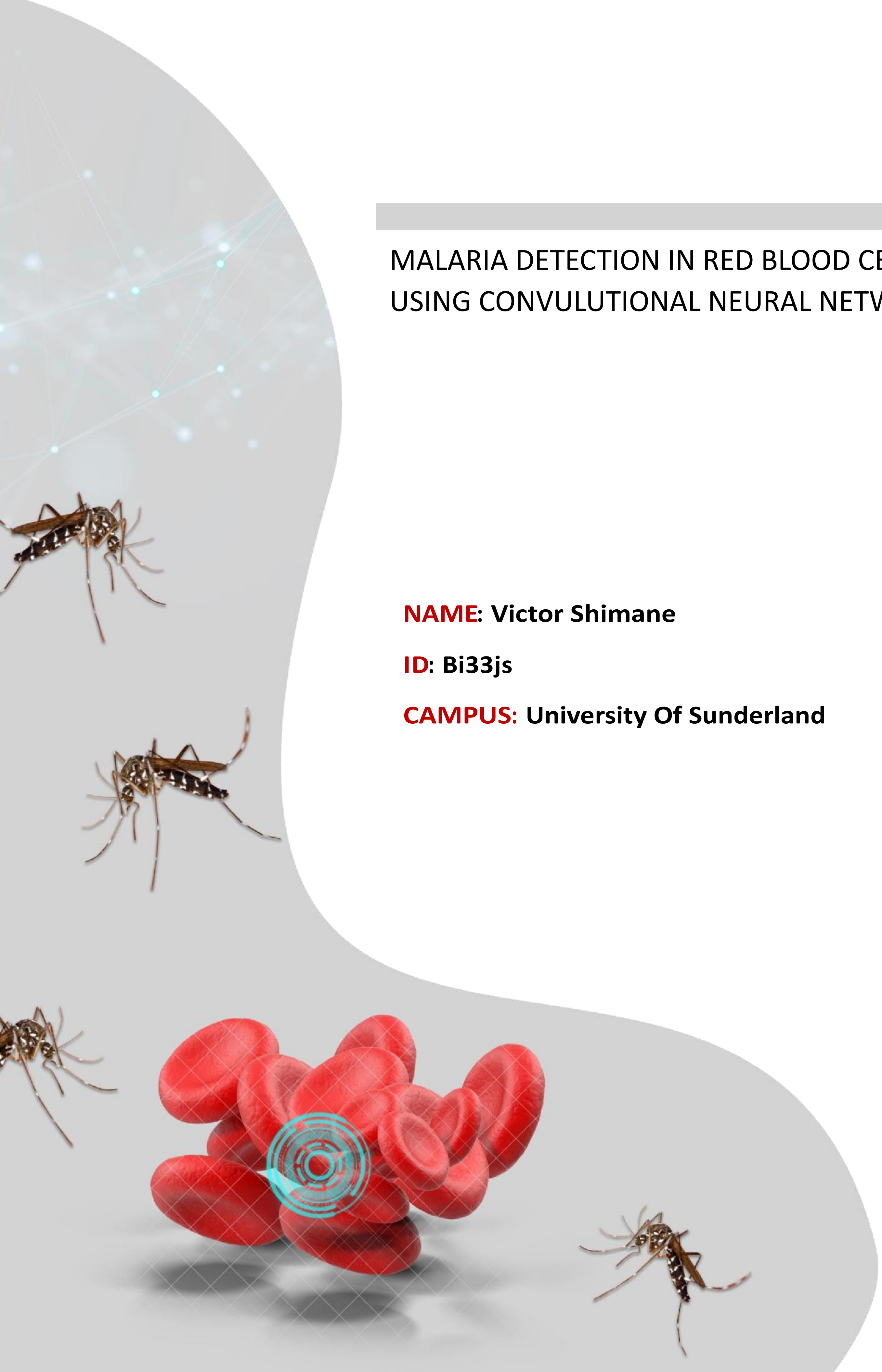
MALARIA DETECTION IN RED BLOOD CELLS USING CONVULUTIONAL NEURAL NETWORKS

NAME: Victor Shimane

ID: Bi33js

CAMPUS: University Of Sunderland



Contents

PROTOTYPE PLANNING

INTRODUCTION

One of the major outcomes for the module CET313 Artificial Intelligence was to help students gain knowledge on vast range of AI technologies which are currently applied in industries or research. This allowed students to be in a better position to select and apply the correct AI techniques for the problems which arise in their current era. As such, to equip students with this skillset they were tasked to develop an Intelligent Prototype to solve a real-life problem in any industry sector using the Python programming as the core foundation. The options to choose from were pathfinding with planning and search or using Machine Learning (ML) to solve a real-world problem.

E-Portfolio Justification

The option discussed in this paper is a machine learning solution - a prototype that detects malaria in red blood cells using Convolutional Neural Networks (CNNs). This option was selected based on the comprehensive knowledge attained during practical tasks and activities from the module. Machine learning, which can be traced backed to the 90's, catapulted by many pioneers who rigorously studied this field – the likes of Arthur Samuel and Frank Rosenblatt made contribution the concept of making intelligent machines a reality. Therefore, treading in the same vision, this option was selected based on the resilience of this field in solving the most complicated problems efficiently. Also, the comprehensive knowledge gained from module activities summarized below:

1. **An Introduction to Artificial Intelligence**

   This section of the module gave an insight into the background of AI and its contribution in society. This was of paramount importance as the research task given in the module space gave a guide on the latest Developments in AI. This research task inspired the selected malaria detection model in red blood cells based on the vast knowledge acquired on how AI is solving problems in different industry sectors. For example, vision-based student attendance systems that use Machine Learning. Moreover, practical tasks like python programming basics had a significant contribution as well since the model was to be built using python.

2. **Introduction to Machine Learning**

   The tasks in the module space gave an overview on the machine learning process based on understanding a problem, the data around it, how to process the data and how to build an actual model. These tasks were essential and are the reason behind the model selected rather than the pathfinding with planning and search prototype. This was due to knowledge gained from the activities that gave an understanding on the power behind data and the information it relays and also the sophisticated patterns in the data that make it possible to make predictive analytics regarding the data.

3. **Other E-Portfolio Tasks**

   The E-Portfolio tasks were indeed the contributing factor to chosen approach, as it covered all relevant Machine Learning tasks that were important for the success of this prototype. From image-based recognition systems to regression-based models the weekly tasks provided current and pertinent information on how to build and implement such models in real life.

SECTION 1: PROTOTYPE IDENTIFICATION AND PLANNING

1.1 Literature Review on Prototype Identification

1.1.1 Problem Scope

Malaria is a mosquito-borne infectious disease that affects humans, it is caused by protozoan parasites belonging to the genus Plasmodium. It is transmitted to human cells by female mosquitoes' species of the Anopheles genus. The Anopheles mosquito uses its proboscis to inject the parasite directly into the human blood which then travels to the liver to mature and reproduce, leading to adverse health effects. Recent reports by the World Health Organization (2019) show a great spike in the number of malaria infections – in the year 2018, approximately 228 million cases of malaria were registered as compared to 231 million cases in 2017. Moreover, in 2018 a massive death toll of 405 000 was recorded compared to the 416 000 recorded in 2017. Malaria is not only a threat to humans but an economic burden in many countries especially in Asian and African countries.

Taxonomists also face the emotional burden of combating this invisible parasite since optical microscopes remain the widely recommended standard means of diagnosing whether a biological sample contains the Plasmodium parasite. The manual process of examining microscope slides is an ineffective and tedious process that has made global responses slow leading to numerous deaths and infections. Taxonomists acquire blood samples and analyze each of them individually through a microscope lens to identify infected and uninfected red blood cells and thereafter cluster them. This process takes months and requires qualified personnel's that oftentimes are in short supply.

1.1.2 Proposed Prototype

Malaria is a global health issue that requires solutions that can aid in faster and more efficient detection to speed up response time by relevant authorities and this starts by cutting down challenges faced in the diagnostic and classification processes. Consequently, the proposed machine learning prototype to solve this problem is a Deep Learning algorithm that capitalizes on images - Convolutional Neural Networks (CNN's). This is an architecture for Deep Learning tailored for visual system structures and was coined by Hubel and Wiesel in 1962. It comprises of multiple trainable multilayer levels that process and classify an image set (Shamsaldin et al, 2019). The proposed model will be crafted to detect malaria parasites in red blood cell images using convolutional neural networks.

1.1.3 Related Work

This section discusses relevant real-life applications in research narrowed to the scope of malaria detection in red blood using vision-based deep learning methods primarily CNN's. The search criteria and terms to compile the papers was done using the following search terms - ***'Image based malaria detection in red blood cells', 'Convolutional Neural Networks malaria detection' and 'vision-based malaria detection'.***

Scenario 1

Dong et al (2017) carried out an experiment using the three well known types of CNN's namely, the LeNet, AlexNet and GoogleNet. Their experiment was aimed at evaluating the three deep learning algorithms against a support vector machine. Their image data set was compiled using whole slide images of thin blood stains, which were classified into infected and uninfected red blood cells labelled by a group of pathologists. The three CNN's were trained to learn the intrinsic features of the malaria infected cells and non-infected cells. For contrasts, the SVM was trained on preselected inherent features collected from the same dataset.

Experiment results indicated classification precisions of over 95% by the CNN'S compared to the 92% attained by the SVM. Bringing the realization that deep learning algorithms have an upper hand advantage since they are able to automatically extract several layers of features from the input data. Dong et al (2017) mentioned in his conclusion that to improve his CNNS' models they will increase the dataset they currently have by including more pathologist-curated cell images to make their model robust and more accurate. Therefore, making it competent for real life applications.

Scenario 2

A more sophisticated implementation was carried out by Masud et al (2020) who leveraged Deep Learning techniques to detect malaria on blood smears. The custom CNN model implemented used a cyclical learning rate schedule coupled with an automatic learning rate finder and also regularization techniques - batch normalization and dropouts to enhance the model performance. Their model achieved a 97.30% accuracy in classifying infected and uninfected cell images. Moreover, when the model was evaluated using the Matthews correlation coefficient (MCC) it yielded a great value of 94.17% compared to other state-of-the-art models; ResNet-50.

Masud et al (2020) went further to convert their best trained TensorFlow model into a TensorFlow Lite model using the TFLite Converter. Thereafter they deployed the model into a mobile application built using the android framework to run on mobile devices. A user opens a cell image then the model predicts whether its infected or not infected. The project was a great success as the model accurately classified the image sets tested using the mobile app. Masud et al (2020) model is promising and can be improved using image augmentation on the training data to alleviate model overfitting problems they experienced.

1.1.4 Software Development Planning

Project planning is essential as it determines the processes involved in building a complete model. ML project planning is iterative, such that each task is looped until it reaches a highly satisfactory level, then proceeding to the next step that might direct back to the previous task performed. Machine learning models are fine-tuned through this iterative process. The following diagram shows the 7-development planning cycle of a Machine Learning project of which will be implemented in this project.

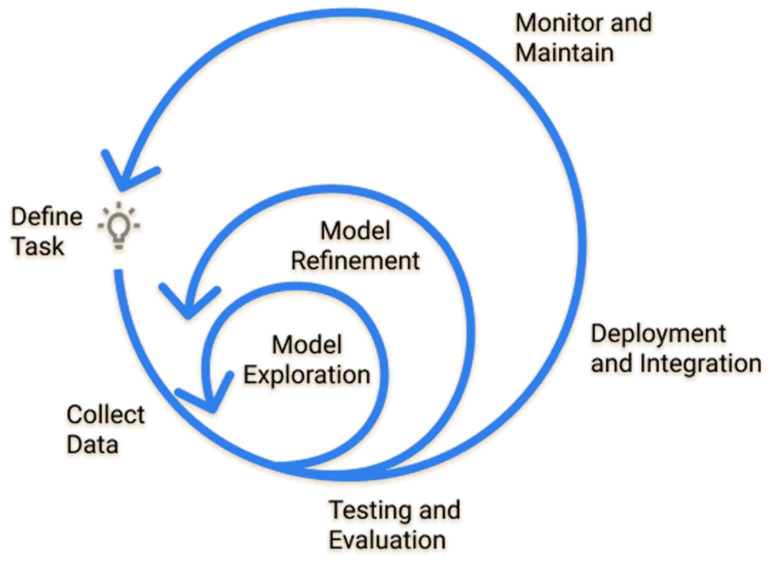
1. Planning and project setup
* Define the task and scope out requirements
* Determine project feasibility
* Discuss general model tradeoffs (accuracy vs speed)
Set up project codebase

2. Data collection and labeling
* Define ground truth (create labeling documentation)
* Build data ingestion pipeline
* Validate quality of data
* Label data and ensure ground truth is well-definend
* Revisit Step 1 and ensure data is sufficient for the task

3. Model exploration
* Establish baselines for model performance
* Start with a simple model using initial data pipeline
* Overfit simple model to training data
* Stay nimble and try many parallel (isolated) ideas during early stages
* Find SoTA model for your problem domain (if available) and reproduce results, then apply to your dataset as a second baseline
* Revisit Step 1 and ensure feasibility
* Revisit Step 2 and ensure data quality is sufficient

4. Model refinement
* Perform model-specific optimizations (ie. hyperparameter tuning)
* Iteratively debug model as complexity is added
* Perform error analysis to uncover common failure modes
* Revisit Step 2 for targeted data collection and labeling of observed failure modes

5. Testing and evaluation
*Evaluate model on test distribution; understand differences between train and test set distributions (how is "data in the wild" different than what you trained on)
*Revisit model evaluation metric; ensure that this metric drives desirable downstream user behavior
Write tests for:
Input data pipeline
Model inference functionality
Model inference performance on validation data
Explicit scenarios expected in production (model is evaluated on a curated set of observations)



6. Model deployment
* Expose model via a REST API
* Deploy new model to small subset of users to ensure everything goes smoothly, then roll out to all users
*Maintain the ability to roll back model to previous versions
* Monitor live data and model prediction distributions

7. Ongoing model maintenance
* Understand that changes can affect the system in unexpected ways
* Periodically retrain model to prevent model staleness
* If there is a transfer in model ownership, educate the new team

Figure 1.0 : Machine Learning Software Development Process (Jordan, 2018).

1.2 Reflection on the Prototype Identification

Based on the knowledge acquired from the project scope, malaria is indeed a deadly disease that has caused havoc worldwide. The appalling statics regarding the number of infections and death rates by the WHO proves the devasting effects of this disease both economically and socially. According to Centers of Disease Control and Prevention (2020) if paramount malaria combating interventions are significantly disrupted due to COVID-19 challenges or any other pandemic, the numbers of malaria cases and deaths will rise exponentially. Hence technologies found in Machine learning have dug deep to solve this problem looking at the solutions by the authors mentioned. The information drawn from their solutions show a need to delve deep in building detection models that can be used globally to aid taxonomist in their line of work - saving lives.

The selected model in this project was difficult to pick due to the complexity behind it, brought by lack of knowledge on how these deep models are built. But the essential information attained from the literature reviewed and Machine Learning Software Development Process discussed beamed light that it is possible to make such a model. Moreover, the activities on the module works space regarding how to build a CNN model from scratch brought forth the conviction and vital skills on how to build this malaria detection in red blood cells prototype.

SECTION 2: PROTOTYPE DEVELOPMENT

Section 2.1 Developed code and planning documents for prototype.

2.1.1 Prototype Inspiration and Benchmarks

To develop the prototype, an assessment was undertaken on numerous implementations by machine learning experts on GitHub and other code repositories to see their approaches related to the selected topic and model functionalities. During the search strategies a prototype was discovered designed by Laxmi Kant a machine learning enthusiast and a Lead Data Scientist at KGPTalkie. The inspiration drawn from Laxmi's code was that he developed a Malaria Parasite Detection Model Using CNN. The implementation of the model he built was a basic sequential CNN model made up with 406,625 trainable parameters. The model was trained with   segmented cells from the thin blood smear slide images summing up to 5510 image datasets of both infected and uninfected cells. Results achieved from his model were promising based on the 91.9 % validation accuracy attained. The model proposed faced a challenge of overfitting based on the 94.6 training accuracy thus in this project the aim was to capitalized on new strategies such as effective code usage, image augmentation and transfer learning to refine Laxmi's' model.

Git Link to Laxmi's' Repository - https://github.com/laxmimerit/Malaria-Classification-Using-CNN.git

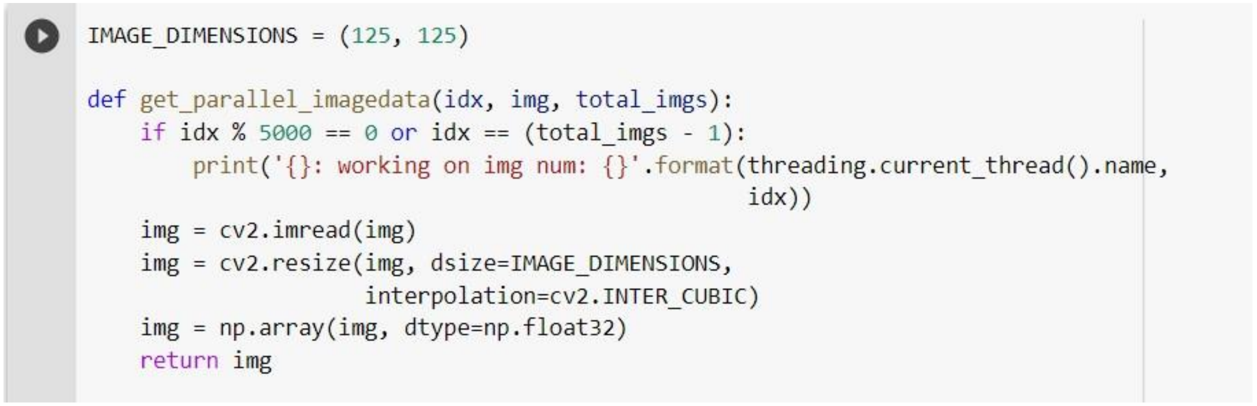
2.2.2 Code Development Process

The model developed in this project is crafted using python programming language and is a deep learning model using Convolutional Neural Networks. This section will entail of a short summary on the development process of the malaria detection model in red blood cells.

DATASET PREPATION

The dataset involved in this project is acquired from the researchers at the Lister Hill National Center for Biomedical Communications (LHNCBC), part of the National Library of Medicine (NLM). The dataset consists of 27563 image datasets segmented into 13784 infected red blood cells and 13779 uninfected cells. The dataset underwent a data exploration process to get insights and clear understanding on the categorical nature of each cell based on its morphology.

A process of data wrangling was carried out on the data, mainly consistent image scaling and resizing hence standard image ratio of 125 by 125 pixels was applied on the dataset for uniformity.



```
IMAGE_DIMENSIONS = (125, 125)

def get_parallel_imagedata(idx, img, total_imgs):
    if idx % 5000 == 0 or idx == (total_imgs - 1):
        print('{}: working on img num: {}'.format(threading.current_thread().name,
                                                   idx))
    img = cv2.imread(img)
    img = cv2.resize(img, dsize=IMAGE_DIMENSIONS,
                     interpolation=cv2.INTER_CUBIC)
    img = np.array(img, dtype=np.float32)
    return img
```

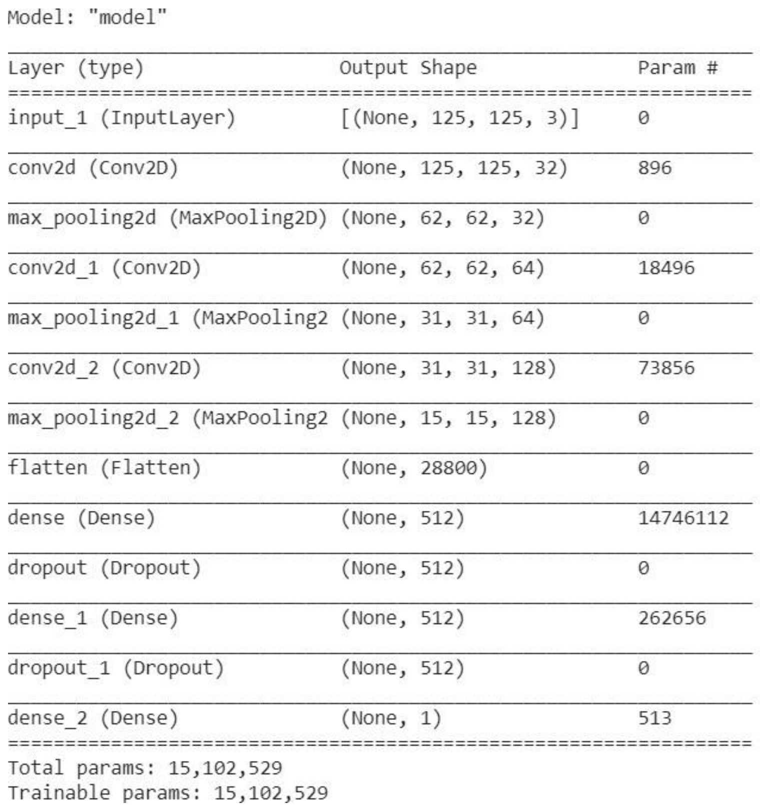
Figure 2.0 : Code snippet of consistent image scaling.

TECHNOLOGY SELECTION

To make a deep learning model that scales to handle and process this huge dataset it was essential to put into consideration the need for powerful hardware in the sense of GPU, thus Google Colab pro was the tool selected. The runtime configurations used was a TPU accelerator and a runtime shape consisting of high RAM, this option was worth choosing considering that cloud-based processing is more efficient and offers safe storage.

MODEL DESIGN

Convolutional neural networks are built with different layers that form trainable building blocks, as such the model architecture of this prototype is built from scratch. It consists of three convolutions, three pooling layers, two dense layers and dropouts for model regularization. The model is then trained, analyzed, and saved.



```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 125, 125, 3)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 125, 125, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 62, 62, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 31, 31, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 15, 15, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 28800)             0
_________________________________________________________________
dense (Dense)                (None, 512)               14746112
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 15,102,529
Trainable params: 15,102,529
```

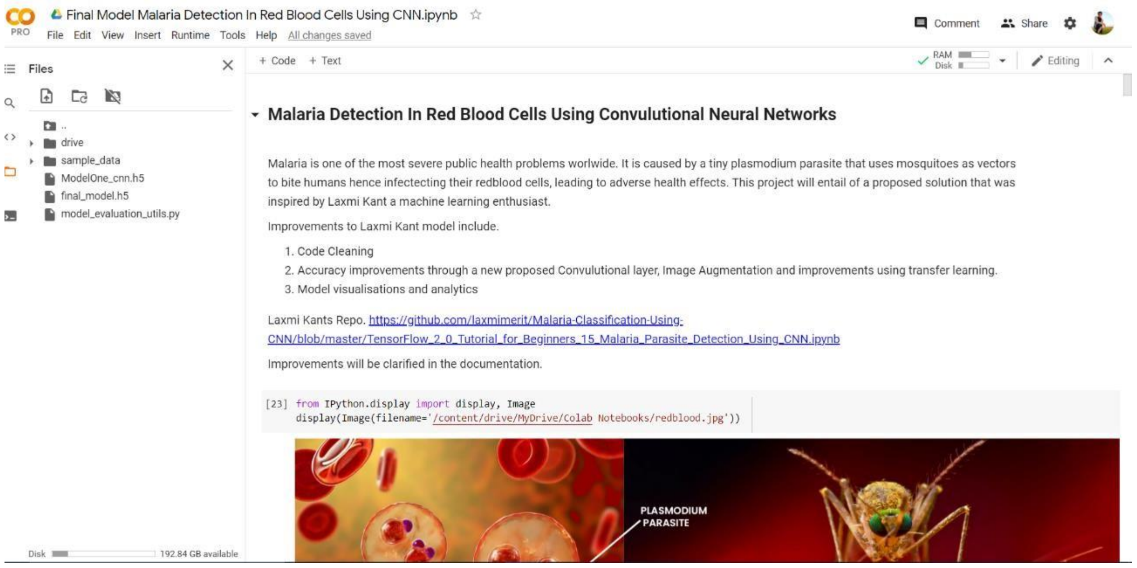
Figure 3.0 : Model Summary.

2.2.3 Steps to Execute model

The Malaria Detection Model has been hosted on the cloud as mentioned, to access the model click the Google drive link provided. The link will open a Google Colab project titled Final Model - Malaria Detection in Red Blood Cells Using CNN.ipynb. To run the model, select the runtime tab and click on run all.

1. Step 01 : Open the link below on a browser. https://colab.research.google.com/drive/1bTf61nTm7fBIxdFbAePCB34sLThJct2u?usp=sharing

2. This will redirect you to a Colab project as per the image below.


Final Model Malaria Detection In Red Blood Cells Using CNN.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
Files
..
drive
sample_data
ModelOne_cnn.h5
final_model.h5
model_evaluation_utils.py
Disk 192.84 GB available
PLASMODIUM PARASITE

Malaria Detection In Red Blood Cells Using Convulutional Neural Networks

Malaria is one of the most severe public health problems worlwide. It is caused by a tiny plasmodium parasite that uses mosquitoes as vectors to bite humans hence infectecting their redblood cells, leading to adverse health effects. This project will entail of a proposed solution that was inspired by Laxmi Kant a machine learning enthusiast.

Improvements to Laxmi Kant model include.

1. Code Cleaning
2. Accuracy improvements through a new proposed Convulutional layer, Image Augmentation and improvements using transfer learning.
3. Model visualisations and analytics

Laxmi Kants Repo. https://github.com/laxmimerit/Malaria-Classification-Using-CNN/blob/master/TensorFlow_2_0_Tutorial_for_Beginners_15_Malaria_Parasite_Detection_Using_CNN.ipynb

Improvements will be clarified in the documentation.

```
[23] from IPython.display import display, Image
     display(Image(filename='/content/drive/MyDrive/Colab Notebooks/redblood.jpg'))
```

Figure 4.0 : Project File in Google Colab.

3. Upload this dataset to your Google Drive space and mount you drive on the files panel.

Link to the dataset to be uploaded to Colab notebooks on google drive.
https://drive.google.com/drive/folders/1MqJF6y7gqK4pgj8OmsEkWXvQQ2BPSdO?usp=sharing

4. On the notebook click on the runtime tab to execute the model.

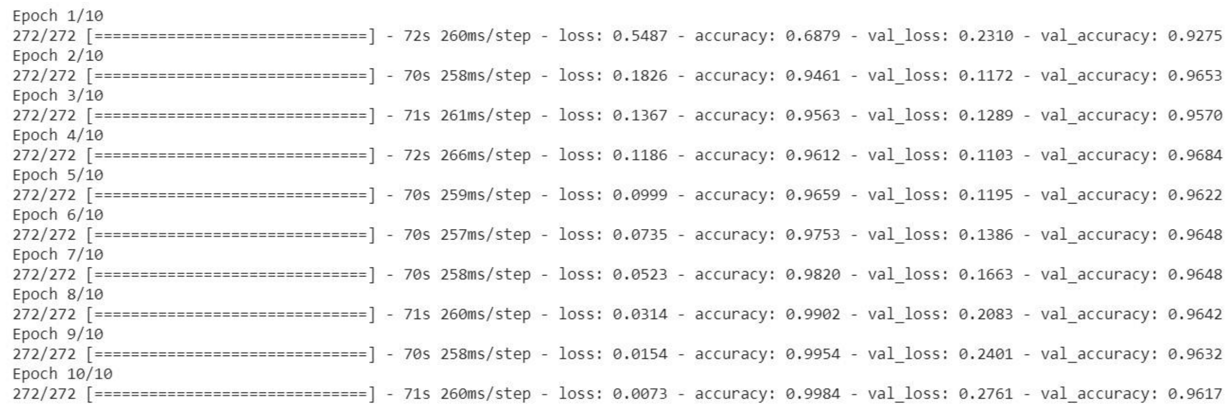
SECTION 3: EVALUATION

This section entails model training, testing and evaluations based on four performance metrics. Moreover, discussions on alterations implemented to improve the model.

3.1.1 TRAINING & VALIDATION PROCESS

Consequently, after building the architectural structure of the malaria detection model the training process came after. This process involved exposing the developed model to infected and uninfected red blood cells dataset that was explored and prepared. The dataset was split into a ratio 60:10:30 – (train, validation, and testing) and an epoch initialization of 10 rounds for the model to go through the entire dataset. Below are results from this process.

RESULT ANALYSIS



```
Epoch 1/10
272/272 [==============================] - 72s 260ms/step - loss: 0.5487 - accuracy: 0.6879 - val_loss: 0.2310 - val_accuracy: 0.9275
Epoch 2/10
272/272 [==============================] - 70s 258ms/step - loss: 0.1826 - accuracy: 0.9461 - val_loss: 0.1172 - val_accuracy: 0.9653
Epoch 3/10
272/272 [==============================] - 71s 261ms/step - loss: 0.1367 - accuracy: 0.9563 - val_loss: 0.1289 - val_accuracy: 0.9570
Epoch 4/10
272/272 [==============================] - 72s 266ms/step - loss: 0.1186 - accuracy: 0.9612 - val_loss: 0.1103 - val_accuracy: 0.9684
Epoch 5/10
272/272 [==============================] - 70s 259ms/step - loss: 0.0999 - accuracy: 0.9659 - val_loss: 0.1195 - val_accuracy: 0.9622
Epoch 6/10
272/272 [==============================] - 70s 257ms/step - loss: 0.0735 - accuracy: 0.9753 - val_loss: 0.1386 - val_accuracy: 0.9648
Epoch 7/10
272/272 [==============================] - 70s 258ms/step - loss: 0.0523 - accuracy: 0.9820 - val_loss: 0.1663 - val_accuracy: 0.9648
Epoch 8/10
272/272 [==============================] - 71s 260ms/step - loss: 0.0314 - accuracy: 0.9902 - val_loss: 0.2083 - val_accuracy: 0.9642
Epoch 9/10
272/272 [==============================] - 70s 258ms/step - loss: 0.0154 - accuracy: 0.9954 - val_loss: 0.2401 - val_accuracy: 0.9632
Epoch 10/10
272/272 [==============================] - 71s 260ms/step - loss: 0.0073 - accuracy: 0.9984 - val_loss: 0.2761 - val_accuracy: 0.9617
```

Figure 4.0 : CNN model training set to 10 Epochs.

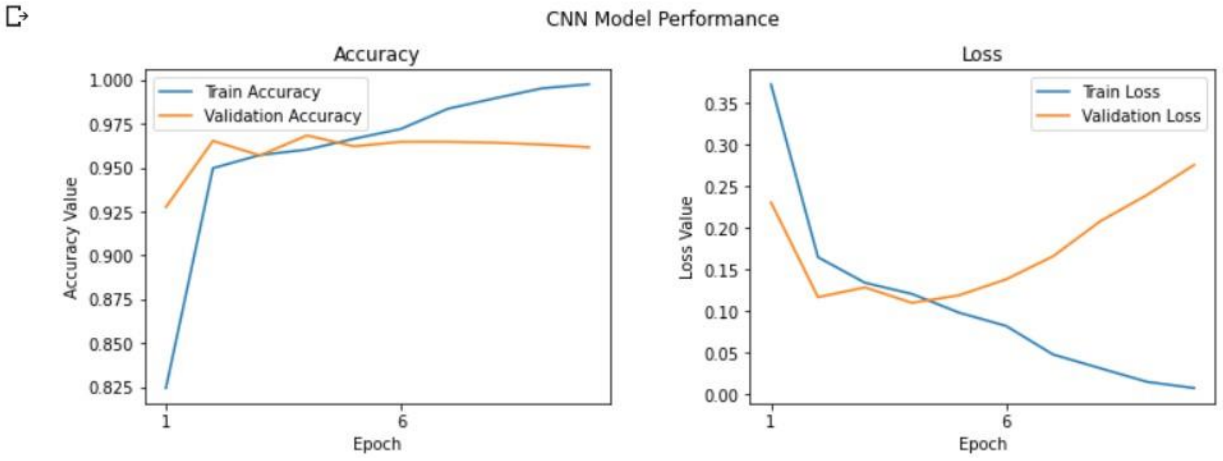


Figure 5.0 : CNN model training accuracy, validation accuracy and validation loss

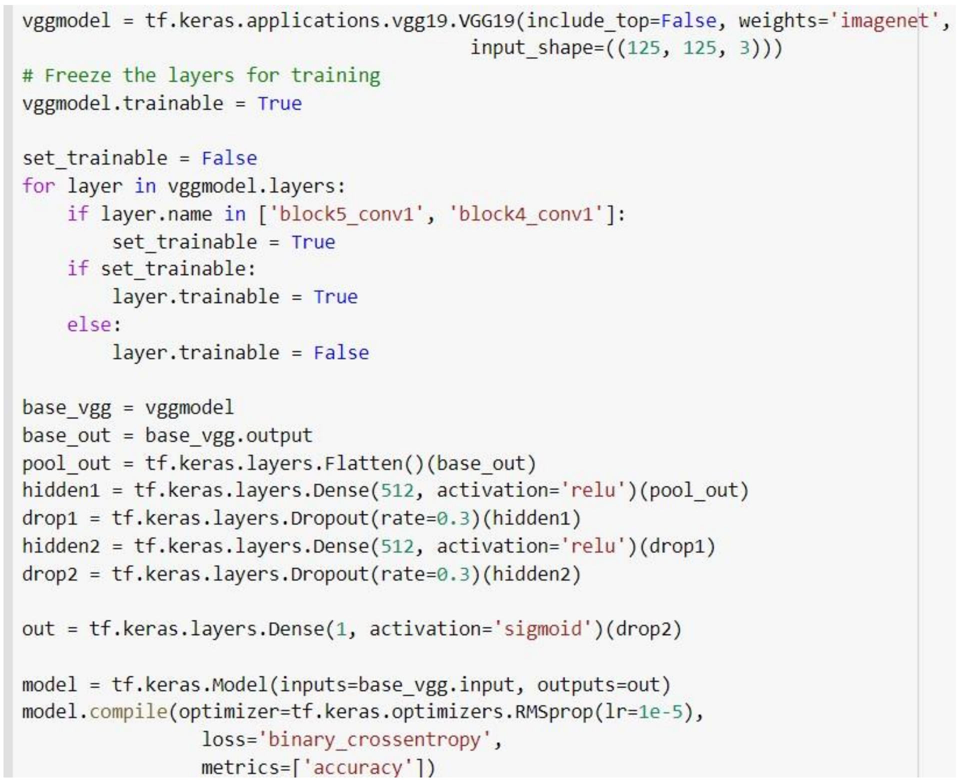
Based on the result above the validation accuracy of the CNN model is 96.2% which very good and shows that the model is learning patterns and similarities in the image dataset that it has been exposed and is able to predict and classify new data brought before it. But the challenge seen is that the model is overfitting based on the 99.8% training accuracy and this is clearly perceived using the training, validation accuracy and loss curves. Right at the 5th Epoch the overall model seems not to be improving at all.

MODEL IMPROVEMENT PROCESS

The malaria detection model proposed is promising but faces a challenge of overfitting. This means the noise or random fluctuations in the training data is picked up and learned as concepts by the model. But these concepts do not apply or relate to new data thus alters the model's aptitude to generalize.

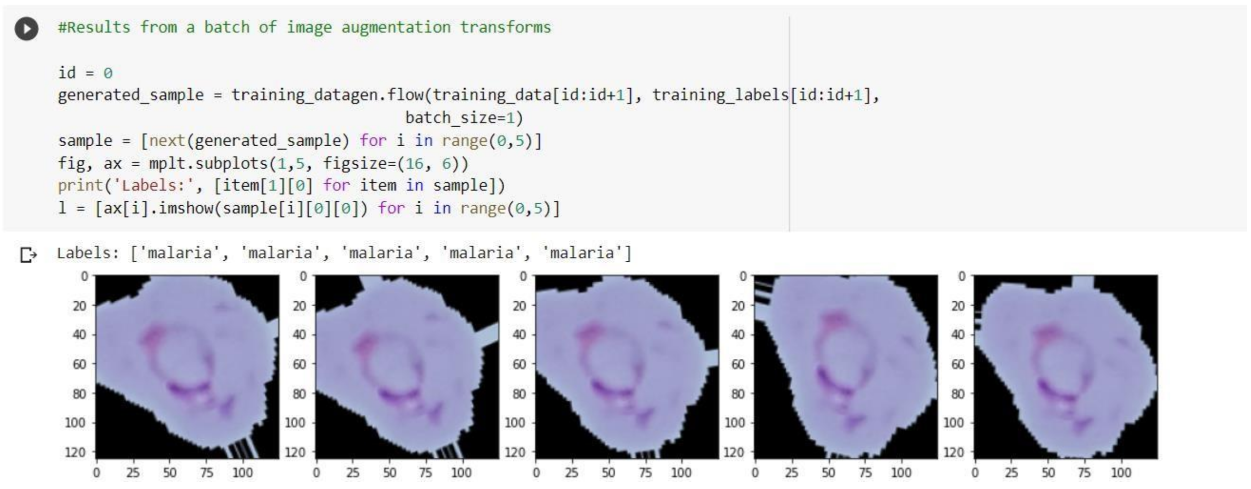
Techniques such as dropouts and regularization we applied on the basic CNN model but still failed to achieve quality results for the model. To combat this, a research was carried out on techniques to overcome model overfitting and Perez et al (2017) discussed a concept of image augmentation and how it reduces model overfitting and allows model training with more epochs without leading to overfitting. This research process led to learning a new concept of transfer learning, which according to Brownlee (2017) this is a process used to overcome the isolated learning paradigm and using domain knowledge from one pretrained CNN to train another model with an aim of increasing convergence rates and accuracy.

Therefore, from this concept discovered from these two papers the strategy to improve this model was to use a pretrained VGG-19 deep learning model. This model was considered since it has been trained on a huge dataset with vast image categories hence it has learned the hierarchy of features, which are spatial-, rotational-, and translation-invariant learnt by CNN models. After building the model it will be trained using an augmented training dataset.



```
vggmodel = tf.keras.applications.vgg19.VGG19(include_top=False, weights='imagenet',
                                        input_shape=((125, 125, 3)))
# Freeze the layers for training
vggmodel.trainable = True

set_trainable = False
for layer in vggmodel.layers:
    if layer.name in ['block5_conv1', 'block4_conv1']:
        set_trainable = True
    if set_trainable:
        layer.trainable = True
    else:
        layer.trainable = False

base_vgg = vggmodel
base_out = base_vgg.output
pool_out = tf.keras.layers.Flatten()(base_out)
hidden1 = tf.keras.layers.Dense(512, activation='relu')(pool_out)
drop1 = tf.keras.layers.Dropout(rate=0.3)(hidden1)
hidden2 = tf.keras.layers.Dense(512, activation='relu')(drop1)
drop2 = tf.keras.layers.Dropout(rate=0.3)(hidden2)

out = tf.keras.layers.Dense(1, activation='sigmoid')(drop2)

model = tf.keras.Model(inputs=base_vgg.input, outputs=out)
model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=1e-5),
                loss='binary_crossentropy',
                metrics=['accuracy'])
```

Figure 6.0: VGG 19 Model architecture.



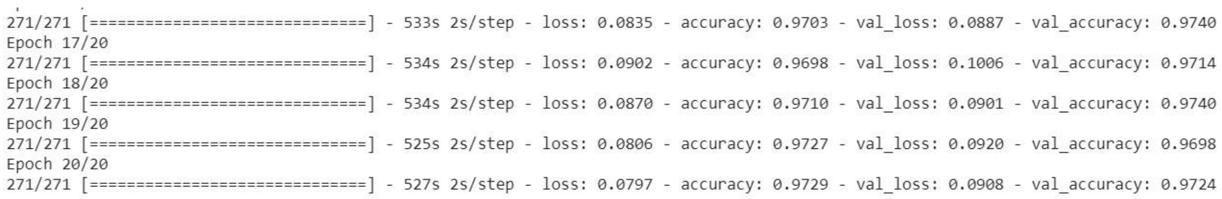
```
#Results from a batch of image augmentation transforms

id = 0
generated_sample = training_datagen.flow(training_data[id:id+1], training_labels[id:id+1],
                                         batch_size=1)
sample = [next(generated_sample) for i in range(0,5)]
fig, ax = mplt.subplots(1,5, figsize=(16, 6))
print('Labels:', [item[1][0] for item in sample])
l = [ax[i].imshow(sample[i][0][0]) for i in range(0,5)]
```

```
Labels: ['malaria', 'malaria', 'malaria', 'malaria', 'malaria']
```

Figure 7.0: Results from Image augmentation transforms.

RESULTS FROM MODEL ENHANCEMENTS

The VGG model was trained and achieved a validation accuracy of 97.2% and training accuracy of 97.3%. The model was not experiencing overfitting and had outperformed the first basic cnn model. This can be seen with the diagrams below.



```
271/271 [==============================] - 533s 2s/step - loss: 0.0835 - accuracy: 0.9703 - val_loss: 0.0887 - val_accuracy: 0.9740
Epoch 17/20
271/271 [==============================] - 534s 2s/step - loss: 0.0902 - accuracy: 0.9698 - val_loss: 0.1006 - val_accuracy: 0.9714
Epoch 18/20
271/271 [==============================] - 534s 2s/step - loss: 0.0870 - accuracy: 0.9710 - val_loss: 0.0901 - val_accuracy: 0.9740
Epoch 19/20
271/271 [==============================] - 525s 2s/step - loss: 0.0806 - accuracy: 0.9727 - val_loss: 0.0920 - val_accuracy: 0.9698
Epoch 20/20
271/271 [==============================] - 527s 2s/step - loss: 0.0797 - accuracy: 0.9729 - val_loss: 0.0908 - val_accuracy: 0.9724
```

Figure 8.0: VGG 19 model training process with 20 epochs configured.

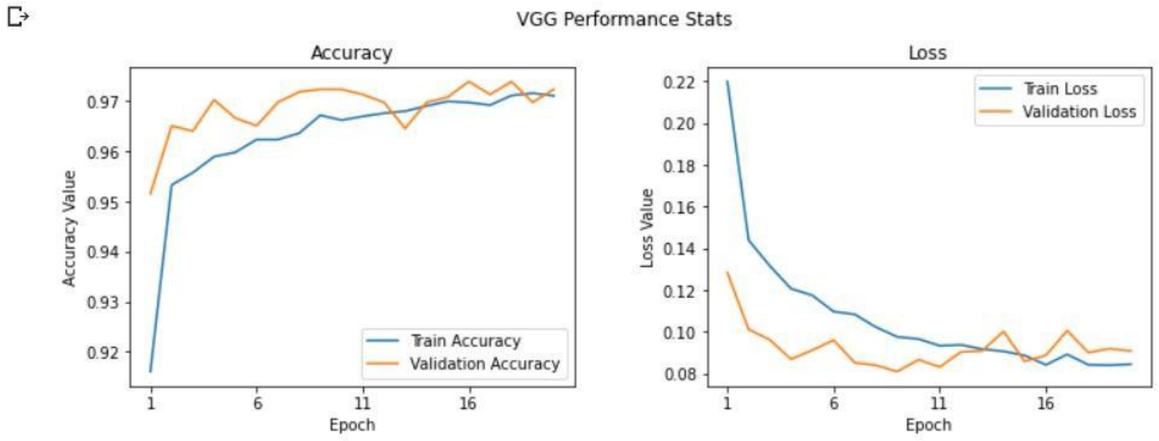
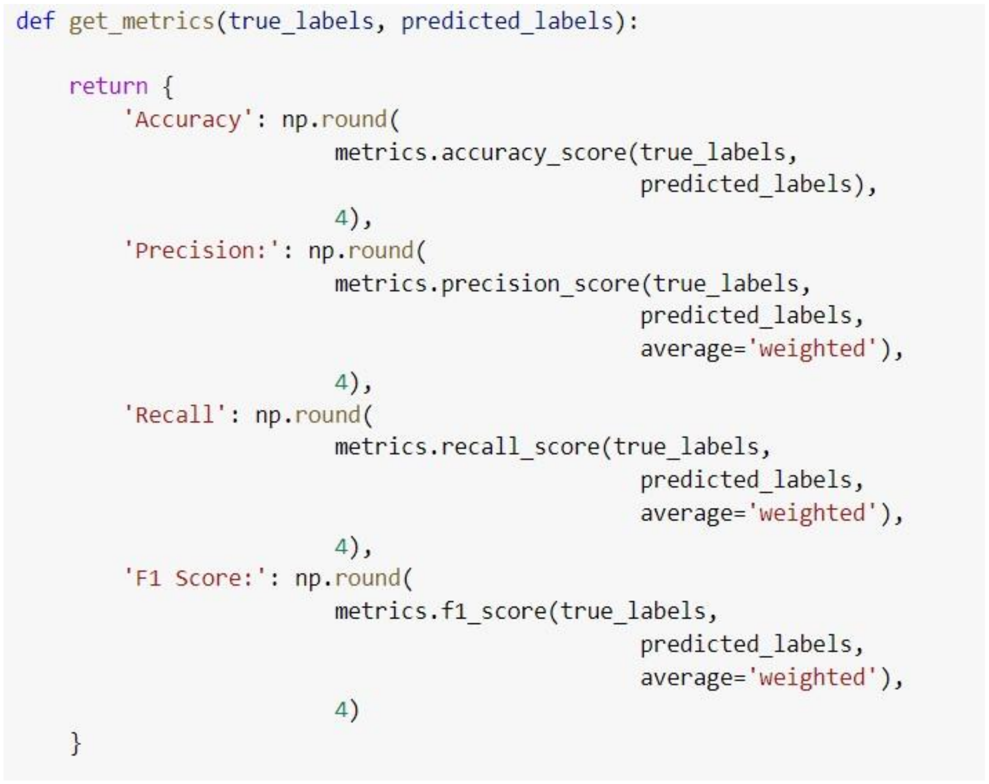


Figure 9.0 : VGG model training accuracy, validation accuracy and validation loss

3.1.2 MODEL TESTING PROCESS

After training and validating these two models the next step was to test them with actual testing data and perceive their prediction accuracies. Having built two models a basic CNN model and a pretrained VGG 19 model. These models will be examined using the testing data that was not used during training nor validation. Evaluation metrics used are Accuracy, F1, Precision and Recall.

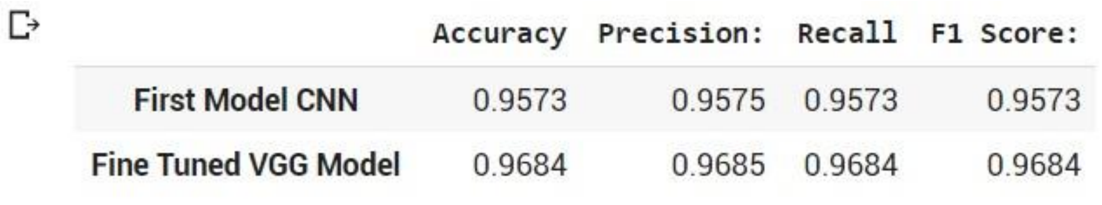


```
def get_metrics(true_labels, predicted_labels):

    return {
        'Accuracy': np.round(
                        metrics.accuracy_score(true_labels,
                                               predicted_labels),
                        4),
        'Precision:': np.round(
                        metrics.precision_score(true_labels,
                                               predicted_labels,
                                               average='weighted'),
                        4),
        'Recall': np.round(
                        metrics.recall_score(true_labels,
                                               predicted_labels,
                                               average='weighted'),
                        4),
        'F1 Score:': np.round(
                        metrics.f1_score(true_labels,
                                               predicted_labels,
                                               average='weighted'),
                        4)
    }
```

Figure 10 : Functions that calculates the four evaluation metrics.

MODEL RESULTS

Results from the testing phase showed that the basic CNN model we made had an impressive F1 score of 95.7% showing that the model is able to accurately classify parasitic and non-parasitic cells correctly. The VGG model proposed outperformed the basic CNN model used at first with a 96.8% F1 accuracy. Thus, putting a stamp that the techniques applied to the VGG model enhanced its learning and accuracy rates.



	Accuracy	Precision:	Recall	F1 Score:
First Model CNN	0.9573	0.9575	0.9573	0.9573
Fine Tuned VGG Model	0.9684	0.9685	0.9684	0.9684

Figure 11 : Test accuracies from both models.

VALIDITY OF OUR EXPERIMENTS

The steps taken to craft the proposed prototype follows good science practice based on the fact both these models were exposed to the same dataset to avoid any form of biasness. Also, the technologies used to run these models are widely used since Google Colab is a free tool to the public hence this experiment is reproduceable. Moreover, the evaluation metrics used to evaluate the veracity of the models are standard metrics recommended by machine learning experts. Thus, this experiment justified and can be tested due to clear specifications of the model development processes provided.

CONCLUSION

This paper has discussed a journey of steps and directions on how to build a proposed machine learning prototype that detects malaria in red blood cells using convolutional neural networks. The model was inspired by the comprehensive knowledge gained from routine tasks in the module workspace, also an extensive research that showed the negative impact of malaria in society and the need to overcome its effects using Ai techniques. The realization is that many experts are toiling to find reliable models that scale globally, and all are following an iterative process to improve their models.

The model proposed initially was a basic CNN model that comprised of a validation accuracy of 96.2% but was not satisfactory since the model was overfitting looking at the 99.8% training accuracy attained. This was an unexpected problem as the expectation was the model will do outstandingly well due to the regularization techniques and dropouts applied to it, but this proved futile. This challenge was a total downfall since the model benchmarked on also faced overfitting therefore this meant the proposed techniques and methodologies proposed in this paper were all in vain, as the efforts applied did not improve the flaws of the experiment carried out by Laxmi.

To overcome this problem of overfitting, in the module workspace there was a task that discussed CNN models with a link to an article online. This article was the drive that led to the discovery of two techniques to enhance the accuracy levels of the model – transfer learning and image augmentation. After implementing these two techniques the final VGG model crafted outperformed the CNN model initially made with a 96.8% F1 accuracy and was not overfitting.

This challenge aided in learning new relevant concepts and discoveries on techniques to fine tune machine learning models. The other challenge experienced was that the training process took more than a month to execute each day the model would fail to train due to idle timeouts and small RAM memory allocated to users with free accounts on Google Colab. To overcome this issue an upgrade to a paid plan was done thus a powerful runtime using TPU accelerators and high RAM were provided. This upgrade aided in training the models in less than an hour after a month struggle. Building this model was not easy at all but the lack of knowledge and the intimidating complexity of this prototype, paved way to the ability to research and find knowledge from what others are doing and implementing the knowledge gained to solve the problems experienced during the development process.

FUTURE WORKS

The extended ideas to make this model more scalable is to connect the model to a front-end dashboard using React js, by connecting the model through an API to the dashboard. Users will then be able to upload images of red blood cells from their phones or laptops and view image transforms on the image as the model analyses the cells and gives out classification results.

REFERENCE LIST

Brownlee, J., 2017. ‘A gentle introduction to transfer learning for deep learning’. *Machine Learning Mastery*.

CDC (2020). *Coronavirus Disease 2019 (COVID-19)*. [online] Centers for Disease Control and Prevention. Available at: https://www.cdc.gov/coronavirus/2019-ncov/global-covid-19/maintain-essential-servicesmalaria.html.

Dong, Y., Jiang, Z., Shen, H., Pan, W.D., Williams, L.A., Reddy, V.V., Benjamin, W.H. and Bryan, A.W., 2017, February. ‘Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells’. In 2017 *IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)* (pp. 101-104). IEEE.

Jordan, J. (2018). *Organizing machine learning projects: project management guidelines.* [online] Jeremy Jordan. Available at: https://www.jeremyjordan.me/ml-projects-guide/.

Masud, M., Alhumyani, H., Alshamrani, S.S., Cheikhrouhou, O., Ibrahim, S., Muhammad, G., Hossain, M.S. and Shorfuzzaman, M., 2020. ‘Leveraging deep learning techniques for malaria parasite detection using mobile application’. *Wireless Communications and Mobile Computing*, *2020*.

Perez, L. and Wang, J., 2017. ‘The effectiveness of data augmentation in image classification using deep learning’. *arXiv preprint arXiv:1712.04621*.

Shamsaldin, A.S., Fattah, P., Rashid, T.A. and Al-Salihi, N.K., 2019. ‘A Study of The Convolutional Neural Networks Applications’. *UKH Journal of Science and Engineering*, *3*(2), pp.31-40.

World Health Organization, (2020). World malaria report 2019. Available at:
https://www.who.int/malaria/publications/world-malaria-report-2019/en/ (Accessed 14 October 2020).