Capstone Project
Malaria Detection

Final Report

1

Executive Summary

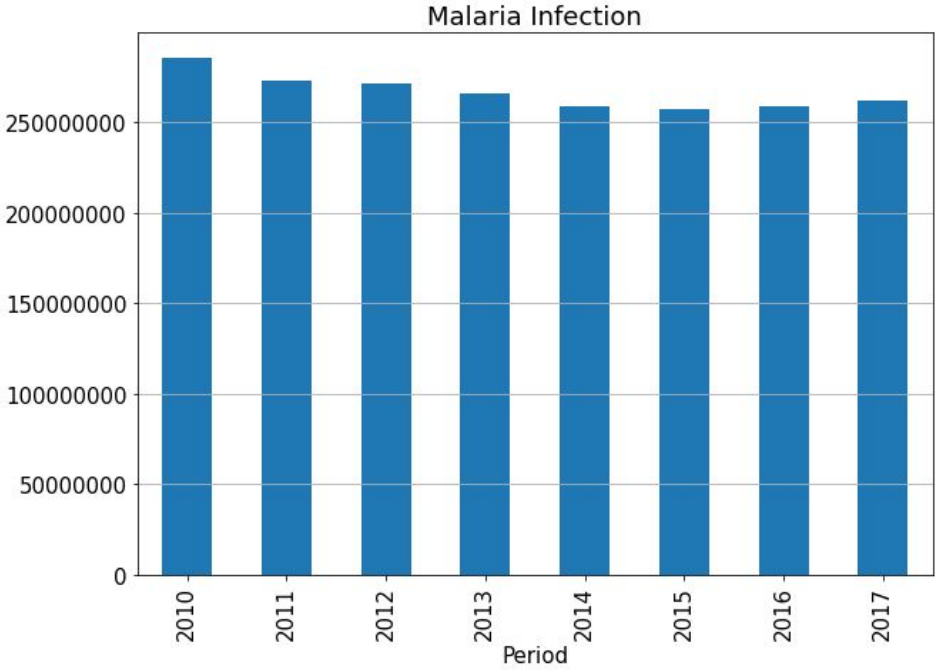
MALARIA DETECTION

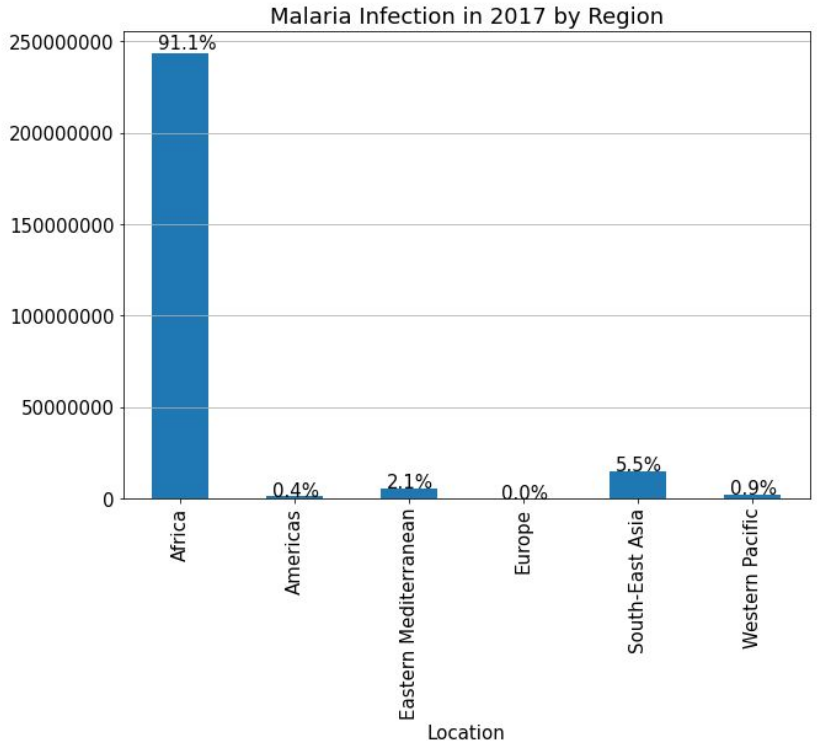
Defining the Problem

MALARIA

- Is caused by **Plasmodium parasites** transmitted to humans through the bites of infected female Anopheles mosquitoes.

A Big Issue


Malaria Infection
250000000
200000000
150000000
100000000
50000000
0
2010
2011
2012
2013
2014
2015
2016
2017
Period


Malaria Infection in 2017 by Region
250000000
200000000
150000000
100000000
50000000
0
91.1%
0.4%
2.1%
0.0%
5.5%
0.9%
Africa
Americas
Eastern Mediterranean
Europe
South-East Asia
Western Pacific
Location

229,000,000

Malaria cases

400,000

Malaria related deaths

in

2019

Working at **60 cases/hour** a doctor will clear only

120,000 cases/yr

he will need to work

1,900 years

to clear all the cases in

In

2019

alone

A **single** computer working at a rate of

60,000 cases/hr

can complete the task in

201 days

Saving approximately

$380 Million

and potentially

400,000

lives

2

Problem and Solution Summary

Proposed Approach

POTENTIAL TECHNIQUES

An **automated system** can help with the early and accurate detection of malaria. Applications of automated classification techniques using Machine Learning (ML) and Artificial Intelligence (AI) have consistently shown higher accuracy than manual classification. It would therefore be highly beneficial to propose a method that performs malaria detection using **Deep Learning Algorithms**.

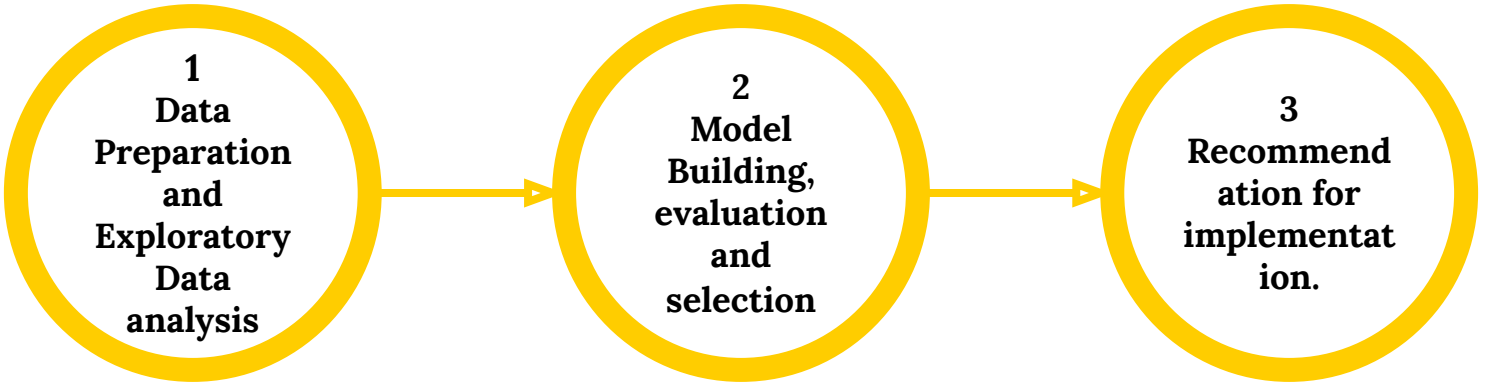
OVERALL SOLUTION DESIGN

We will be using building a **convolution neural network (CNN) model** to classify between infected and non-infected blood cell samples from the images. This is a binary classification problem so we will be using a sigmoid activation function. Our process will be **Data Preparation > Exploratory Data analysis > Model Building > Model evaluation > Model selection > Summary of potential benefits > Recommendation for implementation.**

MEASURES OF SUCCESS

We will attempt various models and **test accuracy with a loss score**. We will also use a 20% validation split for each model to see if training and validation accuracy improve together to determine if the model is overfitting.

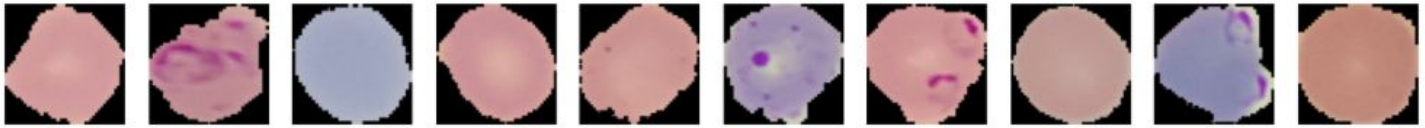
Proposed Approach



See appendix detailed slides of steps 1 & 2. We will look at the second half of steps 2 and 3 today.

Data Exploration

PRINTING OUT SOME OF THE IMAGES AND LOOKING AT THE ASSOCIATED LABELS – AND IT'S CORRECTLY LABELLED



```
label for each of the above image: [0 1 0 0 0 1 1 0 1 0]
```

- We print out a random set of images and compared with the labels and they correspond.
- Next we will take a look at the dataset and check the shape - it should be 4D arrays and should not need to be further reshaped.

```
Training set: (24958, 224, 224, 3) (24958,)
Test set: (2600, 224, 224, 3) (2600,)
```

NEXT WE CHECK TO SEE IF OUR DATA IS BALANCED BETWEEN POSITIVE AND NEGATIVE CASES AND WE SEE THAT WE HAVE AN EQUAL NUMBER OF POSITIVE AND NEGATIVE CASES

```
Positive cases in the training set :  12479
Negative cases in the training set :  12479
Positive cases in the test set :  1300
Negative cases in the test set :  1300
```

Comparison of Various Techniques

MODEL COMPARISONS

We did 4 **Convolutional Neural Network** models and saw the results progressive improved. Let's compare the models to evaluate various techniques and each model's performance. We used the Leaky Relu activation function for the hidden layers and a Sigmoid activation function for the output layer. Below we see the adjustments we made for each model.

CNN Model 1

Sequential Model
Layer 1 16 nodes
Layer 2 32 nodes
Max Pooling Layer
Flatten

Dense Layer 32 nodes
Sigmoid Activation

CNN Model2

Sequential Model
Layer 1 16 nodes
+ 20% Dropout Layer
Layer 2 32 nodes
+ 20% Dropout Layer
Max Pooling Layer
Flatten

Dense Layer 32 nodes
+50% Dropout Layer
Sigmoid Activation

CNN Model3

Sequential Model
Layer 1 16 nodes
+ 32 Node Layer
+ Max Pooling Layer
+ 25% Dropout Layer
Layer 2 32 nodes
+ 64 Node Layer
+ 25% Dropout Layer
Max Pooling Layer
Flatten

Dense Layer 32 nodes
50% Dropout Layer
Sigmoid Activation

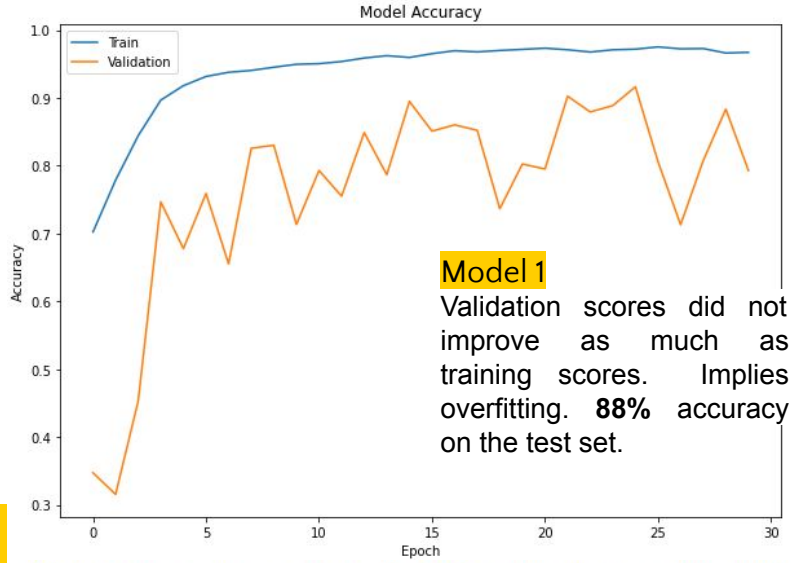
CNN Model4

Sequential Model
Layer 1 16 nodes
+ 20% Dropout Layer
32 Node Layer
+ 64 nodes
+ 20% Dropout Layer
Max Pooling Layer
Flatten

Dense Layer 32 nodes
+50% Dropout Layer
Sigmoid Activation

Comparison of Various Techniques

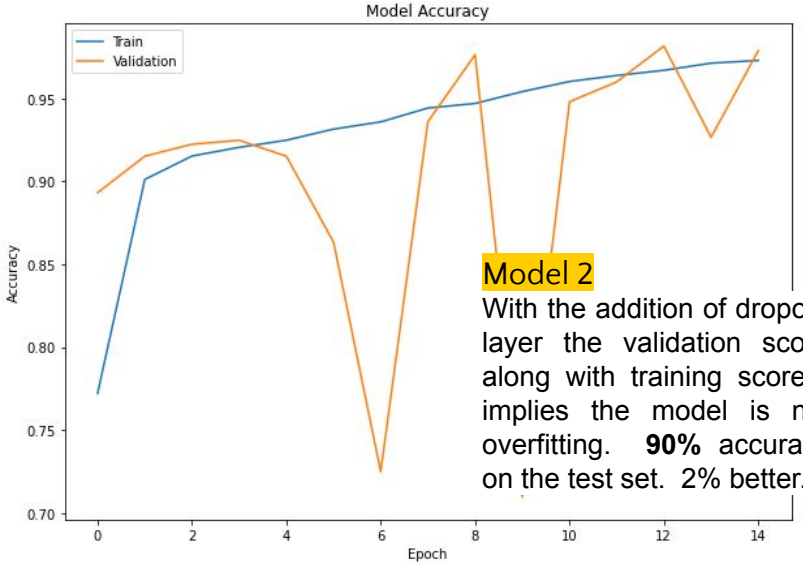
MODEL COMPARISONS



82/82 [==============================] - 2s 25ms/step - loss: 0.4659 - accuracy: 0.8819
Model loss on the test set: 0.46586668491363525
Model accuracy on the test set: 88.19%

Model 1

Validation scores did not improve as much as training scores. Implies overfitting. **88%** accuracy on the test set.



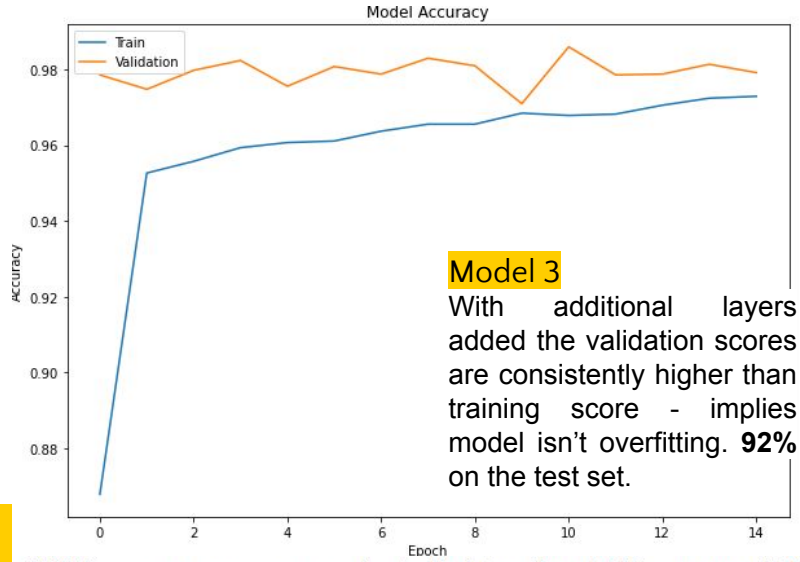
82/82 [==============================] - 2s 21ms/step - loss: 0.4370 - accuracy: 0.9058
Model loss on the test set: 0.43698763847351074
Model accuracy on the test set: 90.58%

Model 2

With the addition of dropout layer the validation score along with training score - implies the model is not overfitting. **90%** accuracy on the test set. 2% better.

Comparison of Various Techniques

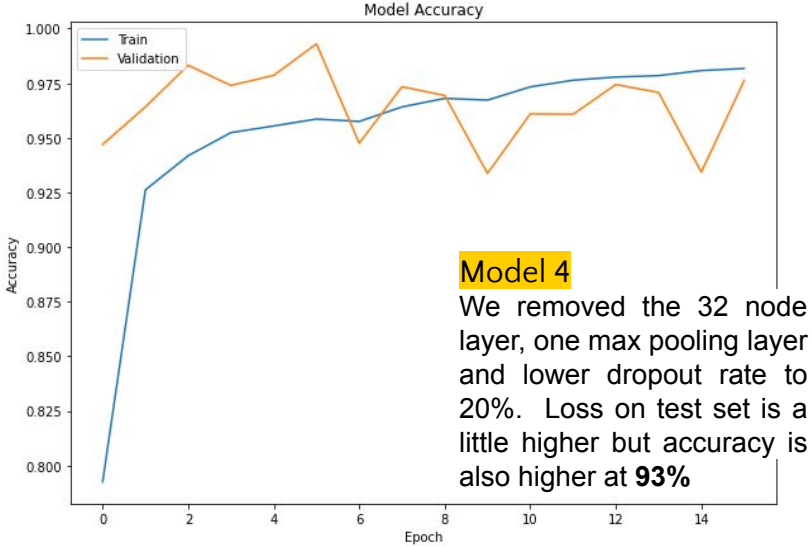
MODEL COMPARISONS



Model 3

With additional layers added the validation scores are consistently higher than training score - implies model isn't overfitting. **92%** on the test set.

```
82/82 [==============================] - 3s 33ms/step - loss: 0.2555 - accuracy: 0.9292
Model loss on the test set: 0.2554693818092346
Model accuracy on the test set:  92.92%
```



Model 4

We removed the 32 node layer, one max pooling layer and lower dropout rate to 20%. Loss on test set is a little higher but accuracy is also higher at **93%**

```
82/82 [==============================] - 5s 55ms/step - loss: 0.3608 - accuracy: 0.9327
Model loss on the test set: 0.36076441407203674
Model accuracy on the test set:  93.27%
```

Hypeparameter Tuning and Transfer Learning Models

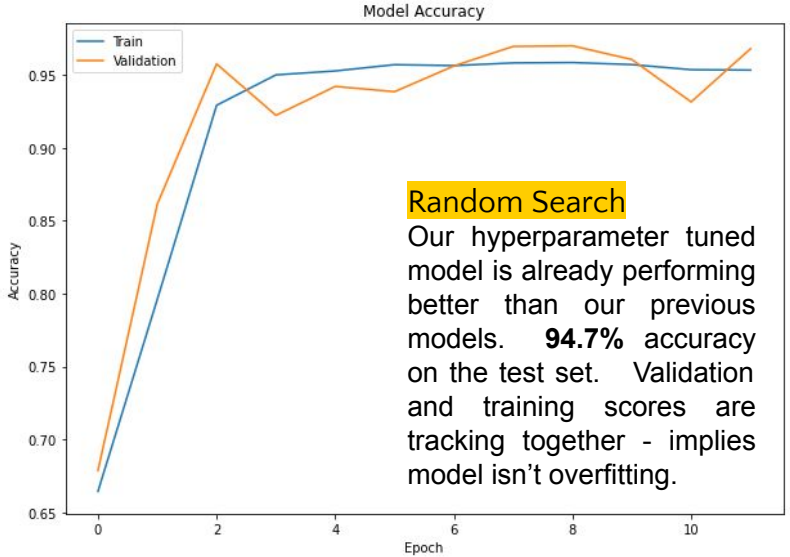
MODEL COMPARISONS

We did 2 Hyperparameter tuned **Convolutional Neural Network** models and 2 transfer learning models. The model architectures are informed by hyperparameter tuning methods and the improved iteratively based on the results. Let's compare the models to evaluate various techniques and each model's performance.

Random Search	Bayesian Optimazation	Transfer Learning	Bayesian / Transfer Learning
Sequential Model	Sequential Model	Sequential Model	Sequential Model
Layer 1 112 nodes + 30% Dropout Layer	Layer 1 128 nodes + 40% Dropout Layer	**VGG16 Model**	**VGG16 Model**
Layer 2 160 nodes + Max Pooling Layer + 30% Dropout Layer	Layer 2 160 nodes + Max Pooling Layer + 40% Dropout Layer	Flatten	Flatten
Layers 3 & 4 32 nodes + Max Pooling Layer + 30% Dropout Layer	Layers 3 & 4 32 nodes + Max Pooling Layer + 40% Dropout Layer	Dense Layer 32 nodes 30% Dropout Layer Sigmoid Activation	Dense Layer 112 nodes 20% Dropout Layer Sigmoid Activation
Flatten	Flatten		
Dense Layer 112 nodes 30% Dropout Layer Sigmoid Activation	Dense Layer 128 nodes 40% Dropout Layer Sigmoid Activation		

Hypeparameter Tuning and Transfer Learning Models

MODEL COMPARISONS

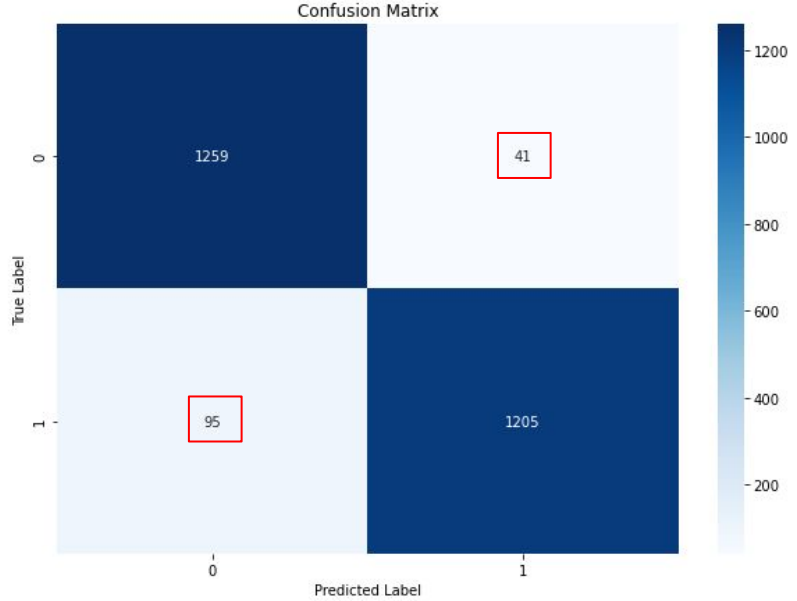


Random Search

Our hyperparameter tuned model is already performing better than our previous models. **94.7%** accuracy on the test set. Validation and training scores are tracking together - implies model isn't overfitting.

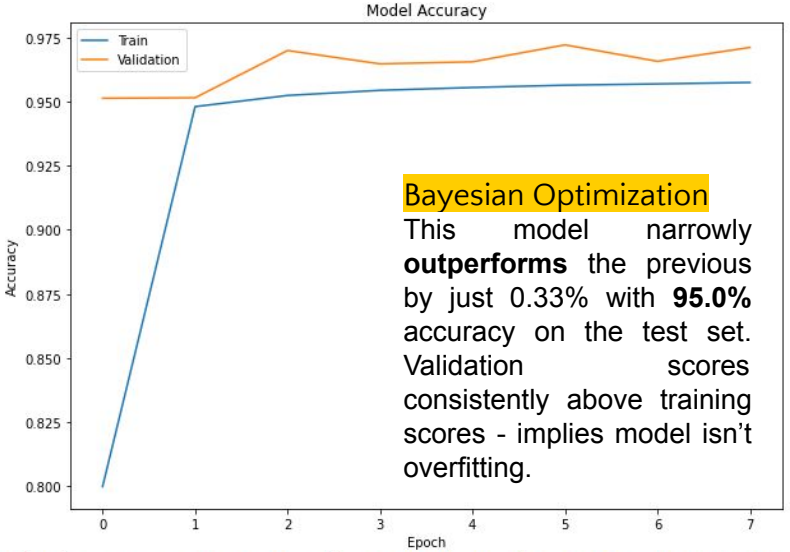
```
82/82 [==============================] - 20s 240ms/step - loss: 0.1611 - accuracy: 0.9477
Model loss on the test set: 0.16108182072639465
Model accuracy on the test set:  94.77%
```

	precision	recall	f1-score	support
0	0.93	0.97	0.95	1300
1	0.97	0.93	0.95	1300
accuracy			0.95	2600



Hypeparameter Tuning and Transfer Learning Models

MODEL COMPARISONS

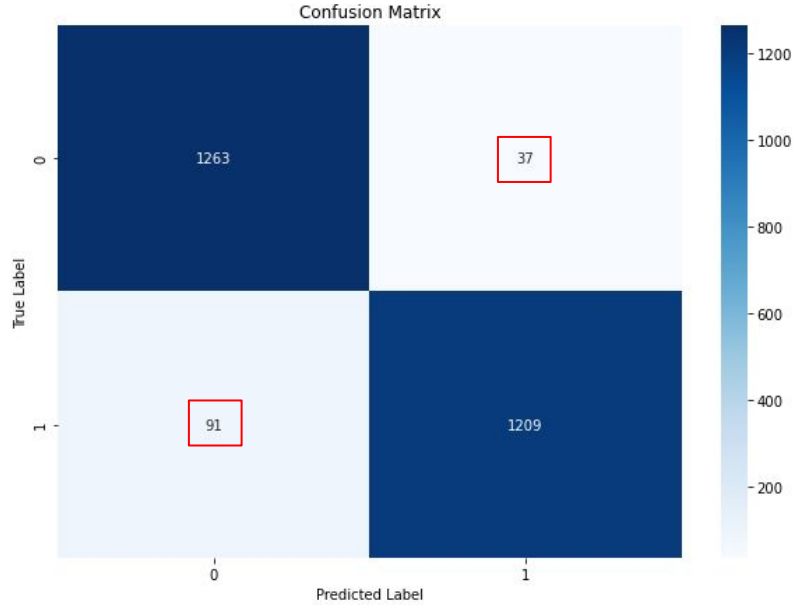


82/82 [==============================] - 21s 259ms/step - loss: 0.1612 - accuracy: 0.9508
Model loss on the test set: 0.1612130105495453
Model accuracy on the test set: 95.08%

Bayesian Optimization

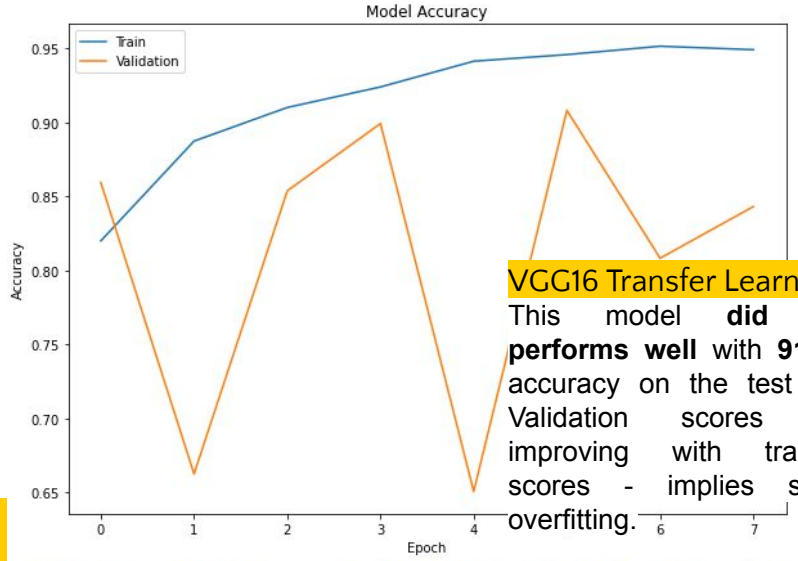
This model narrowly **outperforms** the previous by just 0.33% with **95.0%** accuracy on the test set. Validation scores consistently above training scores - implies model isn't overfitting.

	precision	recall	f1-score	support
0	0.93	0.97	0.95	1300
1	0.97	0.93	0.95	1300
accuracy			0.95	2600



Hypeparameter Tuning and Transfer Learning Models

MODEL COMPARISONS

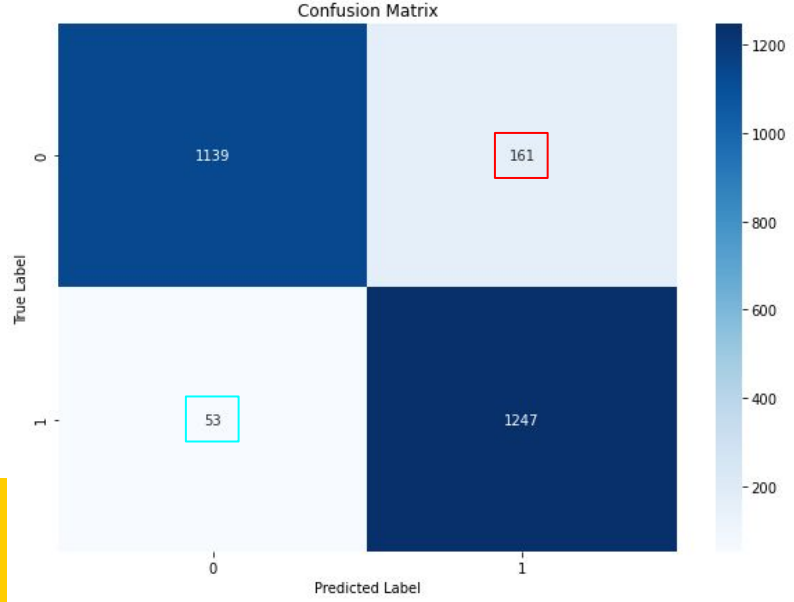


VGG16 Transfer Learning

This model **did not performs well** with **91.7%** accuracy on the test set. Validation scores not improving with training scores - implies some overfitting.

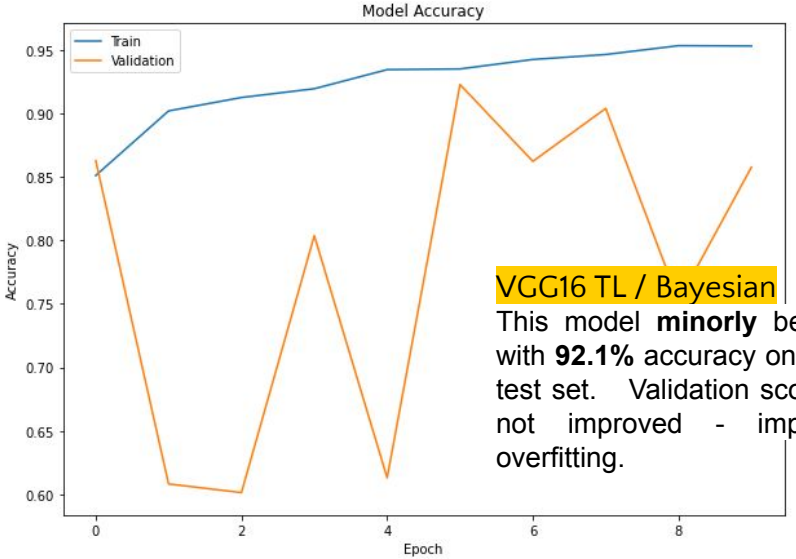
```
82/82 [==============================] - 18s 221ms/step - loss: 0.2482 - accuracy: 0.9177
Model loss on the test set: 0.2482037991285324
Model accuracy on the test set:  91.77%
```

	precision	recall	f1-score	support
0	0.96	0.88	0.91	1300
1	0.89	0.96	0.92	1300
accuracy			0.92	2600



Hypeparameter Tuning and Transfer Learning Models

MODEL COMPARISONS

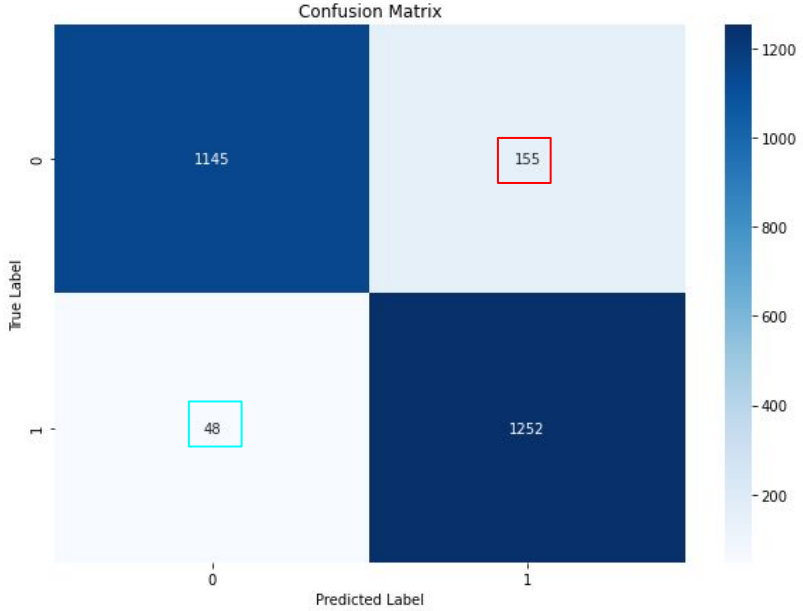



```
82/82 [==============================] - 18s 221ms/step - loss: 0.2675 - accuracy: 0.9219
Model loss on the test set: 0.26750826835632324
Model accuracy on the test set:  92.19%
```

VGG16 TL / Bayesian

This model **minorly** better with **92.1%** accuracy on the test set. Validation scores not improved - implies overfitting.

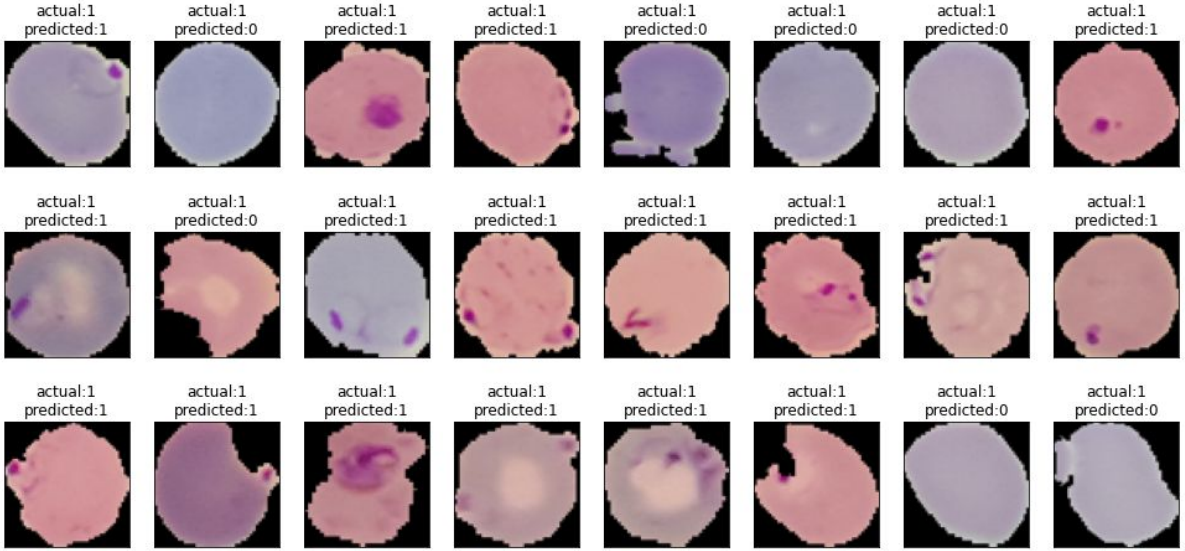
	precision	recall	f1-score	support
0	0.96	0.88	0.91	1300
1	0.89	0.96	0.92	1300
accuracy			0.92	2600



Model Choice

BAYESIAN OPTIMIZED MODEL (THE 2nd MODEL)

Overall it is the most effective model with a accuracy rate with a **93% recall, 97% precision and 95% f1-score** on the test set.



2

Recommendations for Implementation

Conclusion

BENEFITS AND RECOMMENDATIONS

- We've built an automated system can help with the **early** and **accurate detection of malaria** using a Neural Network automated classification techniques and evaluated our models through validation and test scores against actual labels.
- We noted that it can **significantly reduce the time** for addressing the problem significantly from **1,900 years** to just **201 days** taking on 2019's caseload.
- It presents a cost savings of approximately $380 million dollars a year in man hours.
- **Most importantly**, it can **save lives**. Potentially **400,000 lives in 2019 alone**.

IMPLEMENTATION

- The model requires images to be taken in a color format that can be converted to 64x64 image.
- It is recommended that a physician or trained health professional examine the scan following detection of malaria infection by the model to confirm accuracy of diagnosis.
- The model should be periodically retrained with new data to maintain accuracy.

4 Appendix

Data Exploration

DATA DESCRIPTION

There are a total of **24,958** train and **2,600** test images (colored) with ***an equal number*** of parasitized and uninfected instances, where:

- The parasitized cells contain the Plasmodium parasite
- The uninfected cells are free of the Plasmodium parasites but could contain other impurities

PREPPING THE DATASET

- The original dataset is available as **.png files** organized in folders.
- Since the data is already split into train and the test dataset in respective folders and further split into parasitized and uninfected folders. We will need to import from our folders and prep.
- We will convert all images to the **same size** *and* to **4D arrays** so that they can be used as an input for the convolutional neural network. Then we will need to create the labels for both types of images to be able to train and test the model.

Data Exploration

LOADING THE IMAGES

SIZING THE IMAGES

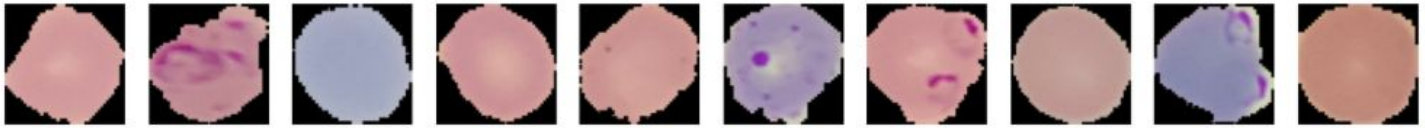
CONVERTING TO AN ARRAY

ADDING CORRECT LABELS

```
#We will run the same code for "parasitized" as well as "uninfected" folders within the "train" folder
for folder_name in ['/parasitized/', '/uninfected/']:

    #Path of the folder
    images_path = os.listdir(train_dir + folder_name)

    for i, image_name in enumerate(images_path):
        try:
            #Opening each image using the path of that image
            image = Image.open(train_dir + folder_name + image_name)

            #Resizing each image to (224,224)
            image = image.resize((SIZE, SIZE))

            #Converting images to arrays and appending that array to the empty list defined above
            train_images.append(np.array(image))

            #Creating labels for parasitized and uninfected images
            if folder_name=='/parasitized/':
                train_labels.append(1)
            else:
                train_labels.append(0)
        except Exception:
            pass

#Converting lists to arrays
train_images = np.array(train_images)
train_labels = np.array(train_labels)
```

Data Exploration

PRINTING OUT SOME OF THE IMAGES AND LOOKING AT THE ASSOCIATED LABELS – AND IT'S CORRECTLY LABELLED



```
label for each of the above image: [0 1 0 0 0 1 1 0 1 0]
```

- We print out a random set of images and compared with the labels and they correspond.
- Next we will take a look at the dataset and check the shape - it should be 4D arrays and should not need to be further reshaped.

```
Training set: (24958, 224, 224, 3) (24958,)
Test set: (2600, 224, 224, 3) (2600,)
```

NEXT WE CHECK TO SEE IF OUR DATA IS BALANCED BETWEEN POSITIVE AND NEGATIVE CASES AND WE SEE THAT WE HAVE AN EQUAL NUMBER OF POSITIVE AND NEGATIVE CASES

```
Positive cases in the training set :  12479
Negative cases in the training set :  12479
Positive cases in the test set :  1300
Negative cases in the test set :  1300
```

Proposed Approach

POTENTIAL TECHNIQUES

An automated system can help with the early and accurate detection of malaria. Applications of automated classification techniques using Machine Learning (ML) and Artificial Intelligence (AI) have consistently shown higher accuracy than manual classification. It would therefore be highly beneficial to propose a method that performs malaria detection using Deep Learning Algorithms.

OVERALL SOLUTION DESIGN

We will be using building a convolution neural network (CNN) model to classify between infected and non-infected blood cell samples from the images. This is a binary classification problem so we will be using a sigmoid activation function.

MEASURES OF SUCCESS

We will attempt various models and test accuracy with a loss score. We will also use a 20% validation split for each model to see if training and validation accuracy improve together to determine if the model is overfitting.

Data Preparation

OPTIMIZING THE IMAGE SIZE

As we build our CNN models we learn that we expend much less computational resources by resizing our image from 224x224 to 64x64. Below we compare the 9th and 10th epoch and we can see that the larger image had comparable and even slightly lower accuracy score but took 10 times longer to process. This lead me to build the rest of the models using 64x64 image size.

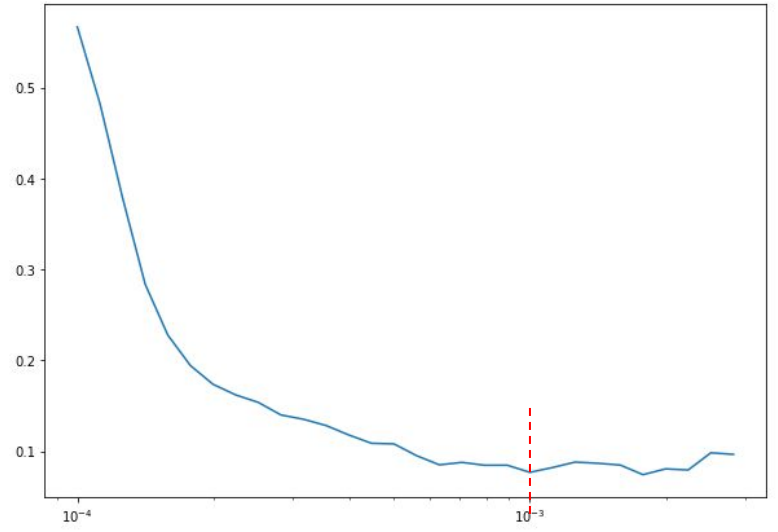
```
Epoch 9/10
624/624 - 501s - loss: 0.0507 - accuracy: 0.9834 - val_loss: 0.6188 - val_accuracy: 0.8233
Epoch 10/10
624/624 - 502s - loss: 0.0560 - accuracy: 0.9827 - val_loss: 0.7765 - val_accuracy: 0.7841
```

```
624/624 - 48s - loss: 0.1621 - accuracy: 0.9404 - val_loss: 0.4836 - val_accuracy: 0.8257
Epoch 9/30
624/624 - 47s - loss: 0.1539 - accuracy: 0.9452 - val_loss: 0.4271 - val_accuracy: 0.8301
Epoch 10/30
```

Refined Insight

OPTIMIZING THE LEARNING RATE

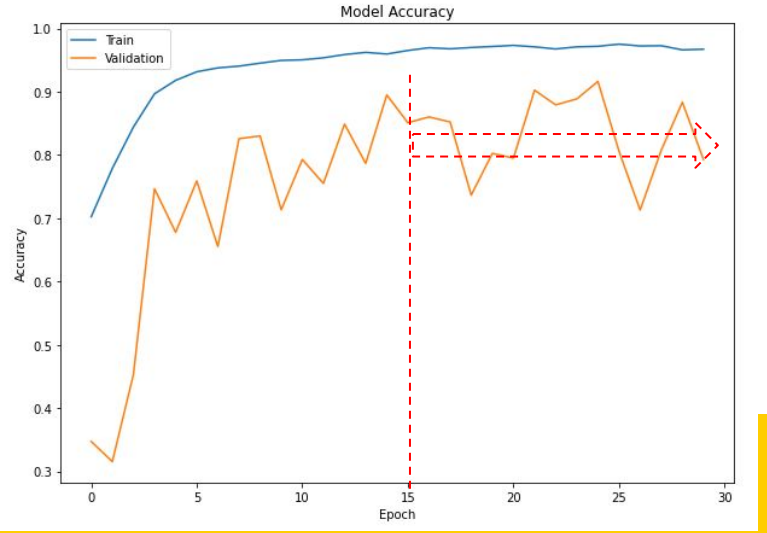
Early on in the model building process, we defined a function to test out different learning rates to find the best learning rate for the model. We plotted our learning rate verse the loss and we found that the loss plateaus at 10^-3, so we used 0.001 as our learning rate.



Refined Insight

OPTIMIZING THE NUMBER OF EPOCHS

We also started with a higher number of epochs and plotted accuracy per epoch and we found that our accuracy starts to taper off starting at 15 epochs so we set our model to train up to 15 epochs.



Comparison of Various Techniques

MODEL COMPARISONS

We did 4 Convolutional Neural Network models and saw the results progressive improved. Let's compare the models to evaluate various techniques and each model's performance. We used the Leaky Relu activation function for the hidden layers and a Sigmoid activation function for the output layer. Below we see the adjustments we made for each model.

CNN Model 1

Sequential Model
Layer 1 16 nodes
Layer 2 32 nodes
Max Pooling Layer
Flatten

Dense Layer 32 nodes
Sigmoid Activation

CNN Model2

Sequential Model
Layer 1 16 nodes
+ 20% Dropout Layer
Layer 2 32 nodes
+ 20% Dropout Layer
Max Pooling Layer
Flatten

Dense Layer 32 nodes
+50% Dropout Layer
Sigmoid Activation

CNN Model3

Sequential Model
Layer 1 16 nodes
+ 32 Node Layer
+ Max Pooling Layer
+ 25% Dropout Layer
Layer 2 32 nodes
+ 64 Node Layer
+ 25% Dropout Layer
Max Pooling Layer
Flatten

Dense Layer 32 nodes
50% Dropout Layer
Sigmoid Activation

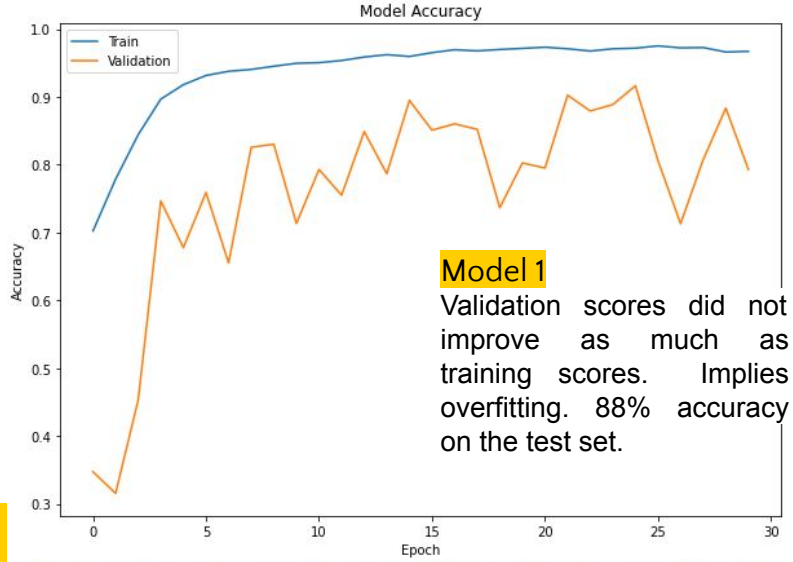
CNN Model4

Sequential Model
Layer 1 16 nodes
+ 20% Dropout Layer
32 Node Layer
+ 64 nodes
+ 20% Dropout Layer
Max Pooling Layer
Flatten

Dense Layer 32 nodes
+50% Dropout Layer
Sigmoid Activation

Comparison of Various Techniques

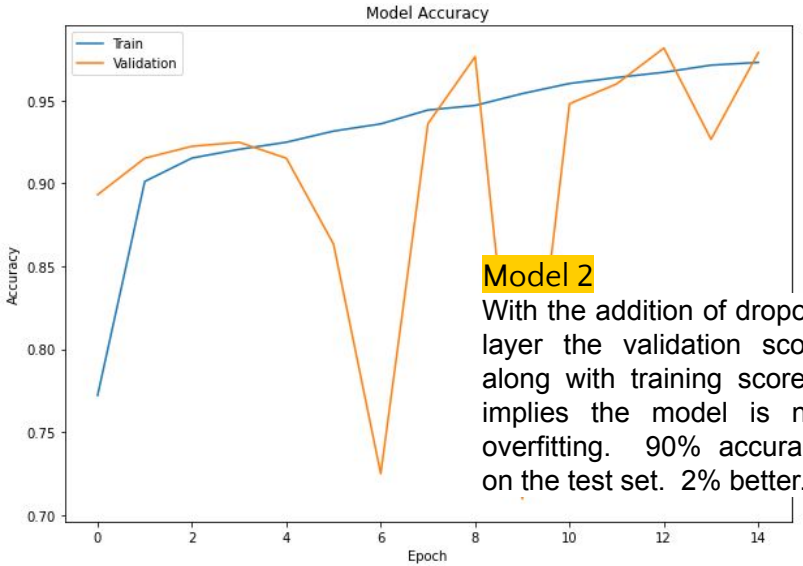
MODEL COMPARISONS



82/82 [==============================] - 2s 25ms/step - loss: 0.4659 - accuracy: 0.8819
Model loss on the test set: 0.46586668491363525
Model accuracy on the test set: 88.19%

Model 1

Validation scores did not improve as much as training scores. Implies overfitting. 88% accuracy on the test set.



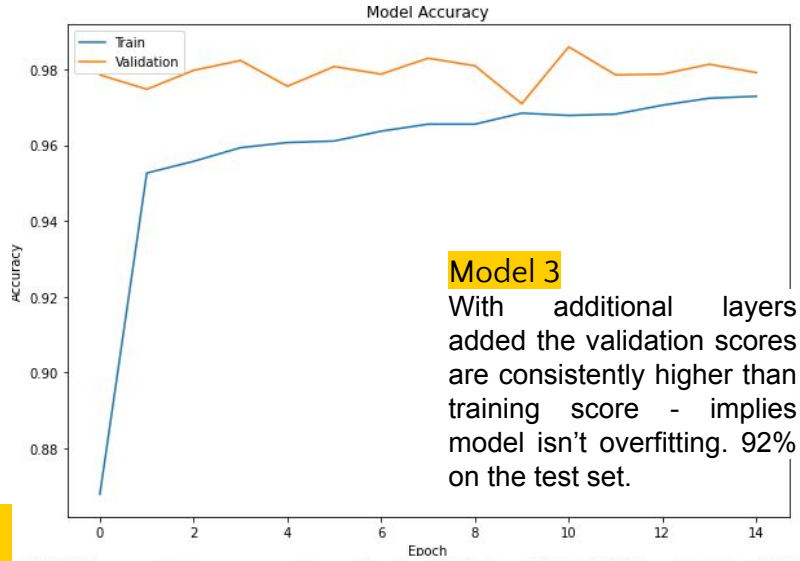
82/82 [==============================] - 2s 21ms/step - loss: 0.4370 - accuracy: 0.9058
Model loss on the test set: 0.43698763847351074
Model accuracy on the test set: 90.58%

Model 2

With the addition of dropout layer the validation score along with training score - implies the model is not overfitting. 90% accuracy on the test set. 2% better.

Comparison of Various Techniques

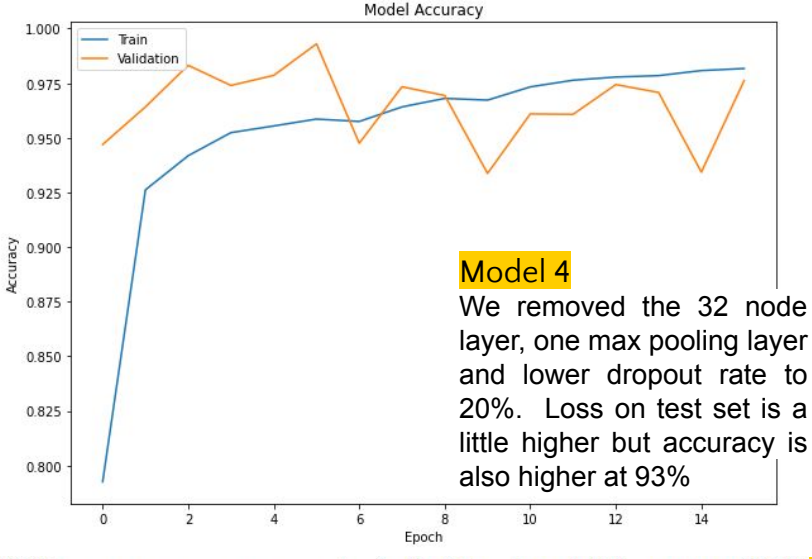
MODEL COMPARISONS



Model 3

With additional layers added the validation scores are consistently higher than training score - implies model isn't overfitting. 92% on the test set.

```
82/82 [==============================] - 3s 33ms/step - loss: 0.2555 - accuracy: 0.9292
Model loss on the test set: 0.2554693818092346
Model accuracy on the test set:  92.92%
```

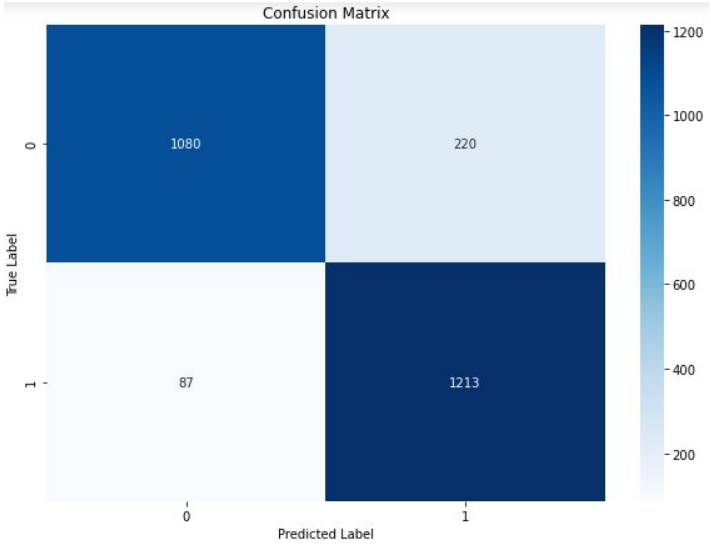


Model 4

We removed the 32 node layer, one max pooling layer and lower dropout rate to 20%. Loss on test set is a little higher but accuracy is also higher at 93%

```
82/82 [==============================] - 5s 55ms/step - loss: 0.3608 - accuracy: 0.9327
Model loss on the test set: 0.36076441407203674
Model accuracy on the test set:  93.27%
```

Comparison of Various Techniques



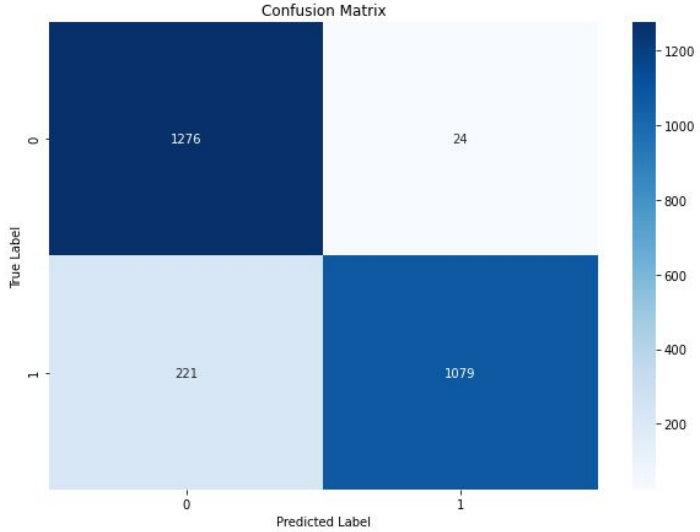
	precision	recall	f1-score	support
0	0.93	0.83	0.88	1300
1	0.85	0.93	0.89	1300
accuracy			0.88	2600
macro avg	0.89	0.88	0.88	2600
weighted avg	0.89	0.88	0.88	2600

Model 1

Since having false negatives would be worse than false positives in malaria detection, we want to emphasize Recall score. This model has 93% recall, 85% precision and 89% f1-score on the test set.

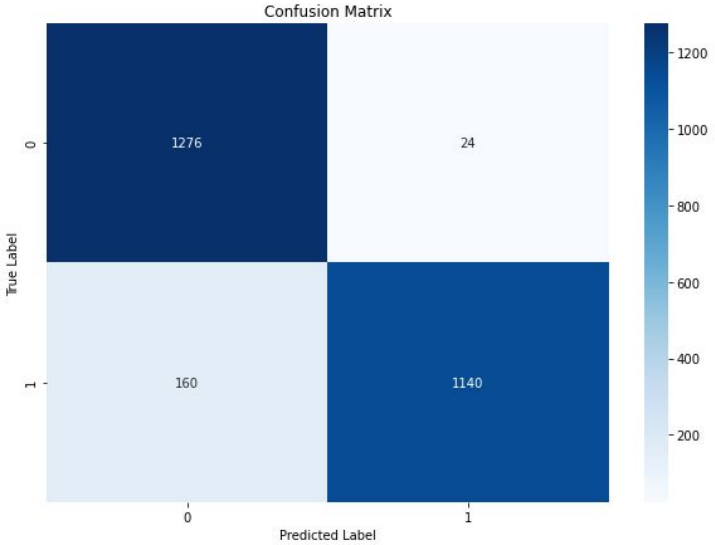
Model 2

In comparison, model 2 has 83% recall, 98% precision and 90% f1-score. It has much more false negatives than model which is not optimal.



	precision	recall	f1-score	support
0	0.85	0.98	0.91	1300
1	0.98	0.83	0.90	1300
accuracy			0.91	2600
macro avg	0.92	0.91	0.91	2600
weighted avg	0.92	0.91	0.91	2600

Comparison of Various Techniques



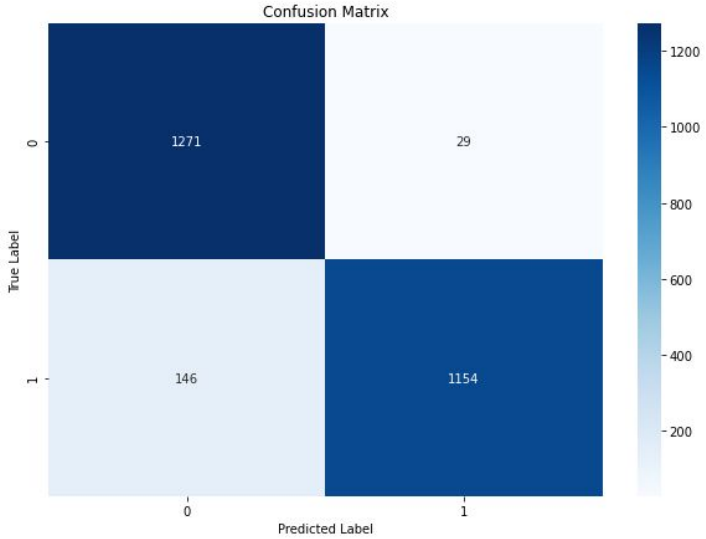
	precision	recall	f1-score	support
0	0.89	0.98	0.93	1300
1	0.98	0.88	0.93	1300
accuracy			0.93	2600
macro avg	0.93	0.93	0.93	2600
weighted avg	0.93	0.93	0.93	2600

Model 3

model 3 has 88% recall, 98% precision and 93% f1-score. It's an improvement on model 2.

Model 4

model 4 has 89% recall, 98% precision and 93% f1-score. It has a little more false positives but it also has less false negative than model 3, which is a good thing.



	precision	recall	f1-score	support
0	0.90	0.98	0.94	1300
1	0.98	0.89	0.93	1300
accuracy			0.93	2600
macro avg	0.94	0.93	0.93	2600
weighted avg	0.94	0.93	0.93	2600