.1766 - accuracy: 0.9335
Epoch 6/6
1378/1378 [==============================] - 110s 80ms/step - loss: 0.1611 - accuracy: 0.9470 - val_loss: 0.1758 - val_accuracy: 0.9410
```

Achieved 95% accuracy

Streamlit

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc. With Streamlit, no call-backs are needed since widgets are treated as variables. Data caching simplifies and speeds up computation pipelines. Streamlit watches for changes on updates of the linked Git repository and the application will be deployed automatically in the shared link.



REFERNCES:

[1] https://www.who.int/news-room/fact-sheets/detail/malaria

[2] https://www.cdc.gov/malaria/diagnosis_treatment/diagnosis.html

[3]https://towardsdatascience.com/detecting-malaria-with-deep-learning-9e45c1e34b60

[4]https://medium.com/datadriveninvestor/convolutional-neural-network-cnn-simplified-ecafd4ee52c5

[5]https://github.com/krishnakatyal/Malaria-Detection-with-Deep-Learning/blob/master/Deep_learning_for_Malaria_detection.ipynb

[6]https://github.com/sayakpaul/Malaria-Detection-with-Deep-