Group 3244-2010-0025 - Differentiating Malaria Cells from Normal Cells

Authors: Lee Siying, Dylan Tan Cheng Song, Jennifer Chessa, Samuel Kristianto Tjong, Gerry Lim Wee Kiat, Teo Zhi Yu Dawn
Email: {E0323792, E0323317, E0425355, E0439295, E0324770, E0322943}@u.nus.edu

Abstract

This project aims to build a model that can detect Malaria infected cells with high accuracy. We experimented with some models and decided to focus on one that gives us the best accuracy so far i.e. Convolutional Neural Network (CNN). Knowing there is still room for improvements, we have identified several parameters to experiment with so as to further increase our accuracy.

Introduction

Given that Malaria is no longer endemic, health-care providers may not be familiar with the disease. Clinical diagnosis and detection based on examining the symptoms are difficult as they are usually non-specific to illnesses[1]. The most common method used for detection now is microscopy. However, this can cause potential delays in seeking treatment and even misdiagnosis due to human error when identifying Malaria.

Hence, our group aimed to improve the accuracy of detections using Machine Learning techniques, specifically Convolutional Neural Networks (CNN). Several CNN models were trained with different hyperparameters using infected and non-infected blood cell images.

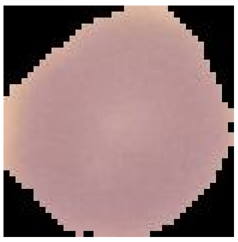
Fig. 1: Uninfected cell

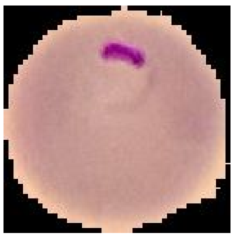
Fig. 2: Parasitised cell

The questions that we decided to answer are as follows:

- Which model was the best to detect Malaria cells?
- How can we further improve the accuracy of the model?
- What hyperparameters/conditions were important for high accuracy

Related Work

Malaria is predominantly found in poorer, tropical and subtropical areas such as Africa, where there are scarce resources and medical experts. It was noted that in 2018, nearly 85% of global malaria deaths were concentrated in 20 countries in the WHO African Region and India and young children aged under 5 years old accounted for 67% of all malaria deaths worldwide[2].

Prompt diagnosis and treatment is needed to prevent mild malaria cases from developing into severe cases or even death. However, it is clear that in those countries mentioned above, there is a lack of awareness of malaria symptoms. Malaria's nonspecific symptoms may cause people to overlook the possibility of them contracting it. Between 2015 and 2018, only a median of 42% of children with a fever were taken to a trained medical provider for care in the public sector, 10% in the formal private sector and 3% in the informal private sector.[2]

Traditionally, one popular method of malaria detection is to use microscopic diagnostics[3], which involves manual counting of parasites in both thick and thin blood smears by microscopists. The execution of this method operates at a low cost as compared to other methods of detection. However, it requires a long process of training for a microscopist to be skilful at reading the microscopic image. Since this method depends heavily on the experience and skill of the microscopist, it is highly susceptible to human error. Especially for countries that do not have the relevant resources to carry out the diagnosis, it is difficult to ensure the accuracy of the microscopist's diagnostic skill. Moreover, microscopy results can only be obtained days to weeks later, which defeats the purpose of prompt diagnosis and treatment.

Another commonly used method to detect malaria is through the usage of RDT or rapid diagnostic test. According to a study on Malaria detection in Nigeria, RDT had a sensitivity of 51.4%, a specificity of 73.2%, a false-positive rate of 26.8% and a false-negative rate of 48.6%.[4] It is apparent that the sensitivity is very low, while the false-negative rate is significantly high. This implies that many malaria-infected patients often go undetected, which is dangerous as malaria is life-threatening when not treated timely.

With the current limitations in diagnosing malaria promptly and accurately, there is a rising need for the use of machine learning in the field of malaria detection. It tackles on the ability to increase the reliability of detecting malaria and also provides a method to extend the testing to more patients at the same time. This reduces the overall dependency on the workers needed for malaria detection, and also cuts the cost of diagnosis.

In the context of this project, we aim to improve malaria detection using the models that we had experimented and trained. The dataset that we used from Kaggle was obtained from the Chittagong Medical College Hospital, Bangladesh[5]. The aim of collecting this image dataset was to minimise the dependency on trained microscopists in countries with low resources, as well as to increase the accuracy of malaria detection. Hence, researchers from the Lister Hill National Centre for Biomedical Communications (LHNCBC) managed to come up with an application that connects the microscope to a regular mobile phone.

Blood-smear slides from 50 healthy patients and 150 malaria-infected patients were used to photograph the cell images, which were then identified and labelled into parasitised or uninfected by a professional microscopist at the Mahidol-Oxford Tropical Medicine Research Unit in Thailand.

With this labelled image dataset, we conducted our training using different models, which will be further explained in the following sections.

Method

Our dataset contained 27,558 images of infected and uninfected blood cell images. We first preprocessed our image dataset by converting the images to arrays of 50 by 50 by 3, and storing the respective image labels. This was done so as to standardise the input array size.

The keras package[6] resizes image arrays to the 50 by 50 by 3 size through a process called Bilinear Interpolation[7]. By "plotting" 50 by 50 evenly spaced points in each of the 3 channels of the original array, the resize function calculates a new value at each point by taking a distance weighted average of its nearest neighbours.

The images are stored in RGB settings, which accounts for the 3 layer channels in each image array. We chose to stick with layers of size 50 by 50, as we tried out multiple numbers (e.g. 16, 32, 150), and decided that 50 was the best number that helped us keep the target features for detecting Malaria in the cells, while keeping the size of each image array small.

We also augmented the dataset by adding in 2 randomly rotated images on top of the original, such that we could account for the images having the infection on any edge of the cell. This helped us to increase the number of training and testing examples.

In the early stage of our project, we experimented with a couple of models, namely Logistic Regression, Perceptron Learning Algorithm, and Convolution Neural Network (Simplified VGG-16) models. The results of each method are shown in the table below.

Model	**Accuracy score (%)**
Logistic Regression	67.5
Perceptron Learning Algorithm (PLA)	51.3
Convolution Neural Network (CNN)	90.8

Fig. 3: Baseline results on validation set

We decided to use CNN as our target model for our image classification problem since it had the highest accuracy results among the three.

For the case of CNN, our team had difficulty determining the architecture we wanted in our CNN model, as there were many potential combinations of layers to choose from. Our final image detection model took inspiration from the highly acclaimed VGG-16[8] architecture.

Our base CNN model used 10 layers by removing some of the Conv2D, MaxPool, and Dense layers. In essence, it has less Convolution and Dense layers as compared to the original.

Moving forward, we focused on tuning the hyperparameters to see if we could achieve greater test accuracy.

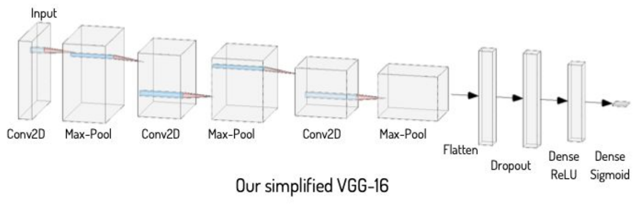

Fig. 4: Base VGG-16 structure, adapted from NN-SVG

Experiments

The very first experiment we did is on the number of layers. We believe it is one of the most significant factors in determining the accuracy of our models as it affects how well our model can extract deterministic features for classification. The full VGG-16 was only able to predict all cells as infected, which was unusual since it is supposed to be a state-of-the-art architecture. We suspected the model might be overfitting and decided to add a dropout layer as a form of regularisation, which yielded much greater results. From the table below, we can see that the results are not necessarily better when we increase the number of layers.

No. of Layers	7	12	16 (Full model no dropout)	16 (Full model with dropout)
Val_Accuracy (%)	96.0	94.0	50.5	94.7

Fig .5: Validation set accuracy of models with different layers

Moreover, we also tried transfer learning using Keras' VGG 16 model. We loaded ImageNet pre-trained weights into the model, removed the top layer and froze the middle layers to ensure that the model is able to train on our cell images while making full use of its already available powerful feature extraction capability. This method yielded a test accuracy of 91.4% and FNR of 6.64%

For the loss function, we decided to use binary cross-entropy since our task is a binary classification task and it can only have two possible outputs, parasitised or uninfected.

For the same reason mentioned above, we chose the sigmoid function to be the activation function for the final output layer. There were other possible activation functions that we could have used such as ReLU and Softmax. However, ReLu was not suitable to be used as the final output layer in this problem as the output was not regression-based. Meanwhile, Softmax is better suited for multi-class classification.

To find the most optimal hyperparameters, we used *Hyperas*[9] which is a wrapper around *hyperopt* for fast prototyping with keras models. It allows us to provide a list of hyper-parameter ranges as inputs and returns the one that produces the best results. Upon receiving an output after running our program with Hyperas, we further experimented with several main hyperparameters manually. Finally, these were our three best combinations of hyperparameters (with highest test accuracy).

Hyper-parameters	**Experiment 1**	**Experiment 2**	**Experiment 3**
Epochs	20	20	50
Optimiser	Adamax	Nadam	RMSprop
Batch size	64	64	50
Learning rate	0.001	0.001	0.0001
Final test accuracy	**95.3%**	**95.1%**	**95.6%**

Fig. 6: Hyperas results

For the optimisers, we experimented with RMSprop, Nadam, Adamax, Adam, SGD and Adadelta.

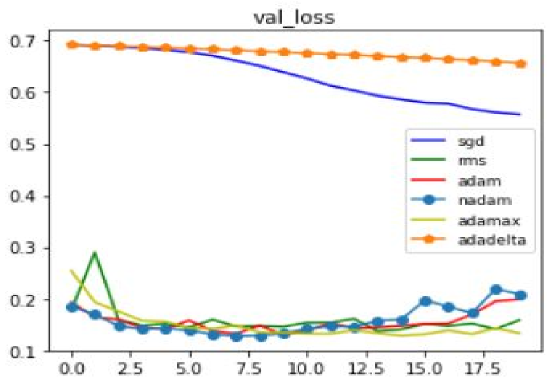

Fig. 7: Optimiser results for 20 epochs, batchsize 64 and learning rate 0.001

From the plot, we can see that Adadelta and SGD are decreasing and RMSprop has low validation loss and also decreasing, in contrast to the other optimisers that are starting to increase after 20 epochs. Thus, we narrowed down to SGD for our optimisers.

Since our project focuses on disease detection, we recognised that the cost of wrongly classifying an actually infected cell is disproportionately high. As such, when fitting the model we decided to assign the parasitised data points different weights to find out which would give us the lowest false negative rate. We thus set the penalty of predicting an infected cell as 4, and predicting an uninfected cell as 1. The results are as follows.

Weights	2	4	6	8	10
Test Accuracy (%)	88.8	92.4	90.5	89.5	88.5
False Negative Rate (%)	12.4	8.57	3.71	3.17	2.5

Fig. 8: Effect of weights on FNR

We then chose the final model to a VGG-16 with lesser layers as illustrated below in Figure 9.

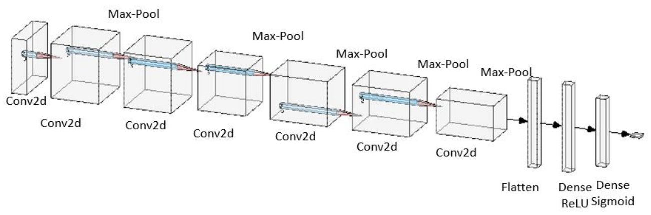

Fig. 9: Final Modified VGG-16 Model, adapted from NN-SVG[17]

Evaluation

Accuracy

Our dataset consists of an equal number of infected and uninfected images, so we did not face the problem of imbalanced classes for our image classification.

For the baseline target, we chose the following:

- Reasonable baseline accuracy target: 91.25%
- Our team's ideal accuracy target: 98%

91.25% is a reasonable baseline accuracy target as it was found that most deep learning models which aim to detect diseases using medical images on average achieve 88.6% sensitivity and 93.9% specificity. These results are better than the predictions made by health-care professionals who on average achieve 79.4% sensitivity and 88.1% specificity. As such, we believe that 91.25% is a reasonable baseline accuracy target for our model which we obtained by working out the average between 88.6% and 93.9%[10].

As for our team's ideal accuracy target, we chose it to be 98% as our first run using the simplified VGG-16 model already returned 90.8% accuracy which encouraged us to experiment with the model to further boost our prediction.

Upon deliberating on possible causes for the difficulty in improving our accuracy, we suspected that there could have been edge cases where the cells border on being infected or parasitised, making it hard to determine its actual status. As such, it is possible that we will get higher accuracy if we increase the dimension of our images.

Misclassified Images

Metric	**Final-CNN-Value (%)**	**Base-CNN-Value (%)**
False Positive Rate	7.83	7.94
False Negative Rate	2.72	4.53
Accuracy	94.7	93.8
F1-score	94.8	93.8
Precision	92.5	92.3
Recall	97.2	95.5

Fig. 10: Values of various metrics used for analysing final and base models

Labels	**Predicted Uninfected**	**Predicted Parasitised**
Actual Uninfected	8091	197
Actual Parasitised	513	7734

Fig. 11: Confusion matrix of Final Modified VGG-16

The final model had a False Positive Rate of about 7.83%, and a False Negative Rate of about 2.71% on the test set, which was much lower as compared to the base model. This model was trained at 40 epochs as we also wanted a model that can quickly converge.

Upon obtaining the above values, our team wanted to find out the distribution of the inaccurate predictions of our model, for possible future analysis. We analysed the inaccurate predictions by plotting them onto these histograms below (Figures 12 and 13). The x axis shows the difference between the real and expected outputs (0 - 1) in which a value of 0.5 means that the prediction is off by a lot while a value of 0.05 means the model almost got it correct. The y- axis shows the number of cell images. From Figure 13, we can see that most of the False Positives were very off in the predictions, while in Figure 12, most False Negatives were close to predicting the correct class.

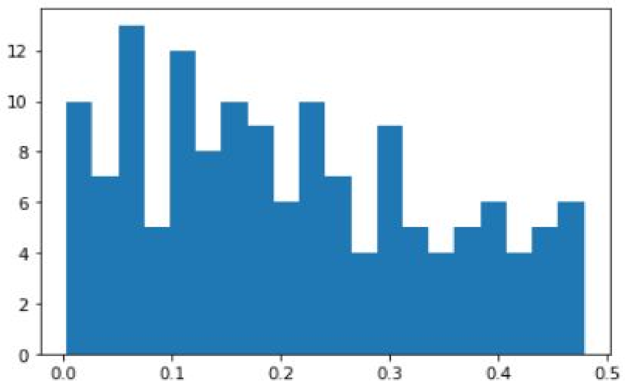

Fig. 12: False Negative Error Distribution

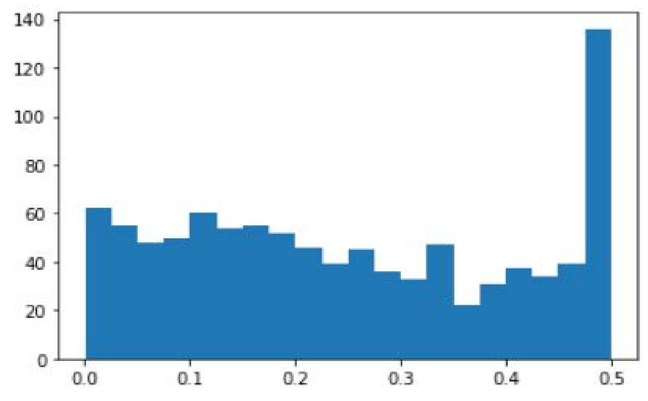
Fig. 13: False Positive Error Distribution

To further analyse the false positives and false negatives, we also printed the false negative and false positive images to try and understand what the root cause was.

For false positives, the images closely resembled parasitised cells, i.e. they have a distinct purple blob within the cell.

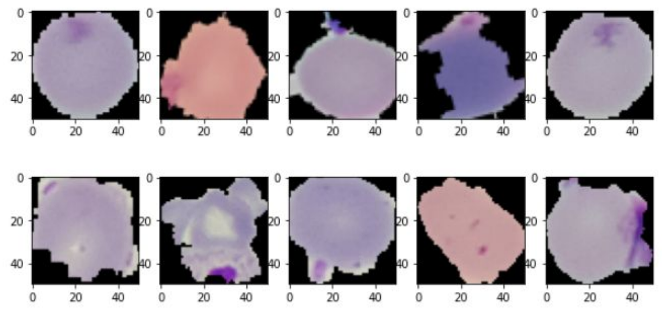
Fig. 14: False Positives (CNN)

It came as a surprise since our team thought that those with purple stains would be marked as infected, but these cells are marked as uninfected instead. Printing out the images of the false negatives, we see a similar contradiction.

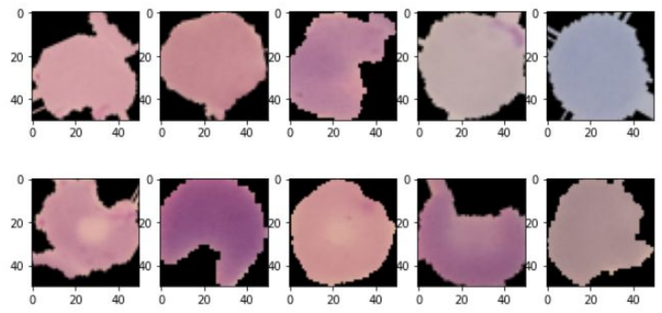
Fig. 15: False Negatives (CNN)

Cells which, by the human eye, would be classified as uninfected, are marked as parasitised. These cells have no distinct purple blob. As we only printed out a small sample of misclassified images, we cannot rule out the possibility that shape or colour is a factor.

Hence, with regards to the misclassified images, we have 2 hypotheses:

1. Some images were wrongly classified in the dataset.
2. Resizing the images caused the majority of the purple blobs to be cropped off, thus the model could not classify it accurately.

As mentioned in point 2 on the cropping of parts of the images, one possible reason to account for the false negatives could be in the way bilinear interpolation works in our resize function. Since the bilinear interpolation took an average of the values near it, the resulting value could have caused important deterministic features to be left out or totally lost[11] through aggregation with the rest of its neighbouring values.

Discussion

By reading up more on Convolutional Neural Networks, we were able to obtain a better understanding of how the hyperparameters and the number of layers might affect the accuracy of the model.

Learning rate affects the training speed of the model, in that low learning rate will result in longer training time and potentially, overfitting. High learning rate might not allow the model to capture the correct signals, but also allows for regularisation[12].

Meanwhile, a larger batch size allows for the usage of a large learning rate. Larger batch sizes tend to have low early losses while training whereas the final loss values are low when the batch size is reduced[12].

Of the two optimisers we decided on, RMSprop makes use of a moving average of the squared gradient and makes use of previously obtained gradient descents to normalize the gradient. Learning rate is adjusted automatically and RMSprop chooses a different learning rate for each parameter. This implies that RMSprop allows for learning rates best suited to each parameter to be chosen[13].

For Stochastic Gradient Descent (SGD), we take each parameter theta θ and update it by taking the original parameter θ and subtract the learning rate η times the ratio of change $\nabla J(\theta)$. SGD is one of the older algorithms for optimisation in neural networks, but nevertheless, it is also comparatively the easiest to learn. However, it also converges slower than newer algorithms like Adam. It also has more problems with being stuck in a local minimum than newer approaches, and underperforms when optimising the cost function[14].

As seen from Figure 13 below, when comparing the loss between the various optimisers over time, most optimisers except Adagrad have similar losses with higher number of iterations. Therefore, it is possible to conclude that with enough iterations, the optimiser used will have a small effect on the results and accuracy of the model.

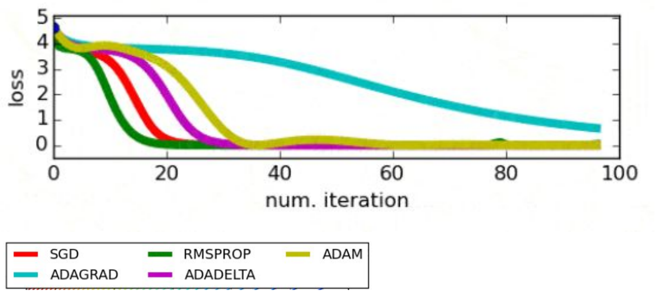

Fig. 16: Comparison of various optimisers[15]

One unexpected outcome we observed after carrying out our experiments is that the more layers used to build the model, does not affect the accuracy positively, as shown by the 50% accuracy for full VGG-16 in Figure 5. Dropout was needed to regulate the model so that it works optimally. We suspect that it might be due to the images not having many deterministic features besides the purple blobs to classify the cell images which makes adding too many layers redundant.

On Misclassified Images

By tracing the data back to its source, we found that the blood cells were "annotated by an expert slide reader at the Mahidol-Oxford Tropical Medicine Research Unit in Bangkok, Thailand"[5]. With the benefit of doubt, if the images are indeed correctly classified, then whether a cell is infected or parasitised could go beyond the visual dimension, as there were cells marked as infected without distinct purple stains, as well as uninfected cells with purple stains.

According to the CDC, malaria parasites are identified by staining blood cells with Giemsa stain, a chemical reagent, to give the malaria parasites a distinctive appearance. Although no direct mention of the Giemsa stain is stated in the data provenance, the distinct purple colour makes it reasonable to assume that it was used to identify the malaria cells in the dataset[1].

Further readings show that the application of Giemsa stain involves soaking the blood specimen in Giemsa solution for 30 minutes before washing with tap water[16]. Thus, there is reason to believe that this process could have contaminated other uninfected cells by chance. A possible solution would be to use different reagents that do not adhere to cells without malaria parasites at all, or another technological innovation that allows visual identification of malaria cells more independently.

Limitations of Using Logistic Regression and PLA

Model	**False Positive Rate (%)**	**False Negative Rate (%)**
Logistic Regression	27.1	31.4
PLA	8.6	65.0
CNN	2.48	6.54

Fig. 17: FPR and FNR of the respective models

Our previous Logistic Regression model had a False Positive Rate of 27.1% and a False Negative Rate of 31.4%, which were substantially higher than those of CNN. From the false negative cells (Figure 19), the majority of them have purple stains within the cell, but were still classified as normal cells.

This showed that Logistic Regression still failed to detect even the easily identifiable parasitised cells (cells with distinct purple stains). Moreover, by comparing Figure 19 with Figure 15, where there were hardly any purple stains within the cell images, we can infer that CNN was more accurate in detecting the parasitised cells. One possible reason for this is because of CNN's ability in identifying important features in an image, which in this case may be the purple stains within the cell images. Meanwhile, Logistic Regression was unable to do so and hence still classified cells with purple stains as normal cells.

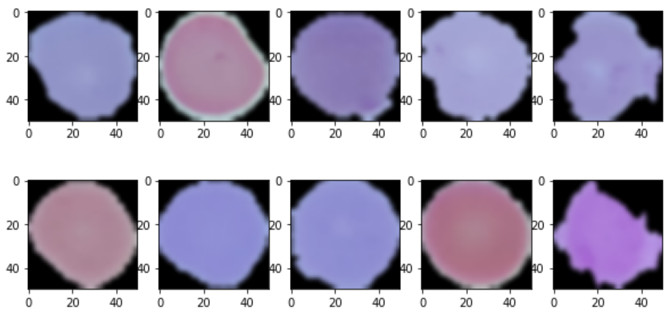
Fig. 18: False Positives (Logistic Regression)

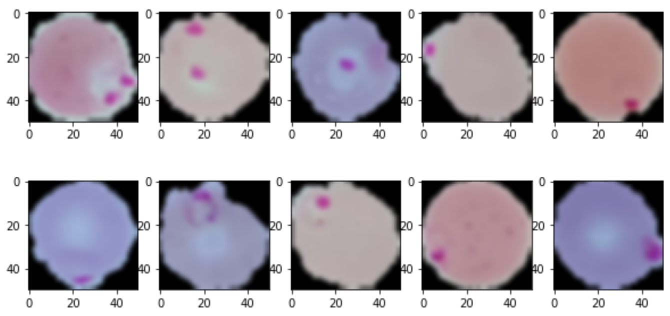
Fig. 19: False Negatives (Logistic Regression)

Similar to our Logistic Regression results, the PLA model also had a higher False Positive Rate and False Negative Rate value as compared to CNN, i.e. 8.6% and 65.0% respectively. The significantly higher False Positive and Negative Rate values were expected, as seen from Figure 21, where almost all of the cell images still had purple stains on them. The previous reason also applies here.

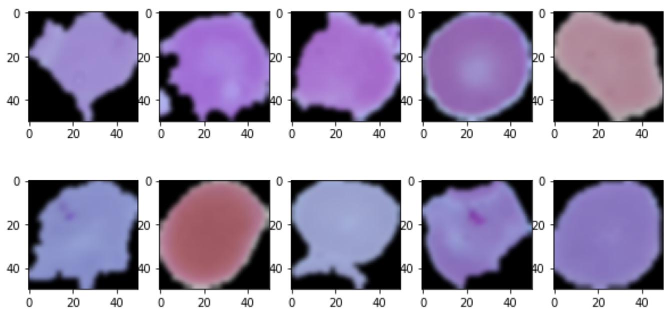
Fig. 20: False Positives (PLA)

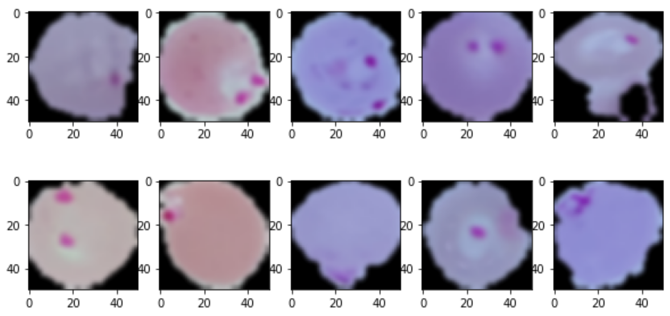
Fig. 21: False Negatives (PLA)

These observations further substantiated our hypothesis that CNN was the best model within the three to detect Malaria cells.

Conclusion

To sum it up, we tested out image classification across a few models, such as Logistic Regression, PLA and CNN. From there, we focused on improving our initial modified VGG-16 as it gave us the best results. For the majority of the span of this project, we attempted to improve the accuracy score of our classification through changing the hyperparameters, number of layers and different optimisers for the base VGG-16 model. The best final accuracy score we obtained from the final VGG-16 model was 94.7%, with a low FNR of 2.72%, which is lower than the base model's FNR of 4.53%.

We also concluded that the number of layers did not correlate with accuracy, as the model could have overfitted on the training set. Dropout was key to regulating models with higher layers, allowing us to receive optimal results when increasing the number of layers. Optimisers did not play a huge part in achieving convergence, but rather the time it took to reach it. However, when training a model, this is inconsequential. Assigning higher weights to false negatives also helped to reduce the FNR without accuracy and FPR performing too poorly.

To further improve the accuracy of our model, we also segmented our dataset from our confusion matrix for the models that we attempted, and observed the cell images of False Positives and False Negatives to find out the possible reasons for the wrong classification. For our CNN model, the resizing of the images could have led to key sections of the cell image, such as the majority of the purple blobs in parasitised cells, to be lost. Another possibility is that the images were either wrongly classified, or the process to stain the cells for malaria identification would stain other uninfected cells as well, making identification of malaria difficult for the model.

Although we managed to achieve high accuracy, our CNN model still made mistakes when it comes to determining cells with purple stains, or cells with an overall purplish hue to them, just like in Logistic Regression and PLA. Logistic Regression and PLA, however, had difficulty in classifying images that clearly belong to specific classes. This shows that both Logistic Regression and PLA are not well suited for image detection as compared to the CNN model.

One possible way moving forward will be to use an autoencoder so as to compress our inputs, without feature loss, instead of trying different array sizes to obtain the best array size. Using this, we will be able to train our model with inputs that correctly reflect the features needed to detect Malaria, and improve our CNN's prediction capability.

Another possible way moving forward will be also to use ensemble methods to further increase the accuracy score and subsequently FNR. The well-known ResNet50 architecture may be considered to form the ensemble.

Acknowledgements

TA Leow Wen Bin for giving us guidance during the early stage of our project and other TAs and Professor Kan Min-Yen who attended our interim presentation for giving us constructive feedback and useful suggestions.

References

[1] CDC - Malaria - Diagnosis & Treatment (United States) - Diagnosis (U.S.). (2018, July 23). Retrieved September 27, 2020, from https://www.cdc.gov/malaria/diagnosis_treatment/diagnosis.html

[2] The "World malaria report 2019" at a glance. (n.d.). Retrieved October 20, 2020, from https://www.who.int/news-room/feature-stories/detail/world-malaria-report-2019

[3] Poostchi, M., Silamut, K., Maude, R. J., Jaeger, S., & Thoma, G. (2018). Image analysis and machine learning for detecting malaria. *Translational Research*, *194*, 36–55. Retrieved September 27, 2020, from https://doi.org/10.1016/j.trsl.2017.12.004

[4] Evaluation of Diagnostic Accuracy of Rapid Diagnostic Test for Malaria Diagnosis among Febrile Children in Calabar, Nigeria. Retrieved October 20, 2020, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6668302/

[5] Malaria Datasets | National Library of Medicine. (n.d.). Retrieved September 14, 2020, from https://lhncbc.nlm.nih.gov/publication/pub9932

[6] Keras-Team. (n.d.). Approach keras used for resizing (interpolation) · Issue #8109 · keras-team/keras. Retrieved from September 14, 2020, https://github.com/keras-team/keras/issues/8109

[7] *Bilinear Interpolation*. (n.d.). Lecture. Retrieved November 2, 2020, from http://web.pdx.edu/~jduh/courses/geog493f09/Students/W6_Bilinear%20Interpolation.pdf

[8] Thakur, R. (2019, August 6). Step by step VGG16 implementation in Keras for beginners. Retrieved on September 22, 2020 from https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

[9] Maxpumperla. (n.d.). Maxpumperla/hyperas. Retrieved October 20, 2020, from https://github.com/maxpumperla/hyperas

[10] Liu, X., Faes, L., Kale, A. U., Wagner, S. K., Fu, D. J., Bruynseels, A., . . . Denniston, A. K. (2019). A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: A systematic review and meta-analysis. The Lancet Digital Health, 1(6). doi:10.1016/s2589-7500(19)30123-2

[11] Image Processing – Bilinear Interpolation. Retrieved November 2, 2020, from https://theailearner.com/2018/12/29/image-processing-bilinear-interpolation/

[12] Bokka, K. (2019, May 05). Guide to choosing Hyperparameters for your Neural Networks. Retrieved October 20, 2020, from https://towardsdatascience.com/guide-to-choosing-hyperparameters-for-your-neural-networks-38244e87dafe

[13] Khandelwal, R. (2019, February 04). Overview of different Optimizers for neural networks. Retrieved October 20, 2020, from https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3

[14] Hansen, C. (2019, December 19). Optimizers Explained - Adam, Momentum and Stochastic Gradient Descent. Retrieved October 20, 2020, from https://mlfromscratch.com/optimizers-explained/

[15] Doshi, S. (2020, August 03). Various Optimization Algorithms For Training Neural Network. Retrieved October 20, 2020, from https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

[16] Mokobi, F., & Jilini, L. (2020, February 05). Giemsa Stain- Principle, Procedure, Results, Interpretation: Staining. Retrieved November 2, 2020, from https://microbenotes.com/giemsa-stain-principle-procedure-results-interpretation/#objectives-of-giemsa-stain

[17] NN-SVG. (n.d.). Retrieved November 2, 2020, from https://alexlenail.me/NN-SVG/

Link to dataset + README(Zipped) :
https://drive.google.com/file/d/1b12lt91XHP3S0PbPXbpMsErLBAWO13vj/view?usp=sharing