Malaria Detection Using Deep Learning
Jin Xing

1. Executive Summary - *What are the key takeaways? What are the key next steps?*

Malaria is a severe and often fatal disease. Conventional malaria detections are performed by microscopists who analyze microscopic blood smear images in laboratory settings, which requires human expertise and large investments. These resources may be inadequate in developing counties, where malaria is more predominant. Deep learning models thus may play a role in facilitating malaria detection and reducing healthcare costs. In this study, I build a Convolutional Neural Network (CNN) based neural network algorithm to classify images of blood cells into being parasitized or uninfected. The algorithm achieves high overall accuracy, about 98%. Other CNN models with similar structures also perform very well with similar performance. Thus, the model performance is robust to CNNs with varying configurations. As such, deep learning techniques show potential in achieving high accuracy even being applied to the healthcare setting fully automatically. Further tuning of the model may obtain even better performance. As smartphones are widely used, apps based on deep learning could be developed so that malaria detection can be widely conducted even with smartphones, increasing the cost-efficiency and the number of tests. However, the algorithm shows limitations as certain uninfected cells are wrongly classified while it fails to detect certain parasitized cells. This provides a caveat in applying the algorithm to actual healthcare settings and points to a direction for future improvement.

2. Problem and Solution Summary
2.1 Research Question - *What problem was being solved?*

Malaria is a severe and often fatal disease caused by parasites such as Plasmodium falciparum. Conventional malaria detections are conducted by microscopists who analyze microscopic blood smear images in laboratories. Their accuracy and efficiency mainly depend on the level and availability of human expertise. For these reasons, deep learning models can aid microscopists in making decisions or even conducting the analysis automatically. They could significantly streamline the detection process and reduce health care costs if successful.

This project aims to build and evaluate Deep Learning algorithms based on the Convolutional Neural Network (CNN) to classify malaria-infected cells from non-infected ones. To this end, I use a dataset from the US National Institutes of Health (NIH), which contains 27,558 different malaria-infected or non-infected blood cell images from 150 malaria-infected patients and 50 healthy patients.


2.2 Methodology - *What are the key points that describe the final proposed solution design?*

2.2.1 Data Processing
Each image from the NIH has a label associated with it, either parasitized or uninfected, and contains three color layers, red, green, and blue (RGB). I resized each image to 64×64. There are 24,958 training images for learning the parameters of the neural network. Out of all the training images, 12,582 are parasitized, and 12,376 are uninfected. Also, there are 2,600 testing images to evaluate the neural network built. Out of all the testing images, 1,300 are parasitized, and 1,300 are uninfected. Thus, the numbers of parasitized or uninfected blood cell images are balanced for training and testing data.
Figure 1 shows sample images of parasitized and uninfected blood cells. We observe that parasitized blood cells are typically stained in dark purple or pink spots, while uninfected blood cells typically have no such spots and are uniform.

Figure 1: Sample images of parasitized and uninfected blood cells
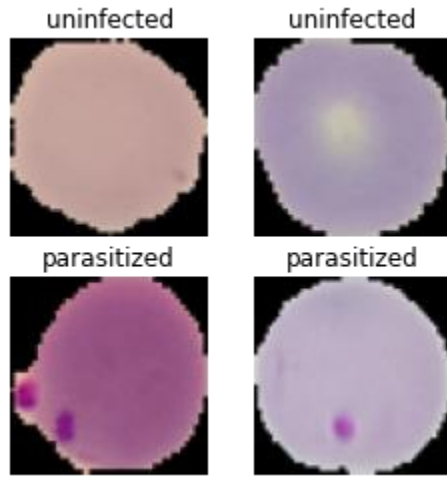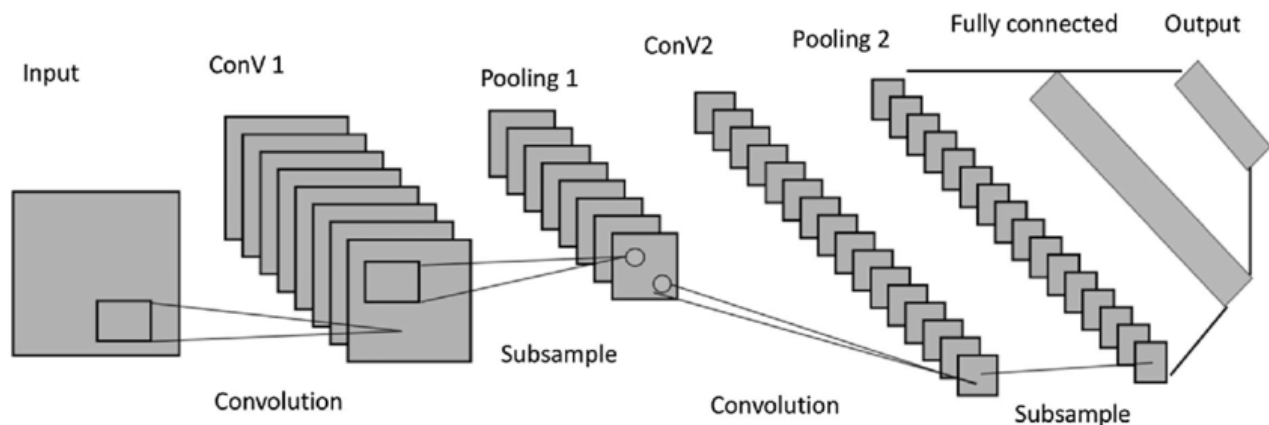

2.2.2 Building CNN-Based Neural Network

CNN classifies images by processing the image through several layers. Three types of layers contained in a typical CNN are listed and described below:

1. Convolutional layer. This first layer applies a kernel to scan the input image, creating an output layer.
2. Pooling layer. This intermediate layer applies operations (e.g., max pooling, dropout, and batch normalization) to alleviate the overfitting on the training data.
3. Fully connected layer. This last layer first flattens the processed image into a vector and then runs it through a fully connected neural network for classification.

Note that the convolutional layer and the pooling layer can be repeated before moving onto the final layer. Figure 2 illustrates a typical architecture for CNN.

Figure 2: Architecture for a typical CNN (Source: ResearchGate)

In this study, I built a CNN-based neural network with a kernel size of 2×2, a dropout rate of 40%, ReLU as the activation function, and softmax as the activation function for the final classification. Figure 3 shows the architecture of my neural network.

Figure 3: Architecture for the CNN used in this study

Layer	Output Shape	Parameter #
Conv2D	(None, 64, 64, 32)	416
Max Pooling	(None, 32, 32, 32)	0
Dropout	(None, 32, 32, 32)	0
Conv2D	(None, 32, 32, 32)	4128
Max Pooling	(None, 16, 16, 32)	0
Dropout	(None, 16, 16, 32)	0
Conv2D	(None, 16, 16, 32)	4128
Max Pooling	(None, 8, 8, 32)	0
Dropout	(None, 8, 8, 32)	0
Conv2D	(None, 8, 8, 32)	4128
Max Pooling	(None, 4, 4, 32)	0
Dropout	(None, 4, 4, 32)	0
Flatten	(None, 512)	0
Dense	(None, 512)	262656
Dropout	(None, 512)	0
Dense	(None, 512)	262656
Dropout	(None, 512)	0
Dense	(None, 2)	1026

Note: The number of total parameters is 539,138, the number of trainable parameters is 539,138, and the number of non-trainable parameters is 0.

2.3 Evaluation - *Why is this a 'valid' solution that is likely to solve the problem?*

Figure 4 shows the evaluation graph, in which the validation split is 0.2, i.e., 20 percent of the training data are kept aside to test the model performance during the model building phase. The model performance is measured by the overall accuracy. The training and validation accuracy become at the same level as the number of iterations (epochs) approaches 4.
The model achieves an overall accuracy of 98% for the testing data. Other models with similar structures, such as those with additional layers, those with batch normalization and LeakyReLU as the activation function, those with augmented images, or pre-trained models (e.g., VGG16), all achieve over 97% accuracy. Therefore, the model performance is robust with different similar structures of CNN.
Figure 5 shows the confusion matrix.
Figure 6 shows error samples, which are image samples that the CNN fails to classify correctly. Certain uninfected cells (labeled 0) have purple spots and therefore are wrongly classified as being parasitized by the CNN. On the other hand, certain parasitized cells (labeled 1) have purple spots, but the CNN fails to detect them.
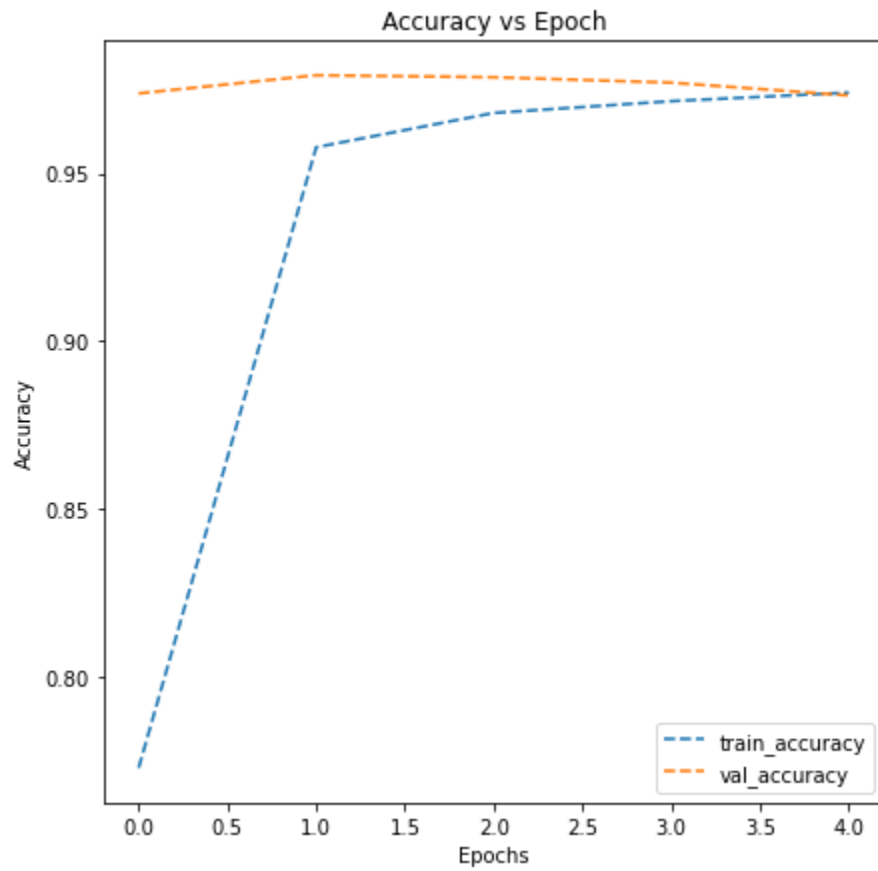
Figure 4: Evaluation graph

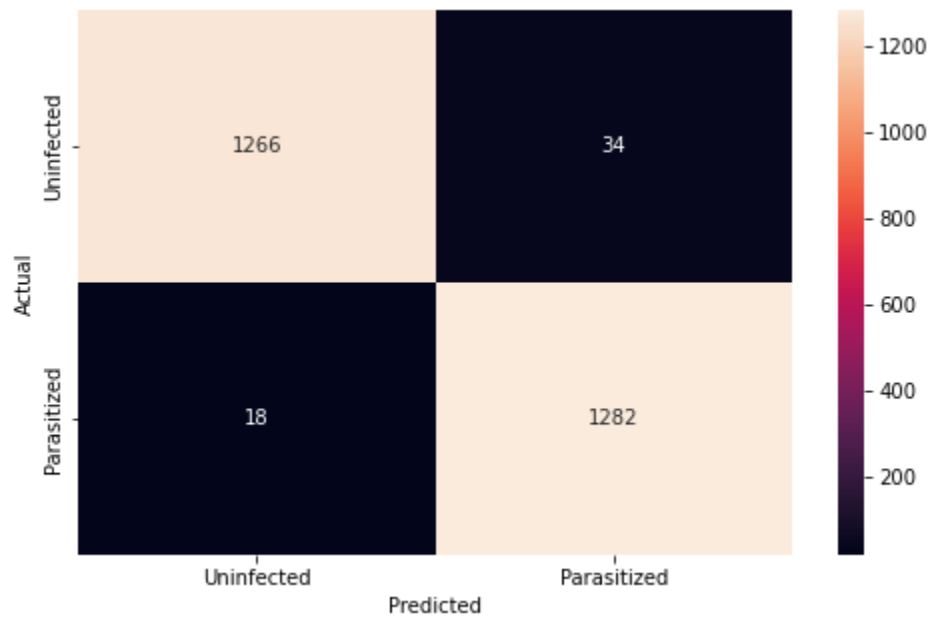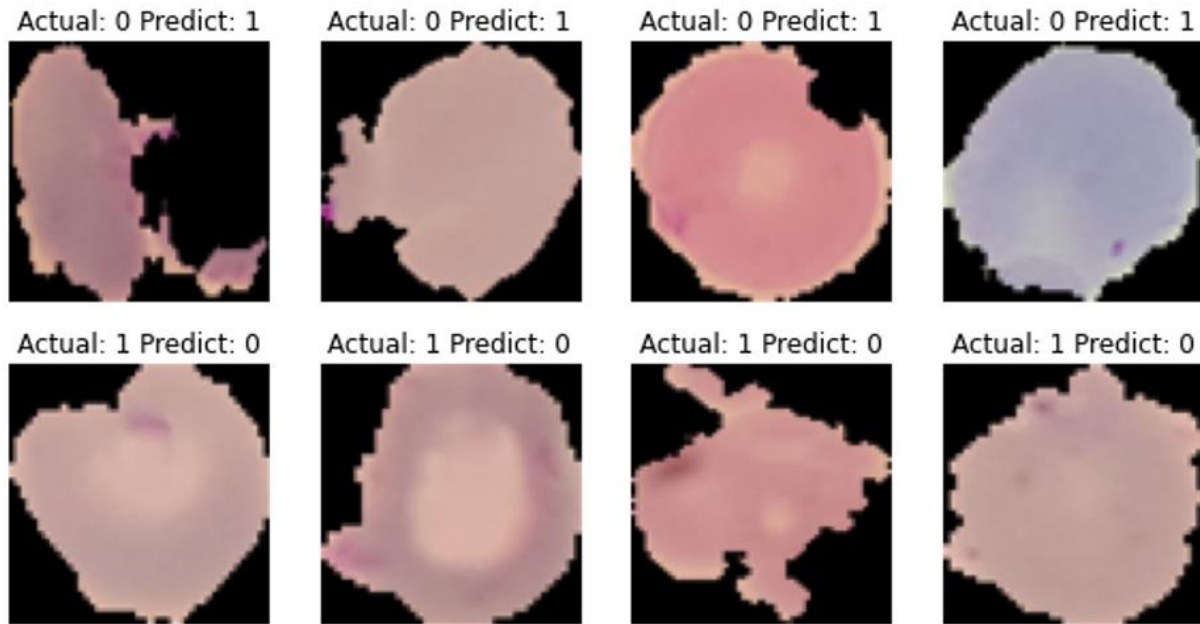
Figure 5: The confusion matrix

Figure 6: Error samples

Note: Uninfected cells are labeled 0. Parasitized cells are labeled 1.

3. Recommendations for Implementation - *What are some key recommendations to implement the solutions? What are the key actionables for stakeholders? What is the expected benefit and/or costs? What are the key risks and challenges? What further analysis needs to be done or what other associated problems need to be solved?*

Figure 6 shows the limitations of the algorithm built as certain uninfected cells are wrongly classified while it fails to detect certain parasitized cells. This points to a direction for future improvement. The model performance may be further improved by:

1. Adjusting the network structure (e.g., the number of layers, the number of nodes in each layer, activation functions, etc.)
2. Tuning hyperparameters (e.g., initial parameters, step size, regularization, stopping, the shape of a local detector, etc.)
3. Using alternative pre-trained models (e.g., VGG-19).
4. Adopting supervised learning (e.g., KNN, random forest, logistic regression) or unsupervised learning (e.g., PCA) for classification instead of a fully connected neural network.