CONVOLUTIONAL NEURAL NETWORKS FOR MALARIA DETECTION

A PREPRINT

Zahid Amadxarif
zamadxarif@gmail.com

June 2, 2019

ABSTRACT

This paper outlines a convolutional neural network which uses patient cell image data to solve a binary classification problem of whether a cell is infected with malaria or not. It outlines one customised architecture and three architectures which use pre-trained networks including ResNet-50, NasNetMobile and InceptionV3. The results show that the customised architecture obtains significantly better results than those obtained by the pre-trained networks. Compared to previous literature, it obtains an improved sensitivity score.

***K*eywords** Malaria detection · Convolutional Neural Networks · Image classification

1 Introduction

1.1 Overview

Malaria is a disease that is caused by parasites and is transmitted through bites of mosquitoes. In 2017, there were an estimated 219 million cases of malaria in the world according to the World Malaria Report 2018. Within Africa, Malaria claims lives of more than 435 000 people each year and every two minutes, it is estimated that a child dies from this preventable disease. Furthermore, reductions in malaria cases have stalled after several years of decline. It is difficult to diagnose malaria as it requires experienced lab technicians to analyse skin cells under a microscope. In countries with fewer experienced technicians, patients face a higher risk of being misdiagnosed as not having malaria. With the advent of machine learning methods, computer image recognition could be used to aid technicians to detect malaria.

Rajaraman et al (2018) introduce a dataset of malaria infected and uninfected cells. The data for this was collected via an app created by researchers at the Lister Hill National Centre for Biomedica Communications (LHNCBC) (Poostchi et al., 2018). The images were then manually annotated by an expert slide reader at the Mahidoi-Oxford Tropical Medicine Research Unit in Bangkok, Thailand. Rajaraman et al (2018) then used this dataset to train a customised Convolutional Neural Network (CNN) as well as utilising Transfer Learning with pre-trained CNNs. In this paper, I utilised the same dataset to train a customised CNN to detect malaria infected cells. The performance of this CNN is benchmarked against the CNN of Rajaraman et al (2018) and pre-trained CNNs that utilised the ImageNet database.

2 Data

2.1 Exploration and Preprocessing

The dataset includes 27,558 images where 13,779 of images are of infected cells and 13,779 number of images are of uninfected images. The images were split into a 60:20:20 ratio of training (16534 images), validation (5512 images) and testing (5512 images) data. Two images were removed from the training set as the folder was incorrectly importing a system file as an image file. Figure 1 illustrates the types of images in the dataset. As you can see, whilst uninfected cells seem to have different shapes and similar shades of colour, infected cells vary in both shape and colour which could be due to the varying degree of infection. Figure 2 shows intensity histograms for three images of different

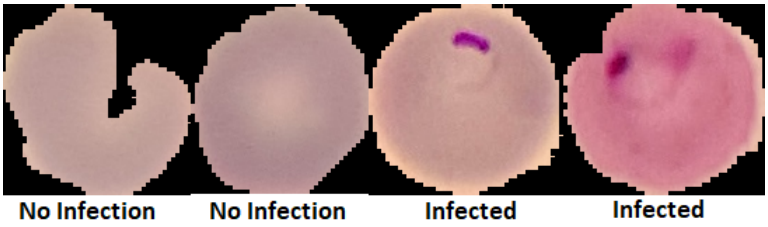

Figure 1: Cell Images

categories: a mildly infected cell, a severly infected cell and an uninfected cell. As you can see, a severely infected cell seems to have a higher contrast than both the mildly infected cell and the uninfected cell. All three images have strong brightness and a high number of pixels. There is also no sign of saturation effects for all three images which means that the pixels remain in their range. These histograms demonstrate that there are no observable reasons to carry out any alterations to the pixels of any images.

Table 1 shows some simple descriptive statistics of the dataset to illustrate the varied size of images in the dataset. The range is somewhat wide but most images are in the 130x148 size. For preprocessing purposes, the images were transformed into 3-dimensional objects of 224x224x3. These were then transformed into 4-dimensional tensors (1x224x224x3) that can be used as an input for TensorFlow based packages in Python. These were then transformed to be ready for use on pre-trained CNNs.

Table 1: Descriptive Statistics for Image Size

Statistic	**Image Size**
Largest	394x241
Smallest	46x79
Median	130x130
Mode	130x148

2.2 Metrics and Benchmark

The metrics I use to evaluate the performance of the CNNs are Accuracy, Sensitivity and Specificity. These can be defined in Table 1 as specified in Baratloo et al (2015).

Table 2: Evaluation Metrics

Measure	**Explanation**	**Formula**
Accuracy	The accuracy of a test is its ability to differentiate the patient and healthy cases correctly. To estimate the accuracy of a test, we should calculate the proportion of true positive and true negative in all evaluated cases.	$\frac{TP+TN}{TP+TN+FP+FN}$ (1)
Sensitivity	The sensitivity of a test is its ability to determine the patient cases correctly. To estimate it, we should calculate the proportion of true positive in patient cases.	$\frac{TP}{TP+FN}$ (2)
Specificity	The specificity of a test is its ability to determine the healthy cases correctly. To estimate it, we should calculate the proportion of true negative in healthy cases.	$\frac{TN}{TN+FP}$ (3)

In Table 1, 'Patient' is defined as someone positive for disease. 'Healthy' is defined as someone negative for disease. True positive (TP) refers to the number of cases correctly identified as patient. False positive (FP) refers to the number of cases incorrectly identified as patient. True negative (TN) refers to the number of cases correctly identified as healthy. False negative (FN) refers to the number of cases incorrectly identified as healthy.

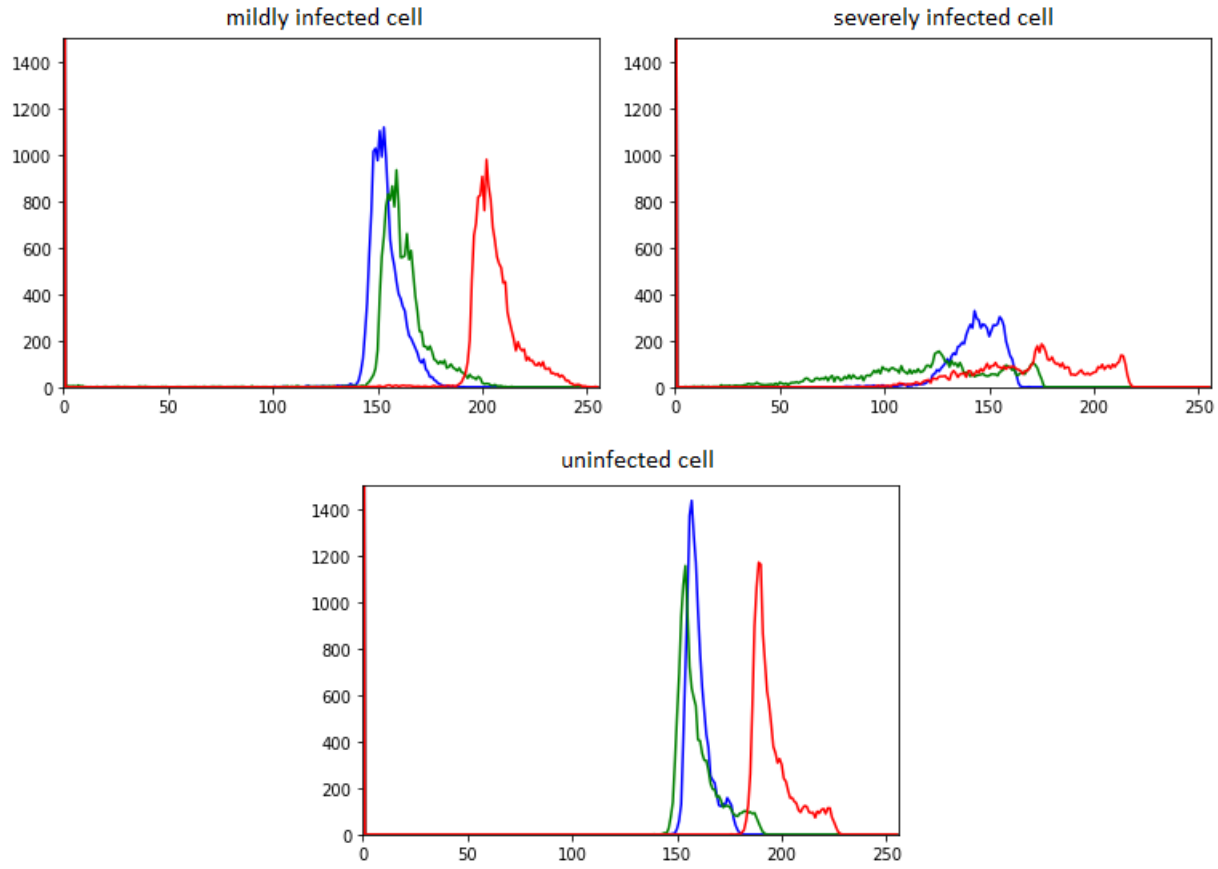

Figure 2: Intensity histograms of cell images

Rajaraman et al (2018) utilised a customised CNN with three convolutional layers and two fully connected layers. They also carried out transfer learning using pre-trained models such as AlexNet, VGG-16, ResNet-50, Xception, DenseNet-121. In this paper, I utilise their results as a benchmark for the performance of the CNN.

3 CNN Architecture

3.1 Customised CNN

Figure 3 illustrates the architecture of the customised CNN. It contains three convolutional layers to extract the features from the images and three maximum pooling layers to help reduce dimensionality. Rajaraman et al (2018) also use three convolutional layers for their customised CNN. Limiting the number of convolutional layers to three is justified as the inclusion of additional layers could lead to overfitting. The architecture includes a dropout layer to also reduce overfitting and a flatten layer to transform the data into a vector. The convolutional layers utilise a relu activation function to remove negative elements and retain non-negative elements. This architecture leads to 98,106 trainable parameters to classify two classes: infected cells and non-infected cells.

The CNN is trained using backpropagation. All layers of the network are fine-tuned with the global learning rate of 0.001 using the RMSProp optimizer. This provides a type of adaptive learning rate method which helps us optimise quicker than through using a learning rate schedule. This CNN uses a categorical cross-entropy loss function to minimise the distance between the predicted and actual probability distributions. The CNN was run for 20 epochs with a batch size of 20. The model is cross-validated and I retain the model with the best validation loss.

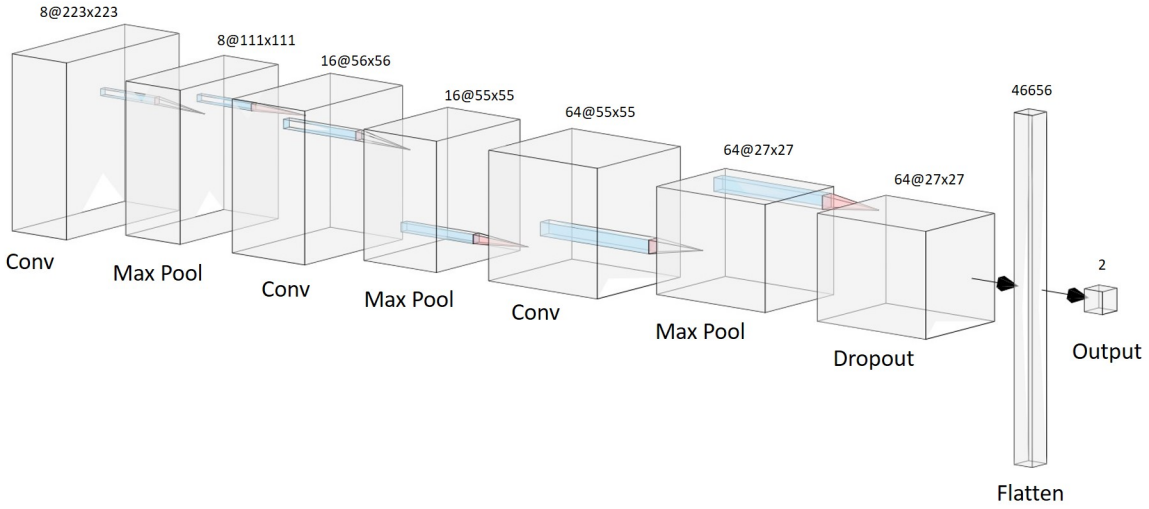

Figure 3: Customised CNN Architecture

3.2 Transfer Learning

In addition to using a customised architecture, I utilise three pre-trained models on the dataset. This practice is termed as transfer learning and it allows using models pre-trained on large datasets on an alternative dataset. I use ResNet-50, InceptionV3 and NasNetMobile which are available to use in Keras. ResNet-50 and InceptionV3 were also utilised by Rajaraman et al (2018) whilst NasNetMobile was not.

The ResNet-50 CNN architecture was pre-trained on approximately 1.28 million images from the ImageNet 2012 classification dataset that consisted of 1000 classes. Their architecture consists of 152 layers which obtains 3.57% error on the ImageNet dataset and won 1st place on the ILSVRC 2015 classification task.

The InceptionV3 CNN architecture was also pre-trained on approximately 1.28 million images from the 2014 ImageNet Large Scale Visual Recognition Challenge that consisted of 1000 classes.

The NasNetMobile CNN architecture was pre-trained on the CIFAR-10 dataset which holds 60000 images with 10 classes and then applied to the ImageNet dataset. This model achieved a 2.4% error rate on CIFAR-10.

As these were trained using different images than my dataset, my methodology for all three transfer learning based CNN's follows Esteva et al (2017) who utilised the InceptionV3 CNN:

1. Remove the last fully connected layer and replace with a layer matching the number of classes in the new data set.
2. Retrain the network from scratch with randomly initialized weights

All three CNNs are trained using backpropagation. All layers of the network are fine-tuned with the global learning rate of 0.001 using the Adam optimizer. The CNNs use a categorical cross-entropy loss function to minimise the distance between the predicted and actual probability distributions. They were run for 20 epochs with a batch size of 20 and the models are cross-validated whilst the model with the best validation loss is retained.

4 Results

The results obtained from running the customised CNN and transfer learning are summarised in Table 3. These report the accuracy, sensitivity and specificity of the model with the lowest validation loss. This shows that the customised architecture performed better than the transfer learning models for all three metrics. In particular, it received a 98.1% sensitivity score which is particularly important for this topic as it measures how many cases of malaria infection we have classified correctly. I argue this is more important than specificity as health professions would prefer to identify all

infection cases compared to non-infection cases. Between the transfer learning models, the ResNet-50 achieved the highest accuracy and specificity whilst NasNetMobile achieved the highest sensitivity.

Table 3: Results

Models	**Accuracy**	**Sensitivity**	**Specificity**
ResNet-50	0.77	0.714	0.826
InceptionV3	0.723	0.781	0.665
NasNetMobile	0.754	0.805	0.704
Customised	0.922	0.981	0.863

Table 4 outlines the results Rajaraman et al (2018) obtained for their ResNet-50 transfer learning model and their customised model. Whilst their customised model received higher accuracy and specificity than the customised model outlined in this paper, they received a lower sensitivity score. Also, their ResNet-50 transfer learning model performed significantly better than the transfer learning model with all three metrics for the same pre-trained network I utilise in this paper. It is unclear what is the reason for this. It could be due to their use of a GPU and an alternative optimiser (SGD optimisiation).

Table 4: Results from Rajaraman et al (2018)

Models	**Accuracy**	**Sensitivity**	**Specificity**
ResNet-50	0.957	0.945	0.969
Customised	0.94	0.931	0.951

As a robustness check, I reran all the models from Table 3 with added 5% Gaussian Noise in the input layer. Adding noise in the input layer will allow us to check if the algorithm generalises well and doesn't overfit. Table 5 shows that the model does generalise well. The results are close to those without Gaussian Noise. In the case of the sensitivity score obtained by the customised CNN, there is a marginal improvement after adding Gaussian Noise. Table 6 displays the confusion matrix for the customised CNN with Gaussian Noise.

Table 5: Results with added Gaussian Noise

Models	**Accuracy**	**Sensitivity**	**Specificity**
ResNet-50	0.744	0.76	0.729
InceptionV3	0.757	0.765	0.749
NasNetMobile	0.741	0.815	0.667
Customised	0.913	0.986	0.841

5 Conclusion

This paper provides an alternative CNN architecture to Rajaraman et al (2018) to classify cell images as being malaria infected or not using the data provided by LHNCBC. This alternative CNN architecture obtained an improved sensitivity score even when compared to transfer learning models in Rajaraman et al (2018) which is arguably more valuable for technicians deciding whether the patient is infected with malaria or not. This paper also uses transfer learning to train pre-trained CNNs. It finds that the customised architecture performs better than the transfer learning models which is unusual for most datasets. However, the transfer learning models utilised by Rajaraman et al (2018) obtained significantly better results than those obtained by the same transfer learning models in this paper.

The models in this paper were trained on a CPU and for 20 epochs. Even though I retained the model with the best validation loss within these epochs, it could possibly obtain improved results if trained for a larger number of epochs. Training this on a CPU limits the number of epochs it could train on due to the longer amount of time required for training. In general, I faced hardware issues with this projects even on pre-processing tasks. I attempted to use three different laptops and Google Colab. On Google Colab and two laptops, the pre-processing task of converting the training images into a 4 dimensional tensor would cause these platforms to crash. This was an issue as it took up a lot of time to run these processes which would eventually fail.

Further research in this area could test the performance of the algorithm to labeling of lab technicians with varying degree of expertise. This could provide us with information as to whom the algorithm would be most useful for. For example, if younger technicians are labeling the cells more incorrectly, the algorithm could be the most useful for them.

Table 6: Confusion Matrix for Customised CNN with Gaussian Noise

	Predicted: No Infection	**Predicted: Infection**
Actual: No Infection	TN = 2320	FP = 440
Actual: Infection	FN = 38	TP = 2714

References

[1] Baratloo A, Hosseini M, Negida A, El Ashal G. Part 1: Simple Definition and Calculation of Accuracy, Sensitivity and Specificity. In *Emerg (Tehran)*, 2015 Spring;3(2):48-9. PubMed PMID: 26495380; PubMed Central PMCID: PMC4614595.

[2] Esteva A., Kuprel B., Novoa R. A., Ko J., Swetter S. M., Blau H. M., Thrun S. Dermatologist-level classification of skin cancer with deep neural networks In *Nature volume 542*, pages 115–118 (02 February 2017)

[3] He K., Zhang X., Ren S., Sun J Deep Residual Learning for Image Recognition *eprint arXiv::1512.03385*, 2015.

[4] Rajaraman, S., Antani, S. K., Poostchi, M., Silamut, K., Hossain, M. A., Maude, R. J., Thoma, G. R. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. In *PeerJ, 6, e4568. 2018*, doi:10.7717/peerj.4568

[5] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z Rethinking the Inception Architecture for Computer Vision *eprint arXiv:1512.00567*, 2015.

[6] Zoph B., Vasudevan V., Shlens J., Le Q. V. Learning Transferable Architectures for Scalable Image Recognition *eprint arXiv:1707.07012*, 2018.

6 Annex - Additional Tables

Table 7: Customised CNN Architecture

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 223, 223, 8)	104
max_pooling2d_1 (MaxPooling2	(None, 111, 111, 8)	0
conv2d_2 (Conv2D)	(None, 110, 110, 16)	528
max_pooling2d_2 (MaxPooling2	(None, 55, 55, 16)	0
conv2d_3 (Conv2D)	(None, 54, 54, 64)	4160
max_pooling2d_3 (MaxPooling2	(None, 27, 27, 64)	0
dropout_1 (Dropout)	(None, 27, 27, 64)	0
flatten_1 (Flatten)	(None, 46656)	0
dense_1 (Dense)	(None, 2)	93314

Total params: 98,106
Trainable params: 98,106
Non-trainable params: 0

Table 8: ResNet-50 Transfer Learning Architecture

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 100352)	0
dense_1 (Dense)	(None, 2)	200706

Total params: 200,706
Trainable params: 200,706
Non-trainable params: 0

Table 9: InceptionV3 Transfer Learning Architecture

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 51200)	0
dense_1 (Dense)	(None, 2)	102402

Total params: 102,402
Trainable params: 102,402
Non-trainable params: 0

Table 10: NasNetMobile Transfer Learning Architecture

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 51744)	0
dense_2 (Dense)	(None, 2)	103490

Total params: 103,490
Trainable params: 103,490
Non-trainable params: 0

Table 11: Confusion Matrix: Resnet with added Gaussian Noise

	Predicted: No Infection	**Predicted: Infection**
Actual: No Infection	TN = 2012	FP = 748
Actual: Infection	FN = 661	TP = 2091

Table 12: Confusion Matrix: Inception with added Gaussian Noise

	Predicted: No Infection	**Predicted: Infection**
Actual: No Infection	TN = 2066	FP = 694
Actual: Infection	FN = 646	TP = 2106

Table 13: Confusion Matrix: Nasnet with added Gaussian Noise

	Predicted: No Infection	**Predicted: Infection**
Actual: No Infection	TN = 1842	FP = 918
Actual: Infection	FN = 510	TP = 2242