Malaria Detection

Context:

Malaria is a deadly disease that affects millions and causes many deaths every year. Traditional detection of malaria requires experts to test the blood cells under a microscope which can be very tedious and resource-intensive. Therefore, an automated system using machine learning can save time, costs, and ultimately the lives of many should an effective model be produced. A deep learning technique known as convolutional neural networks is a strong candidate to help us build this model given that the dataset we are working with involves images, for which CNNs are widely used to process and classify. Some key questions we need to ask for building the best model are how do we optimally pre-process the data to produce optimal performance for our model (blurring? HSV?), what sort of activation function we should use, and what techniques such as batch normalization should we use to improve performance assuming we are unsatisfied with our base model. With Data science, we will thus explore a variety of techniques to minimize error to make sure our model can accurately detect malaria in red blood cells.

The Data:

Our data set contains images of red blood cells where half of them are parasitized cells which contain the Plasmodium parasite, and the other half are uninfected cells which do not contain the Plasmodium parasite but may contain other impurities. The data is balanced, which is important for a convolutional neural network so that it can learn equally without bias towards the weight of a specific class. We also want to make sure that the image pixel values are normalized by dividing each image by 255 (the max pixel value of each image), which is the state that is

optimal for our deep learning model to learn on. Converting the images to HSV or applying Gaussian Blurring are also worth trying to see if we can establish better distinguishability between the classes.

Proposed Approach:

After making sure that our data is pre-processed and suitable for our convolutional neural network, we will start off with a base model and observe the performance. From there, we can begin adding complexity such as more layers, callbacks, batch normalization, and/or changing the activation function. After completing our models, we can also try running our data through a pre-trained model such as VGG16 and see how its performance compares to our models. From there, we can choose the best model for our given problem. As we go through these steps, we carefully observe the confusion matrix to assess the performance of these models, and continue with our hyperparameter tuning to obtain better results. The confusion matrix gives us various metrics such as accuracy, precision, recall, and f1-score which we will observe to gauge performance. We will also make sure that our models are generalizable, meaning that they perform on both the training data and the test data and not just the training data (overfitting). Making sure that our final model is generalizable is crucial because we want it to be able to perform on unseen data.