Analysis of Image Processing Techniques in Malaria Cell Classification using Convolutional Neural Networks

Rayhaan Rasheed[1], William Noone[1] and Samuel Aboagye[1]

[1]Department of Data Science, The George Washington University, Washington, D.C., USA

Abstract

Malaria is a mosquito-borne disease caused by a parasite. People with malaria often experience fever, chills, and flu-like illness. Left untreated, they may develop severe complications and die. In 2016 an estimated 216 million cases of malaria occurred worldwide and 445,000 people died, mostly children in the African Region. Malaria can only be diagnosed by examining a stained blood film of the patient with a microscope. Three different Convolutional Neural Networks(CNN) were trained in classifying cells that are parasitic and non parasitic. The models used various configurations of Convolution, Pooling, and ReLU layers along with two fully connected layers and a SoftMax at the end. Based on the average test accuracy across all three models, unprocessed images performed best with 91.7% with training loss decay oscillating the tighter bounds. Further analysis allowed the average test accuracy to eventually become 93.7%.

Keywords: Malaria, Convolutional Neural Network, Classification, Image Processing, Caffe

1. Introduction

Malaria is a deadly blood-based disease that is contracted by parasites. These parasites are commonly transferred with the help of the female *Anopheles* mosquito. Due to the environment with which these mosquitoes mainly reside in, it is easy to transfer blood from an infected patient to a normal subject. The parasite is mixed with the mosquito's saliva which then comes in contact with the victim. Malaria can only be diagnosed by examining a stained blood film of the patient with a microscope. It can take days to diagnose the disease and often requires examining about 100 different areas of the cell has to be examined by a highly skilled lab technologist and it could take days to confirm a parasitic cell. [1]

2. Background

Many of the issues in classifying samples comes from the variability in the size of the sample, sample orientation, progression of the parasite, and amount of background noise. Traditional methods would heavily process images using a variety of algorithms. The resulting images would then be classified by a physician or an expert in that domain, which hindered the accuracy. Applying this method on large sets of images would take unnecessary amounts of time and resources. As more computational power and better image analysis tools became easily accessible to the public, the number of methods utilizing Machine Learning (ML) and Deep Learning (DL) algorithms increased. Such methods presented a way for individuals analyze very large sets of images in a fraction of the time and with better accuracy. [2]

1.1 Machine Learning Approaches

In previous publications, groups used a combination of image processing and ML-based segmentation algorithms for feature extraction and selection. The ring-like structure of the stain in parasitized images can be used as a main feature of the image. Some experiments in the past have looked a multiple blood cells before they were segmented from the blood smear. Techniques such as watershed segmentation or K-Means clustering were used to identify parasitized and normal cells.[3] Finally, classifiers, primarily Support Vector Machines (SVM), were used to classify the images as either "Parasitized' or "Normal". One example by Sampathila et. al used an RGB segmentation and an HSV color conversion to segment the parasite and extract features before using a neural network for classification of the type or malarial parasite. [4] Majority of the publications do not change the original RGB color space. Since using other color spaces has not been fully explored, there is a perfectly good reason to do so. [5]

1.2 Convolutional Neural Networks

Feature extractors and segmentation algorithms that are manually build may not be able to capture all the vital information needed to perform an accurate classification. Deep Learning methods provide a way to discover features and extract vital information by creating a set of layers that utilize non-linear functions. Currently, the most popular DL technique for analyzing images is the Convolutional Neural Network (CNN). The idea is that the image is sent through cascading layers consisting of convolution and pooling based on filters and initialized kernels. [6] Previous groups have classified malarial cells using pre-trained CNNs such as AlexNet, VGG16, and GoogleLeNet. These models have been trained on ImageNet or CIFAR-100 data before being used on the malaria dataset. [7] Krishnan et al. compared the pre-trained models to their own custom model that consisted of 13 layers using 3 convolution layers, 3 max pooling layers, 3 ReLu layers, 2 fully-connected layers, and softmax at the end. Their model resulted in a slightly better accuracy of 98%, but it took the longest time to analyze all the images. [8] Similarly, Liang et al. developed their own model with more convolution and fully connected layers as well as an added sigmoid layer before the softmax classifier. Even with all the additions, this model only produced an accuracy of 97.37%. [9]

3. Method

3.1 Data

The images used in this study were pulled from the U.S. National LIbrary of Medicine's Communications Engineering Branch. There are a total of 27,558 images of single red blood cells from Giemsa stained thin blood slides. These samples were taken from patients at the Chittagong Medical College hospital in Bangladesh where patients were either normal or infected with the *P. falciparum* parasite. Each image is roughly the same size and is given one of two classes: "Uninfected" or "Parasitized".

3.2 Image Processing

The original images come in an RGB color space of various heights and widths. First off the images were resized to 100x100 so that all input images would be the same size. Next, the images were converted from RGB to Grayscale. This reduced the number of channels that needed to be taken into account. Noise was reduced by applying a Gaussian filter consisting of a standard deviation of one. In order to take advantage of the unique intensity of the malaria stains, a threshold mask is used to allow to isolate the stain and the edge of the blood cell (See Figure 1).

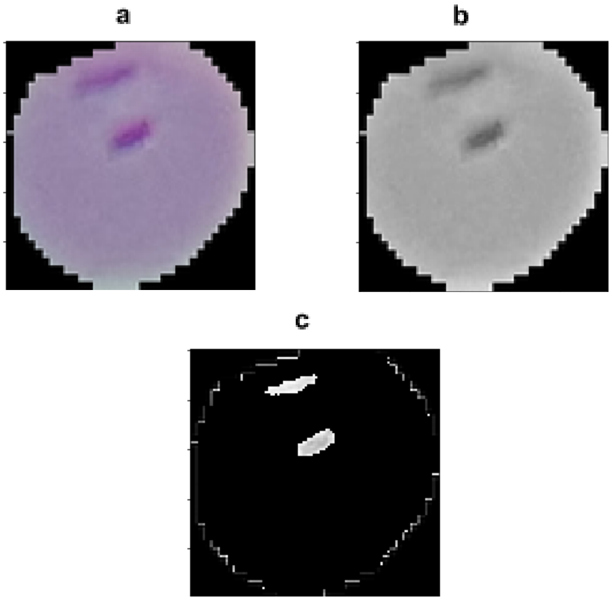


Figure 1: (a) Original Image (b) Grayscale Image (c) Fully Processed Image

3.3 Network Architecture

The team used a Convolution Neural Network built in Caffe, to analyze variations of the same cell images. After copious research, the authors could not find another study to which the Caffe framework was deployed upon this NIH dataset. Within these (2) image sets, the data was split (70/30) into training and validation sets. Training the CNN involved creating a multi-layer feedforward network, whereby convolution and pooling operations extracted sets of features maps from respective layers. Following the forward propagation, the loss function (Softmax) was employed to gauge the performance of the binary class label prediction. Backpropagation then updated weights and bias of by calculating the gradient & sensitivities w.r.t. the parameters. This operation series was performed over 20,000 iterations with accuracy typically hovering in the low 90% range.

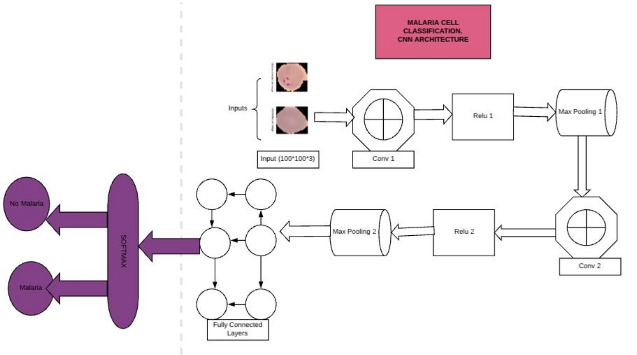


Figure 2: 2-Layer CNN

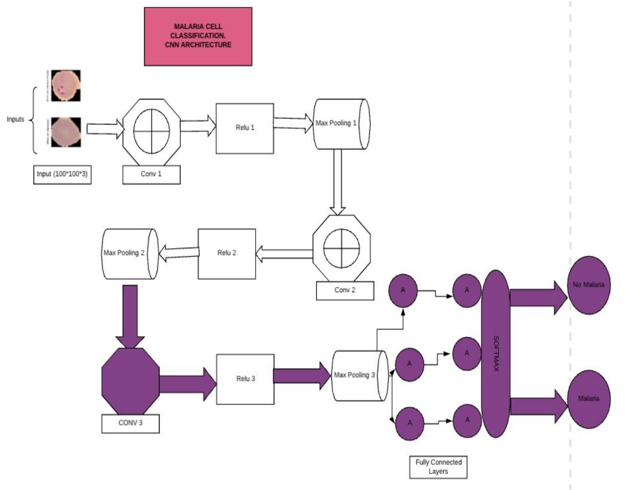


Figure 3: 3-Layer CNN

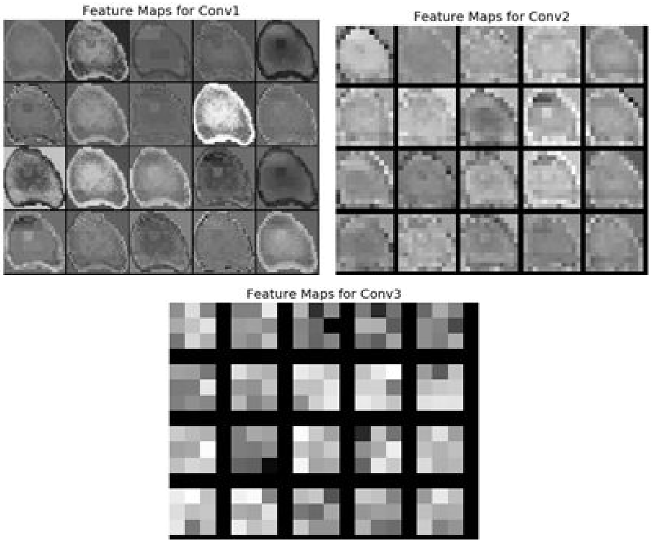
3.4 Model Evaluation

The study aimed to gauge the impact of image processing upon the ability of a CNN to accurately predict binary classifiers. The approach was to first analyze both underlying image sets (unprocessed, processed) within identical CNN models (2 layer & 3 layer all else equal), then take a high performing image/model combination and fine-tune the hyper-parameters. The team opted to measure performance by plotting training loss and test accuracy (w.r.t validation subset), logging mean and standard deviation of both across a 20,000 iteration training/testing phase.

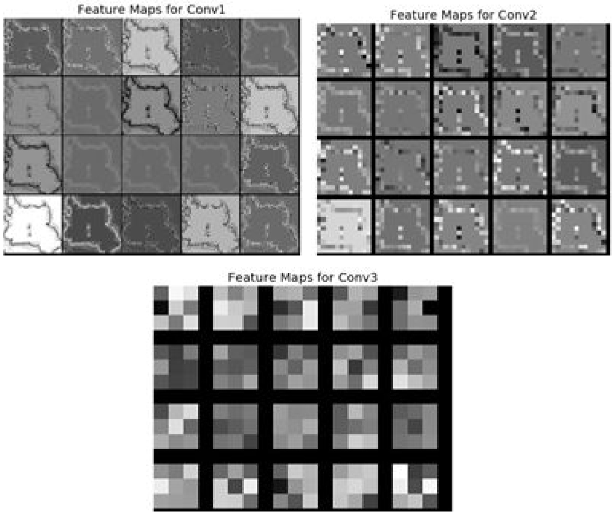
3.5 Feature Maps

An optical scan of the feature maps produced by a simple 3 layer CNN provides insight to the method & acceleration of collapse of principle features across layers. The team hypothesis leaned in favor of processed images yielding superior CNN performance due to image noise reduction via filtering and threshold masking.

Feature Maps - Unprocessed Images



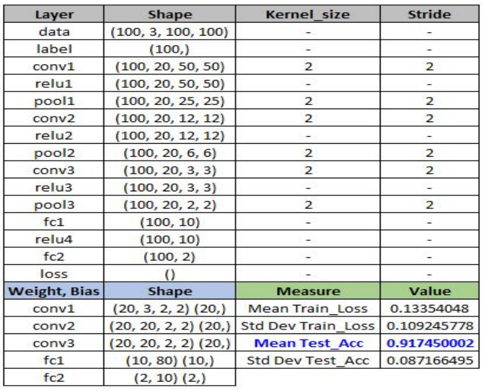
Feature Maps - Processed Images



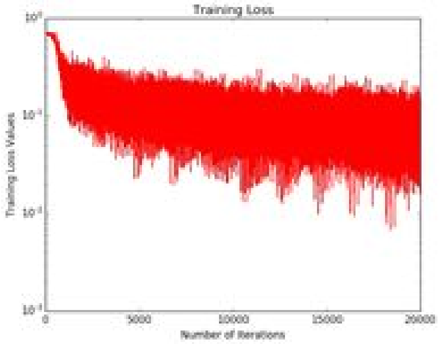
4. Results

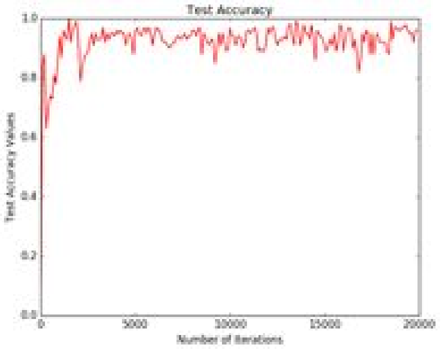
The first model (Model1_Unprocessed) was a 3 layer CNN (20 FMs per layer, kernel_size = 2, stride =2 across all layers) with (2) fully connected layers and a soft-max loss function. The corresponding model (Model1_Processed) was a 3 layer CNN with kernel_size = 2, stride =2 across all layers with (2) fully connected layers and a soft-max loss function. Using mean test_accuracy as a benchmark, it was found that unprocessed images performed best with 91.7% with training loss decay oscillating the tighter bounds (as measure by std dev).

Model 1 - Unprocessed Images

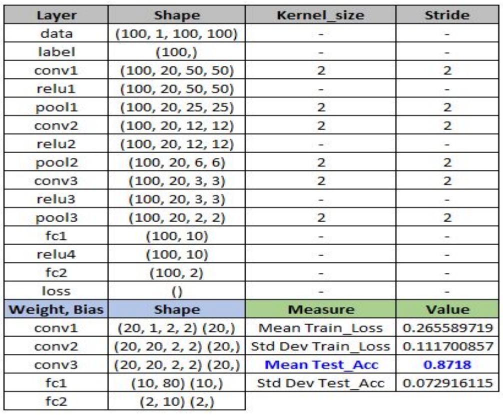


Layer	Shape	Kernel_size	Stride
data	(100, 3, 100, 100)	-	-
label	(100,)	-	-
conv1	(100, 20, 50, 50)	2	2
relu1	(100, 20, 50, 50)	-	-
pool1	(100, 20, 25, 25)	2	2
conv2	(100, 20, 12, 12)	2	2
relu2	(100, 20, 12, 12)	-	-
pool2	(100, 20, 6, 6)	2	2
conv3	(100, 20, 3, 3)	2	2
relu3	(100, 20, 3, 3)	-	-
pool3	(100, 20, 2, 2)	2	2
fc1	(100, 10)	-	-
relu4	(100, 10)	-	-
fc2	(100, 2)	-	-
loss	()	-	-
Weight, Bias	**Shape**	**Measure**	**Value**
conv1	(20, 3, 2, 2) (20,)	Mean Train_Loss	0.13354048
conv2	(20, 20, 2, 2) (20,)	Std Dev Train_Loss	0.109245778
conv3	(20, 20, 2, 2) (20,)	**Mean Test_Acc**	**0.917450002**
fc1	(10, 80) (10,)	Std Dev Test_Acc	0.087166495
fc2	(2, 10) (2,)		

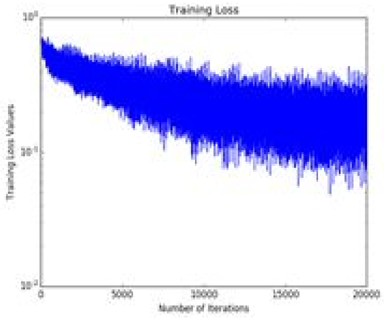


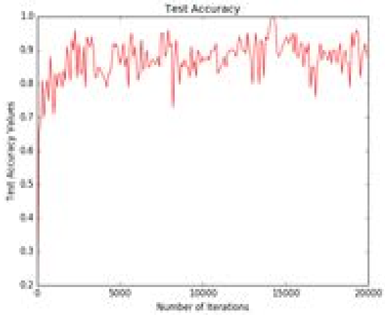


Model 1 - Processed Images



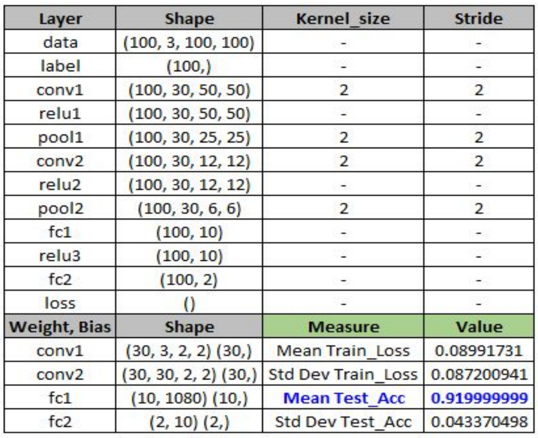
Layer	Shape	Kernel_size	Stride
data	(100, 1, 100, 100)	-	-
label	(100,)	-	-
conv1	(100, 20, 50, 50)	2	2
relu1	(100, 20, 50, 50)	-	-
pool1	(100, 20, 25, 25)	2	2
conv2	(100, 20, 12, 12)	2	2
relu2	(100, 20, 12, 12)	-	-
pool2	(100, 20, 6, 6)	2	2
conv3	(100, 20, 3, 3)	2	2
relu3	(100, 20, 3, 3)	-	-
pool3	(100, 20, 2, 2)	2	2
fc1	(100, 10)	-	-
relu4	(100, 10)	-	-
fc2	(100, 2)	-	-
loss	()	-	-
Weight, Bias	**Shape**	**Measure**	**Value**
conv1	(20, 1, 2, 2) (20,)	Mean Train_Loss	0.265589719
conv2	(20, 20, 2, 2) (20,)	Std Dev Train_Loss	0.111700857
conv3	(20, 20, 2, 2) (20,)	**Mean Test_Acc**	**0.8718**
fc1	(10, 80) (10,)	Std Dev Test_Acc	0.072916115
fc2	(2, 10) (2,)		



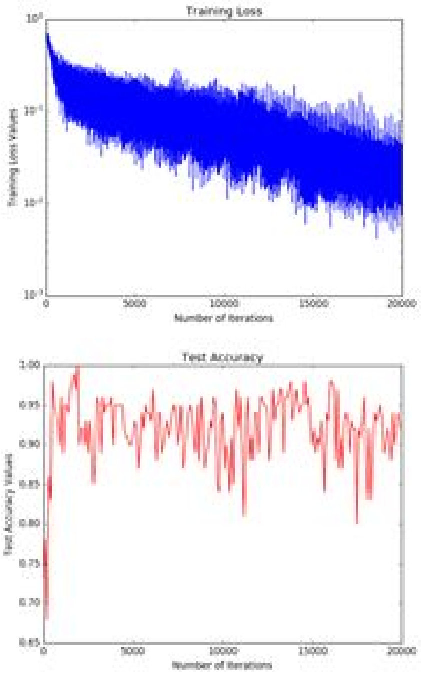


The next model (Model2_Unprocessed) was a 2 layer CNN (30 FMs per layer, kernel_size = 2, stride =2 across all layers with (2) fully connected layers and a soft-max loss function. The corresponding model (Model2_Processed) was a 3 layer CNN with kernel_size = 2, stride =2 across all layers with (2) fully connected layers and a soft-max loss function. Using mean test_accuracy as a benchmark, one can see that unprocessed images again performed best with 91.9% with training loss decay oscillating the tighter bounds (as measure by std dev).

Model 2 - Unprocessed Images

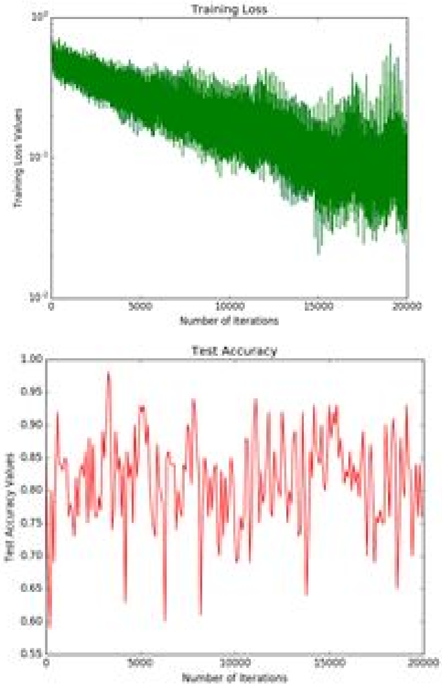


Layer	Shape	Kernel_size	Stride
data	(100, 3, 100, 100)	-	-
label	(100,)	-	-
conv1	(100, 30, 50, 50)	2	2
relu1	(100, 30, 50, 50)	-	-
pool1	(100, 30, 25, 25)	2	2
conv2	(100, 30, 12, 12)	2	2
relu2	(100, 30, 12, 12)	-	-
pool2	(100, 30, 6, 6)	2	2
fc1	(100, 10)	-	-
relu3	(100, 10)	-	-
fc2	(100, 2)	-	-
loss	()	-	-
Weight, Bias	**Shape**	**Measure**	**Value**
conv1	(30, 3, 2, 2) (30,)	Mean Train_Loss	0.08991731
conv2	(30, 30, 2, 2) (30,)	Std Dev Train_Loss	0.087200941
fc1	(10, 1080) (10,)	**Mean Test_Acc**	**0.919999999**
fc2	(2, 10) (2,)	Std Dev Test_Acc	0.043370498



Model 2 - Processed Images

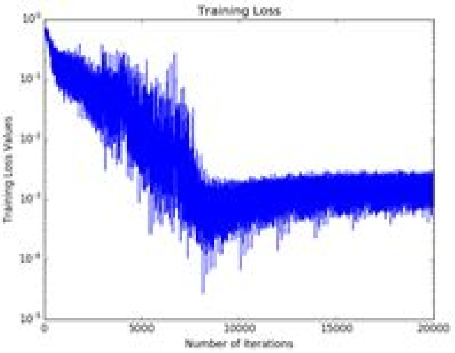
Layer	Shape	Kernel_size	Stride
data	(100, 1, 100, 100)	-	-
label	(100,)	-	-
conv1	(100, 30, 50, 50)	2	2
relu1	(100, 30, 50, 50)	-	-
pool1	(100, 30, 25, 25)	2	2
conv2	(100, 30, 12, 12)	2	2
relu2	(100, 30, 12, 12)	-	-
pool2	(100, 30, 6, 6)	2	2
fc1	(100, 10)	-	-
relu3	(100, 10)	-	-
fc2	(100, 2)	-	-
loss	()	-	-
Weight, Bias	**Shape**	**Measure**	**Value**
conv1	(30, 1, 2, 2) (30,)	Mean Train_Loss	0.213966045
conv2	(30, 30, 2, 2) (30,)	Std Dev Train_Loss	0.124579427
fc1	(10, 1080) (10,)	**Mean Test_Acc**	**0.814999998**
fc2	(2, 10) (2,)	Std Dev Test_Acc	0.07026379

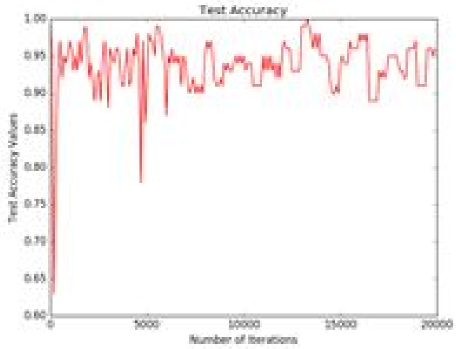


After reviewing Model 1 & 2, the team opted to optimize a 3 layer CNN (Model3_Unprocessed) with kernel_size = 4, stride =2 across all layers, (2) fully connected layers, a soft-max loss function. The material change to Model 3 involved “deepening” layers: Layer 1 = 32 FM, Layer 2 = 32 FM, Layer 3 = 64 FM. After researching multiple configurations, the team was able to increase mean test_accuracy to 93.7%.

Model 3 - Unprocessed Images

Layer	Shape	Kernel_size	Stride
data	(100, 3, 100, 100)	-	-
label	(100,)	-	-
conv1	(100, 32, 49, 49)	4	2
relu1	(100, 32, 49, 49)	-	-
pool1	(100, 32, 25, 25)	2	2
conv2	(100, 32, 11, 11)	4	2
relu2	(100, 32, 11, 11)	-	-
pool2	(100, 32, 6, 6)	2	2
conv3	(100, 64, 2, 2)	4	2
relu3	(100, 64, 2, 2)	-	-
pool3	(100, 64, 1, 1)	2	2
fc1	(100, 64)	-	-
relu4	(100, 64)	-	-
fc2	(100, 2)	-	-
loss	()	-	-
Weight, Bias	**Shape**	**Measure**	**Value**
conv1	(32, 3, 4, 4) (32,)	Mean Train_Loss	0.031756144
conv2	(32, 32, 4, 4) (32,)	Std Dev Train_Loss	0.081596197
conv3	(64, 32, 4, 4) (64,)	**Mean Test_Acc**	**0.937650002**
fc1	(64, 64) (64,)	Std Dev Test_Acc	0.036850747
fc2	(2, 64) (2,)		





5. Conclusion

Surprisingly, image processing techniques had a negative effect upon CNN predictive performance. This could be improved upon by modifying the parameters of the image filtering and masking steps. It is a time intensive process to convert image sets from JPG to LMDB format suitable for the Caffe framework. Additional challenges were the prohibitive with respect to time and available resources.This study underscores the importance of image processing on image classification tasks. The team recognizes that the study is not exhaustive as countless specifications (dropout, learning rate, optimizers, etc) could be altered in the pursuit of performance tuning. More specialized image analysis approaches are needed to take into account the specific patterns and color spaces in these blood cell images. Proper classification of malarial cells is vital for classification of specific strains. Using the proposed classification technique as an a priori for identifying various strains of malaria, unique treatment can be synthesized to help save lives.

References

[1] Center for disease Control and Prevention (CDC). https://www.cdc.gov/malaria/diagnosis_treatment/clinicians1.html

[2] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15 (June 2014), 1929–1958.

[3] Nasir AA, Mashor M, Mohamed Z. Segmentation based approach for detection of malaria parasites using moving k-means clustering. In: EMBS conference on biomedical engineering and sciences (IECBES). IEEE; 2012:653–8.

[4] Poostchi, M., Silamut, K., Maude, R. J., Jaeger, S., Thoma, G. Image analysis and machine learning for detecting malaria. Journal of Translational Research 2017:194-33

[5] Sampathila, N., Shet, N., Basu, A. Computational approach for diagnosis of malaria through classification of malaria parasite from microscopic image of blood smear. Journal of Biomedical Research 2018: 29-970

[6] LeCun Y, Bengio Y, Hinton G. 2015. Deep learning. Nature 521:436–444

[7] Rajaraman et al. (2018), Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ 6:e4568; DOI 10.7717/peerj.4568

[8] Krishnan S., Antani, S., Jaeger, S., Visualizing Deep Learning Activations for Improved Malaria Cell Classification.

[9] Liang et al. (2016), CNN-Based Image Analysis for Malaria Diagnosis. In: EMBS conference on biomedical engineering and sciences (IECBES). IEEE