DETECTING MALARIA IN RED BLOOD CELLS USING MACHINE LEARNING

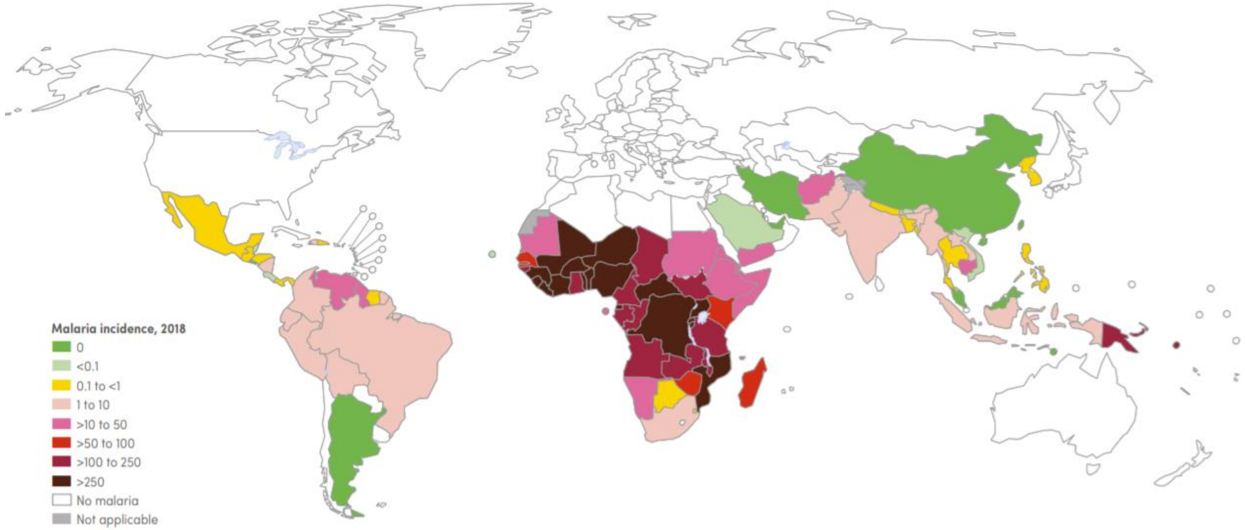
KIERNAN HARDING

MaI.ARIA
IDENTIFICATION MADE SIMPLE

University of Essex

THE PROBLEM

- Malaria has a major impact on global health
- 2019: estimated 229 million cases worldwide & 409,000 deaths [1]
- Only 5 countries account for more than 50% of all cases [1]
- Nigeria accounts for 27% of all cases [2] and has 34% of the world average GDP (per Capita) [3]
- Poor quality testing in less developed countries



Map of malaria case incidence rate (cases per 1000 population at risk) by country (2018) [2]

WHY TEST USING AI?

- Yearly, hundreds of millions of blood films are examined by a trained microscopist
- Microscopy involves manual counting of parasites in red blood cells
  - Timely
  - Costly
- Microscopists in less economically developed areas have poor-quality control settings and little resources
  - RDT test accuracy (among under 5's in 2015) is 79% [4]
  - Testing worrying low in children – only 30% (e.g. Nigeria) [2]
- Lead to incorrect diagnosis
  - False positives and false negatives
- Faster testing
- Reduced workload
- More accurate

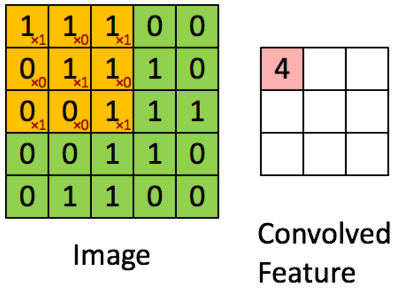
SUMMARISED AIMS & OBJECTIVES

- **Design and implement a classification tool that identifies malaria parasites in pre-segmented red blood cells**

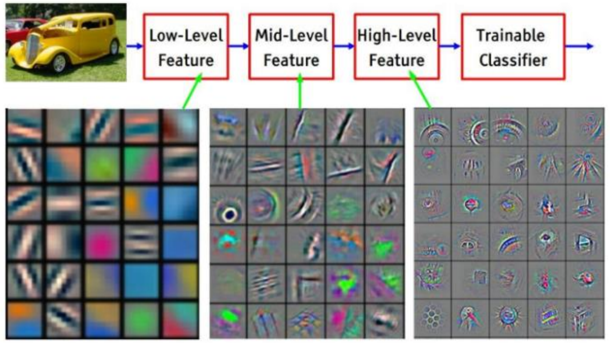
    - Design a Convolutional Neural Network using reputable Python ML libraries
    - Through iterative testing, improve CNN model performance (accuracy) using ML techniques
    - When testing the application on unseen samples, the model should achieve a higher test accuracy than the RDT's diagnostic accuracy of 79% (among under 5's in Nigeria, 2015 [4])
    - Create a classification tool that is accessible via almost any device, enabling its use in less developed counties
    - The application should be simplistic to allow easy use for individuals with limited medical knowledge

WHAT IS A CNN?

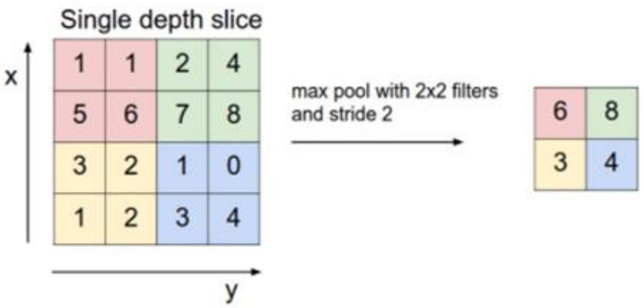
- Short for 'Convolutional Neural Network'
- Class of deep learning neural networks
- Learns similarly to a child – inspired by the human brain
- Trained through backpropagation:
  - Forward pass – pass input data through network as normal as caches values
  - Backwards pass – go back through network and alter loss (similar to telling a child they're incorrect, which they then learn from)
- Different types of layers:
  - Convolutional – identifies features in images, such as straight lines and curves, aids learning process (becoming less abstract over time)
    - Activation function (e.g. ReLU) – aids the network to learn complex patterns in the data – increase non-linearity (ReLU: returns 0 if negative and value x if positive)
  - Pooling (e.g. max pooling) – reduces sample size and thus speeds up processing
  - Fully connected – essentially the output layer (provides probabilistic values)



Convolution [5]



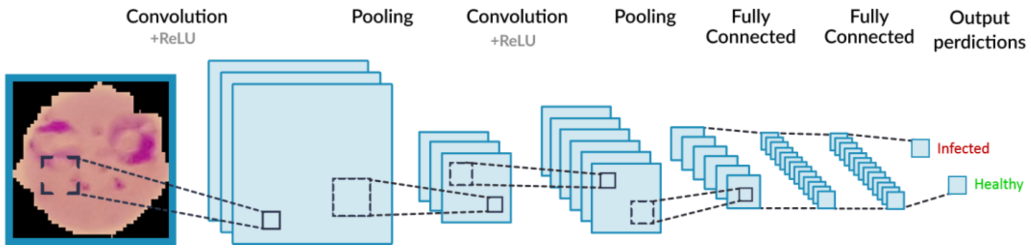
Convolved feature maps [6]



Max pooling [7]

CREATING A CNN MODEL

- Splitting the dataset into training, validation and testing (~ 80/10/10)
- Preparing the dataset for training – rescaling
- Created a convolutional neural network model in Python using TensorFlow and Keras
  - 3 main layers and an output layer (the fully connected layer)
  - These layers include: convolutional, activation (ReLU), pooling and fully connected
- Greyscale, colour…



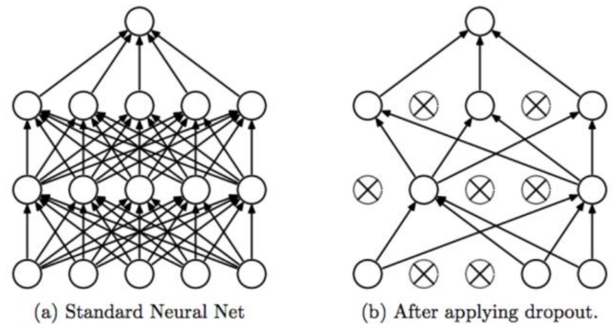
Example CNN model architecture

Final model architecture

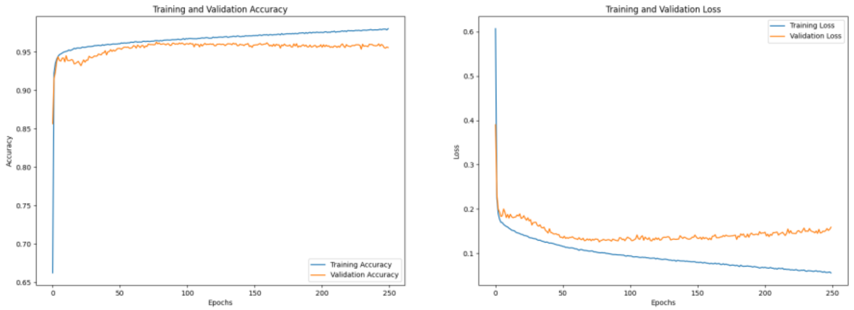
```
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 100, 100, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 100, 100, 32)      896
_________________________________________________________________
activation (Activation)      (None, 100, 100, 32)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 50, 50, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 50, 50, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 50, 50, 32)        9248
_________________________________________________________________
activation_1 (Activation)    (None, 50, 50, 32)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 25, 25, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25, 25, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 25, 25, 32)        9248
_________________________________________________________________
activation_2 (Activation)    (None, 25, 25, 32)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 12, 12, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 128)               589952
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 258
=================================================================
Total params: 609,602
Trainable params: 609,602
Non-trainable params: 0
```

IMPROVING PERFORMANCE

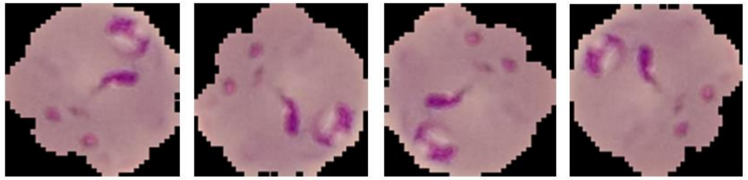
- Iteratively improved performance by evaluating a current model's validation accuracy
- Machine learning techniques to reduce *overfitting* and improve model performance:
    - Data Visualisation
        - Different perspectives, easy comparisons
    - Data Augmentation
        - Rotate, flipping, translation, scaling…
    - Regularisation
        - Applying dropout
    - Optimizer Adjustment
        - Adam, SGD, RMSprop
    - Feature Map Visualisation
        - Identify key areas that are detected



Dropout representation [8]

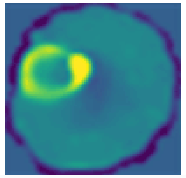


Final CNN model training graph



Original Rotated 90° Rotated 180° Rotated 270°

Data augmentation



Feature map sample

EVALUATION/MILESTONE RESULTS

- I evaluated the performance by calculating the model accuracy
- I iteratively improved on my model performance
- The model achieves a higher test accuracy than the RDT's diagnostic accuracy of 79% (among under 5's in Nigeria, 2015 [4])
- NIH only achieved ~2% higher accuracy in a similar project

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

Accuracy equation [9]

Model	Validation Accuracy	Test Accuracy
Greyscale (Interim)	94.2%	71.4%
Initial Colour	93.6%	93.3%
Colour Model (Final)	96.2%	95.6%

UI – WEB APPLICATION

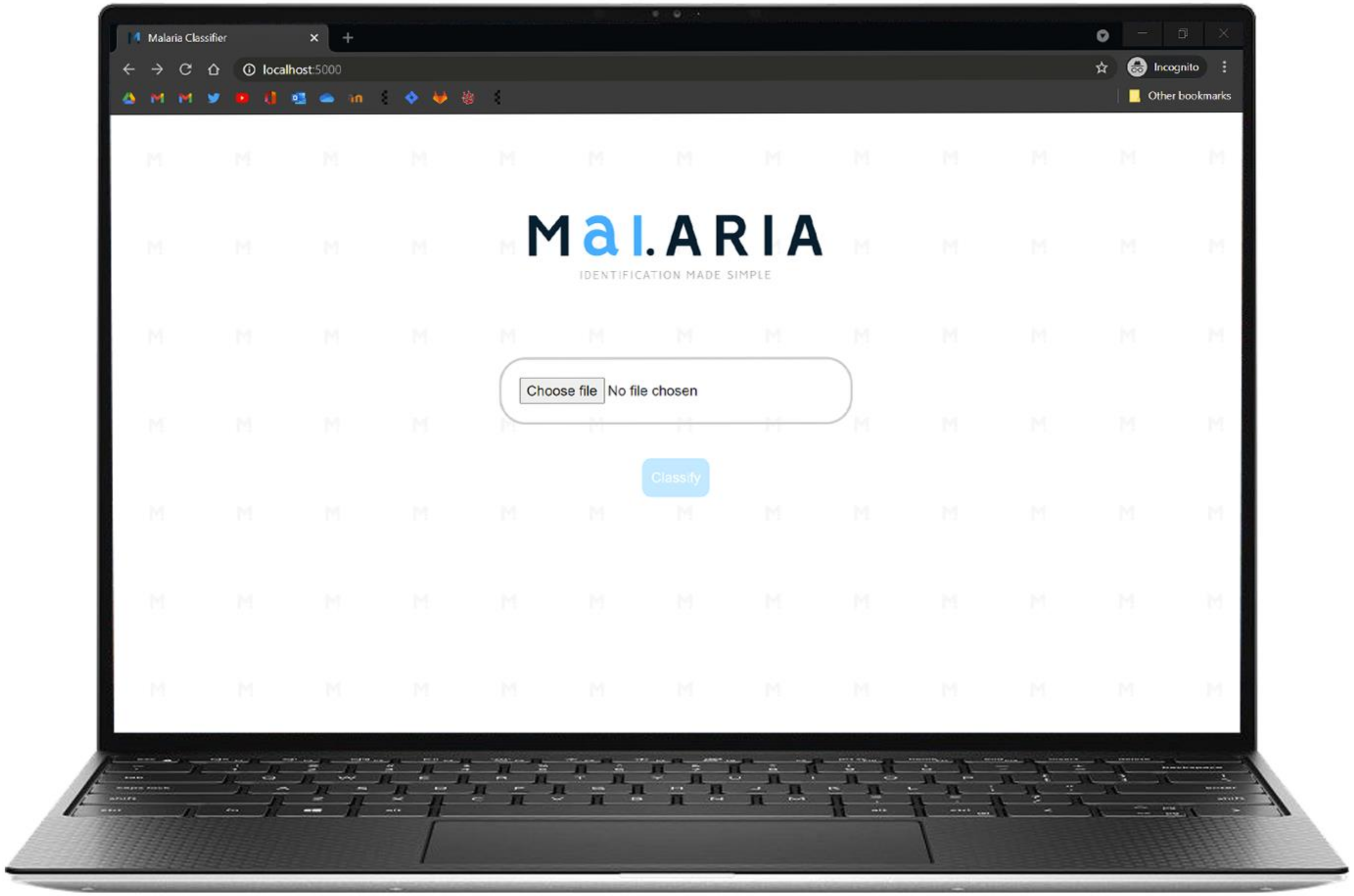
- Use of web development to increase the applications accessibility in remote areas (localhost)
- No specific requirements (e.g. android application = limits reach)
- Uses the Flask library to implement the Keras CNN model into a web application
- Bootstrap styling…
- Simple to use and navigate
- Takes an image input of a blood sample
- Classifies and outputs whether the sample is infected

DEMO

- A web application that includes a back-end machine learning classifying model
- Can be used on any device with an internet browser
- Live examples of infected blood cells being classified
- Live examples of healthy blood cells being classified

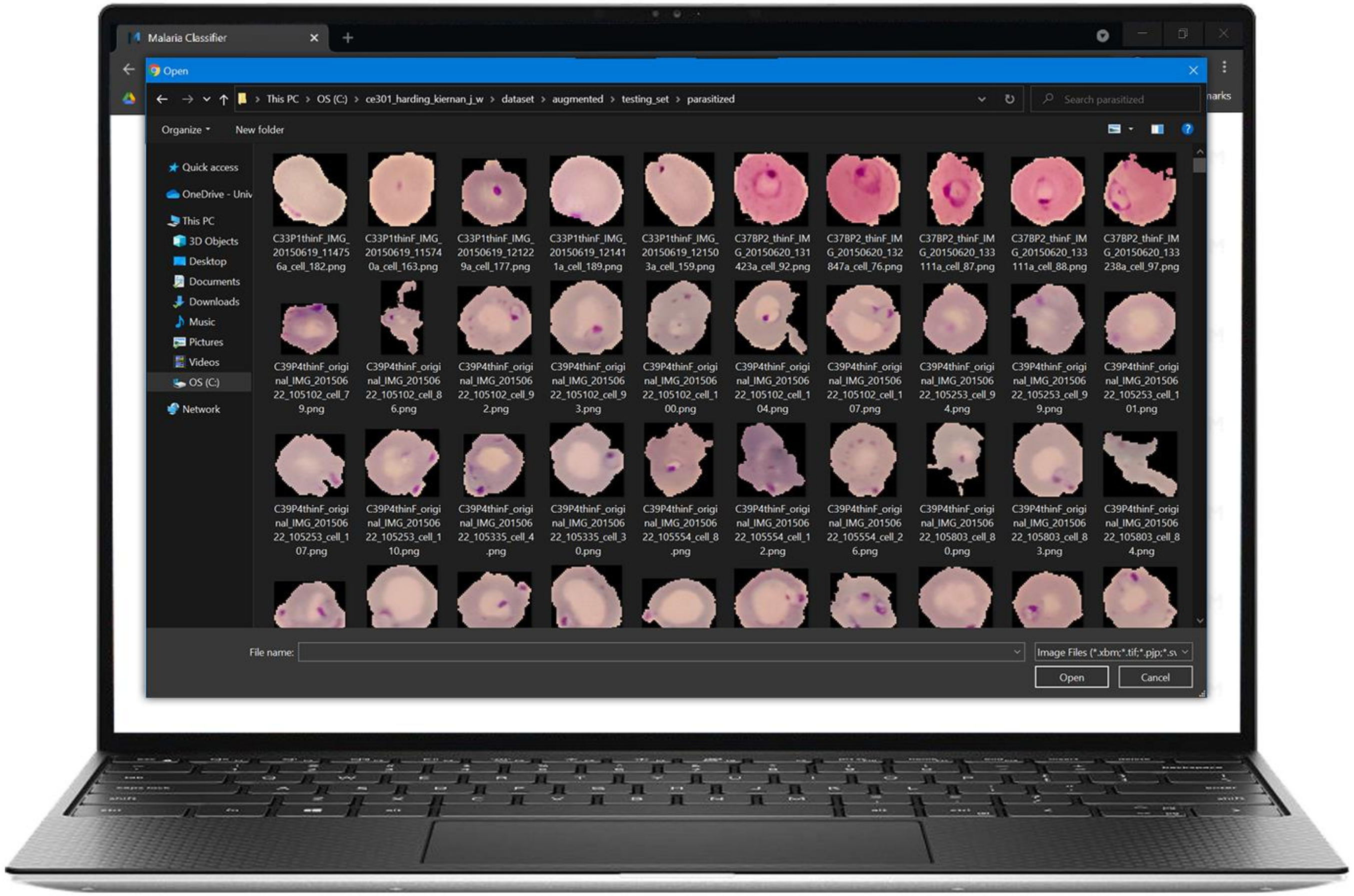
DEMO

1. The home page



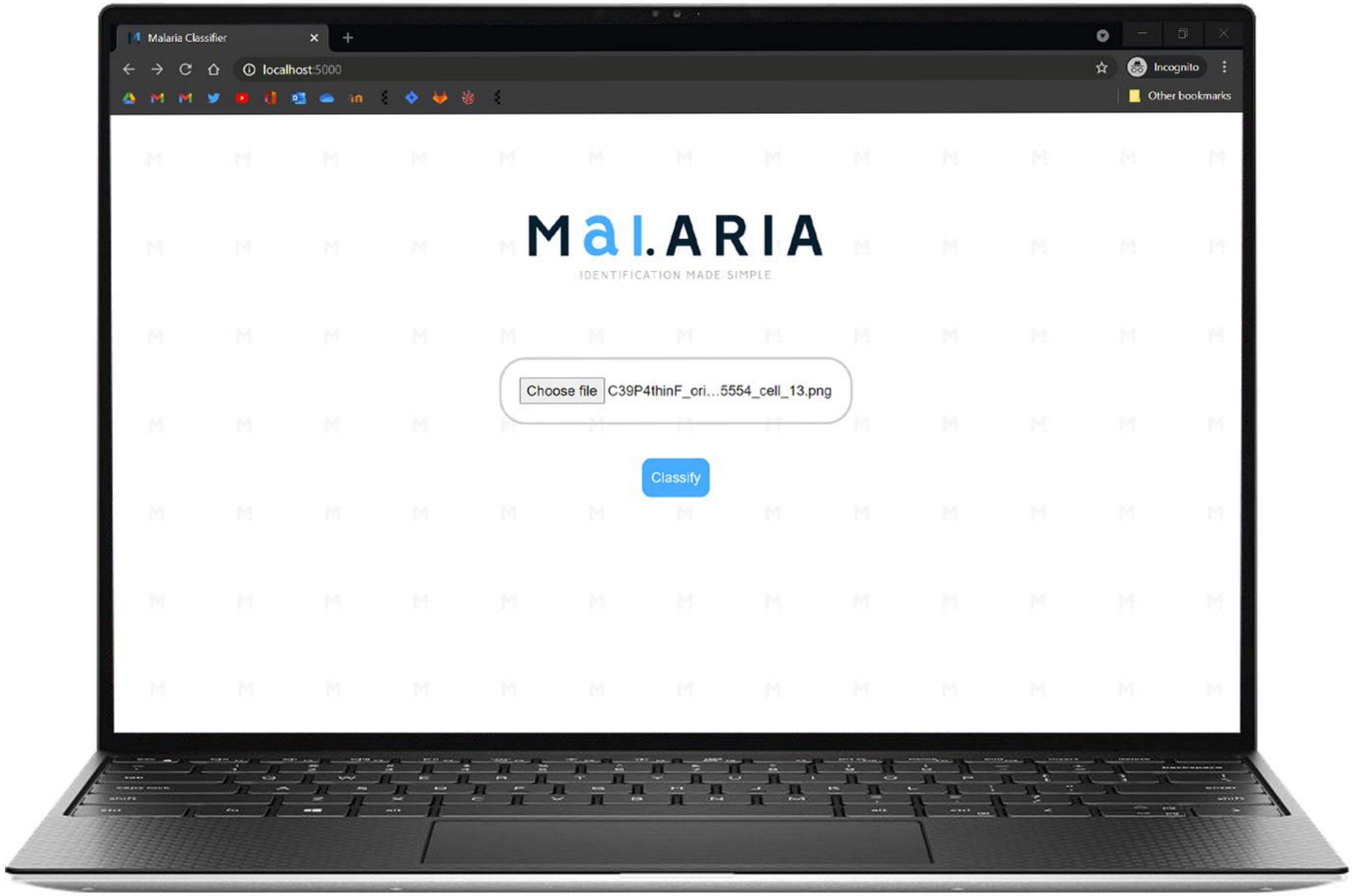
DEMO

2. Choosing the pre-segmented red blood cell to classify



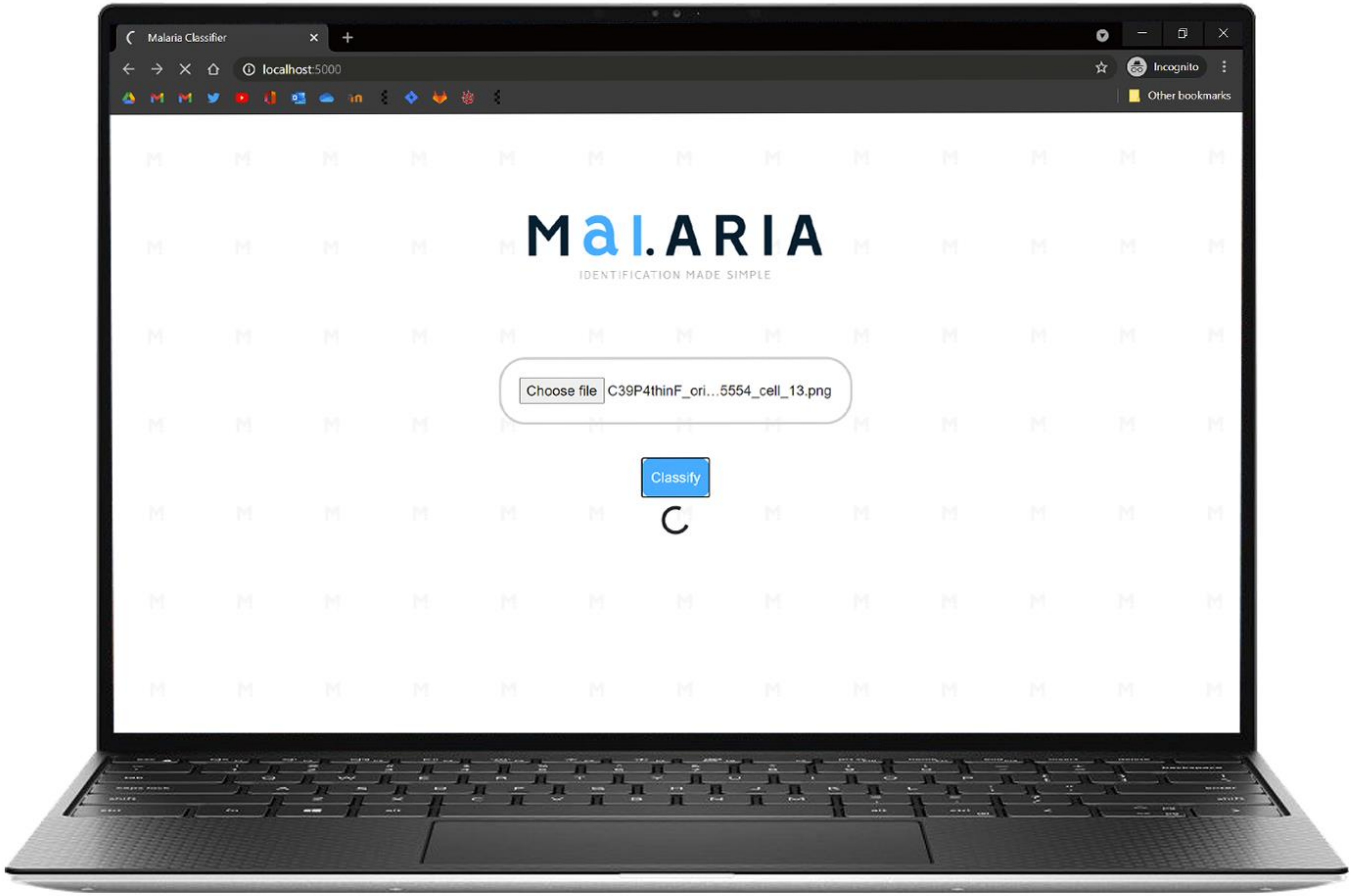
DEMO

3. The sample has been selected



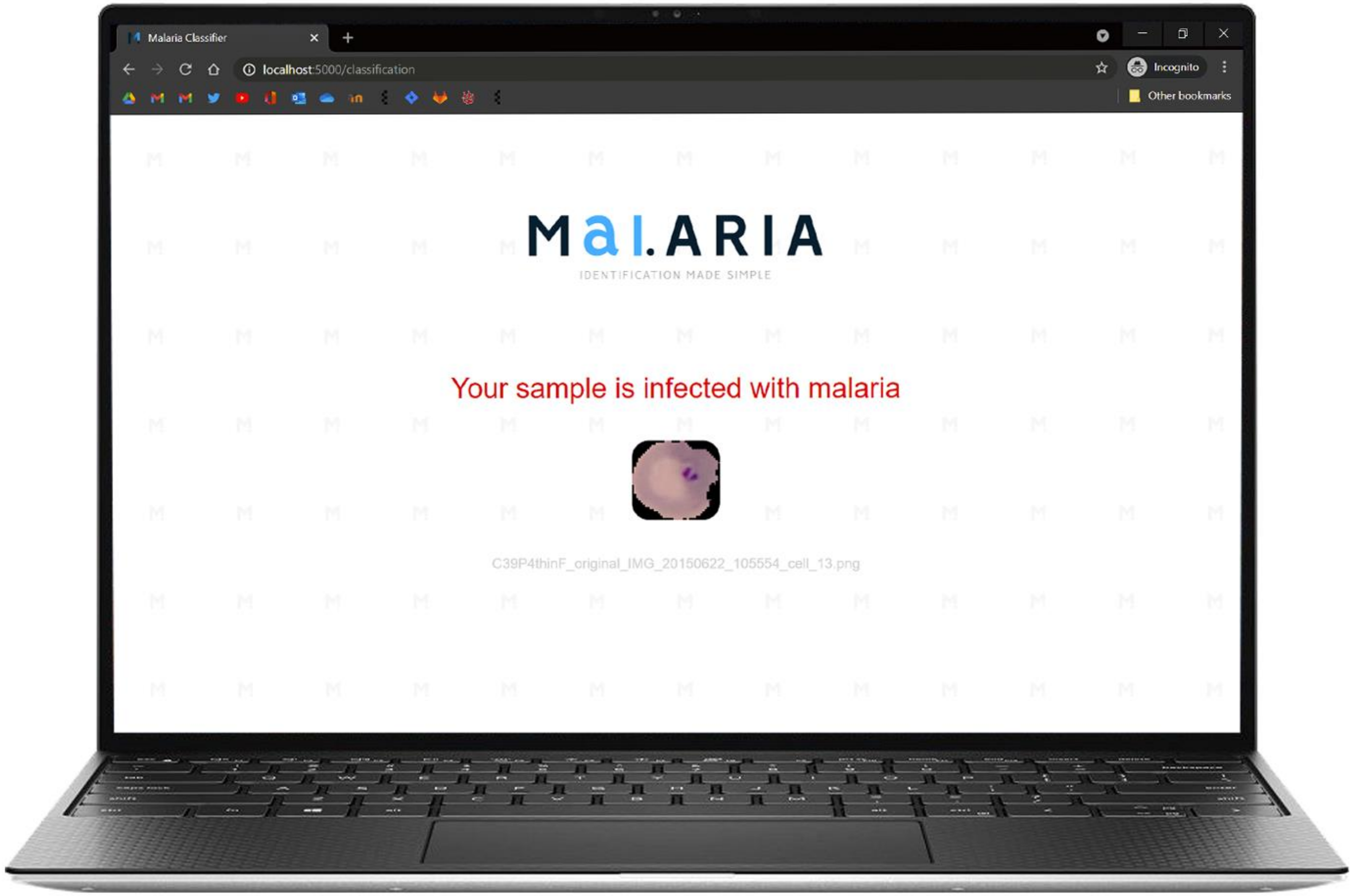
DEMO

4. The ‘classify’ button has been clicked and the sample is being classified



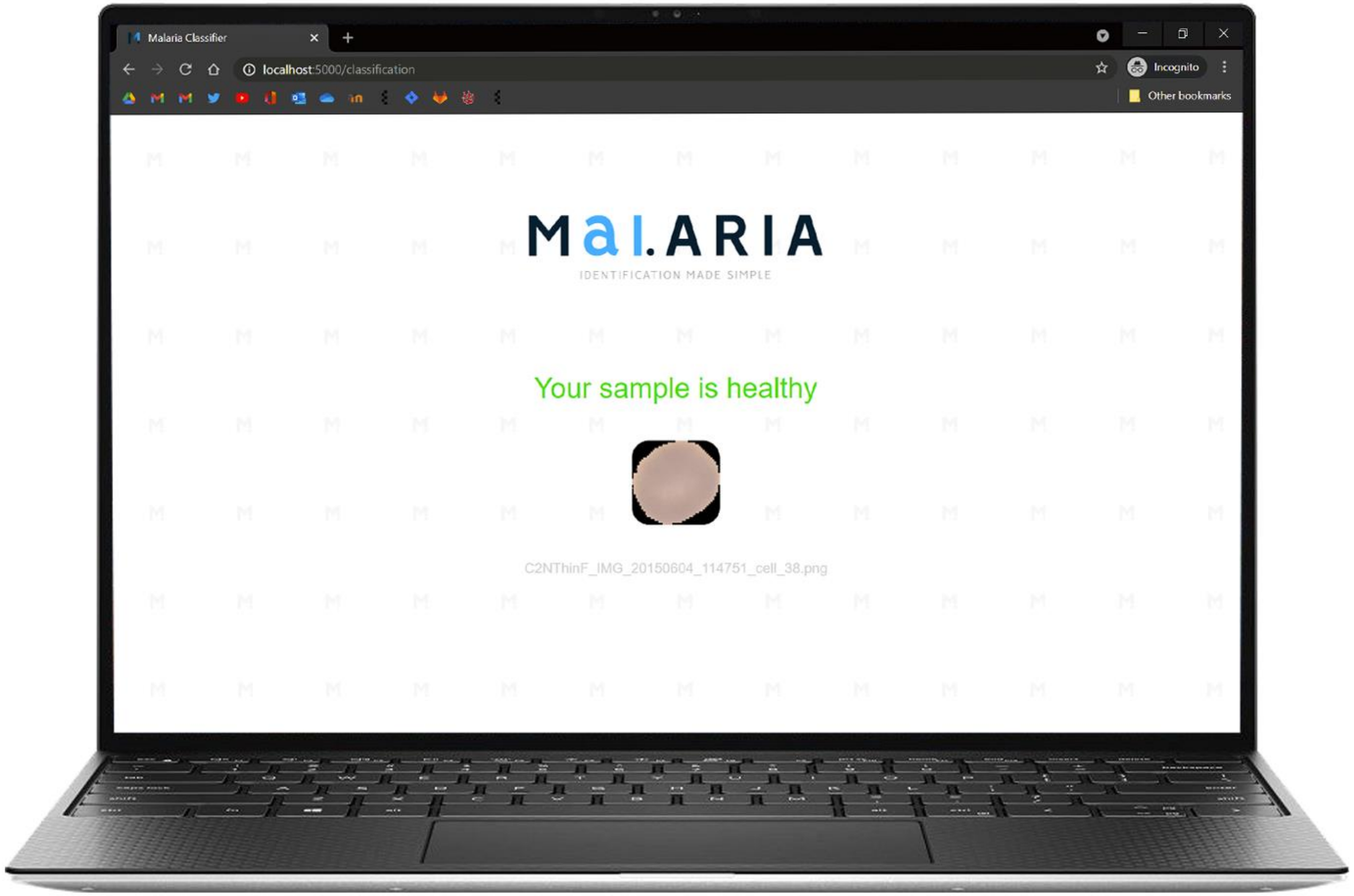
DEMO

5a. The classification output for an infected sample



DEMO

5b. The classification output for a healthy sample



DID I MEET MY AIMS & OBJECTIVES?

- Design and implement a classification tool that identifies malaria parasites in pre-segmented red blood cells

  - ✓ Design a Convolutional Neural Network using reputable Python ML libraries
  - ✓ Through iterative testing, improve CNN model performance (accuracy) using ML techniques
  - ✓ When testing the application on unseen samples, the model should achieve a higher test accuracy than the RDT's diagnostic accuracy of 79% (among under 5's in Nigeria, 2015 [4])
  - ✓ Create a classification tool that is accessible via almost any device, enabling its use in less developed counties
  - ✓ The application should be simplistic to allow easy use for individuals with limited medical knowledge

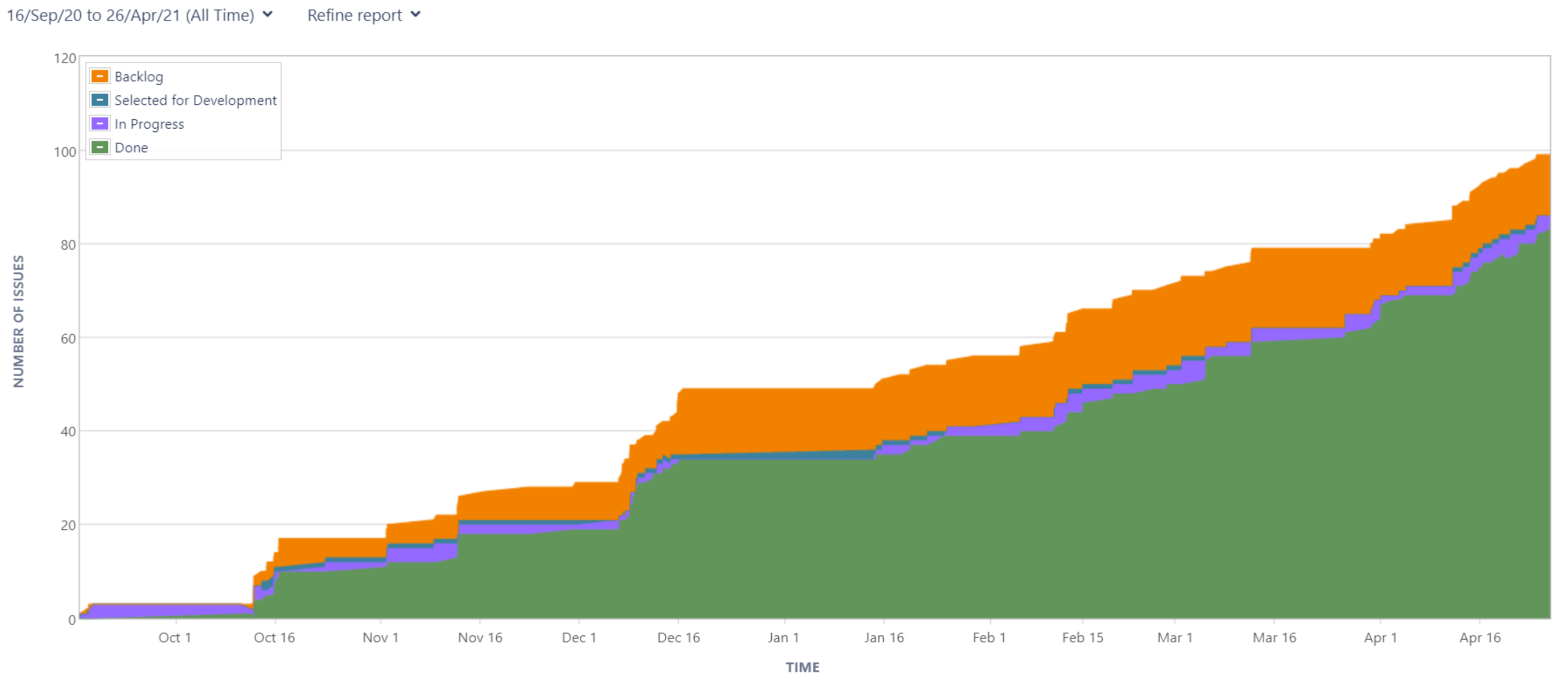
INTERIM → "TIMELINE OF FUTURE OBJECTIVES"

Improve my convolutional neural network model through research and structured testing *(e.g. add colour)*	Implement bootstrap into my website to improve usability across all devices	Improve the visual styling of the user interface	Slack for any issues encountered before the submissions
8/2/21	22/2/21	8/3/21	17/3/21

Now → 17/3/21 – Open Day Submissions

USE OF JIRA

- Consistent use of Jira
- Use of tasks and sub-tasks to add detail
- Comments to document updates/plan future tasks

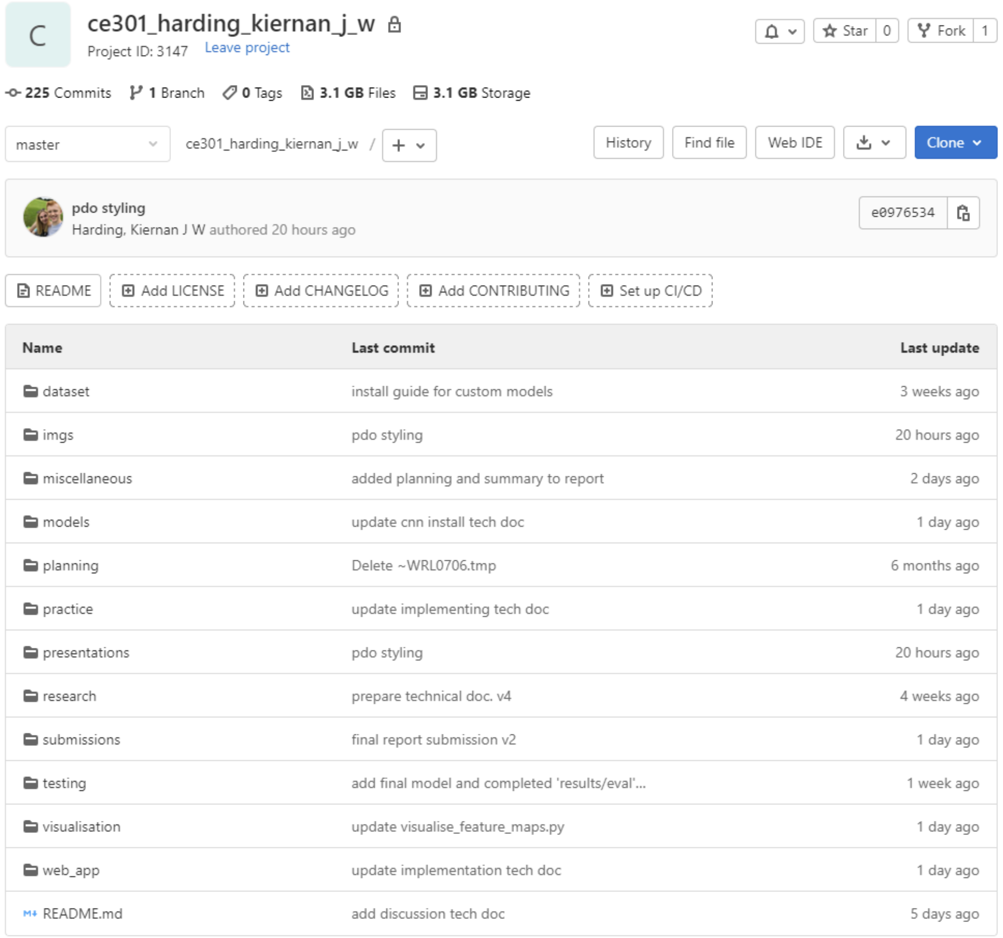


A cumulative flow diagram of issues from Jira (25/4/2021)

USE OF GIT

- Consistent use of Git
- Regular commits to Git for:
    - Significant changes
    - Keep important work backed-up!
    - Small required alterations
- Short, meaningful commit messages

- Use of .md files to keep a diary of milestone achievements and each CNN model's performance



A snapshot of my Git repository (1/5/2021)

MAIN ISSUES ENCOUNTERED

- Installation of the required local environment – e.g. libraries such as TensorFlow
- Issues with array types between libraries – Occurred whilst creating a working convolutional neural network using tutorials
- Trying to save, use and classify the input image whilst designing the web application

HOW TO IMPROVE

- Research further ML techniques to improve the model's performance (increase accuracy)
- Significantly altering the CNN architecture and more of its parameters
- Investigate segmenting red blood cells from thin blood smears, allowing the whole classification process to be automated

THANK YOU, I HOPE YOU ENJOYED!

ANY QUESTIONS?

REFERENCES

[1] WHO, "World Malaria Report 2020," 30 November 2020. [Online]. Available: https://www.who.int/publications/i/item/9789240015791.

[2] WHO, "World Malaria Report 2019," 4 December 2019. [Online]. Available: https://www.who.int/publications/i/item/9789241565721.

[3] Worldometer, "GDP per Capita," 2017. [Online]. Available: https://www.worldometers.info/gdp/gdp-per-capita/.

[4] A. F. Fagbamigbe, "On the discriminatory and predictive accuracy of the RDT against the microscopy in the diagnosis of malaria among under-five children in Nigeria," Malaria journal, vol. 18, no. 1, pp. 1–12, 21 February 2019.

[5] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks," 15 December 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[6] R. Gandhi, "Build Your Own Convolution Neural Network in 5 mins," 18 May 2018. [Online]. Available: https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f.

[7] Stanford, "CS231n Convolutional Neural Networks for Visual Recognition," [Online]. Available: https://cs231n.github.io/convolutional-networks/. [Accessed 1 April 2021].

[8] A. Budhiraja, "Dropout in (Deep) Machine learning," 15 December 2016. [Online]. Available: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5.

[9] E. Dauria, "Accuracy, Recall & Precision," 8 December 2019. [Online]. Available: https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d.