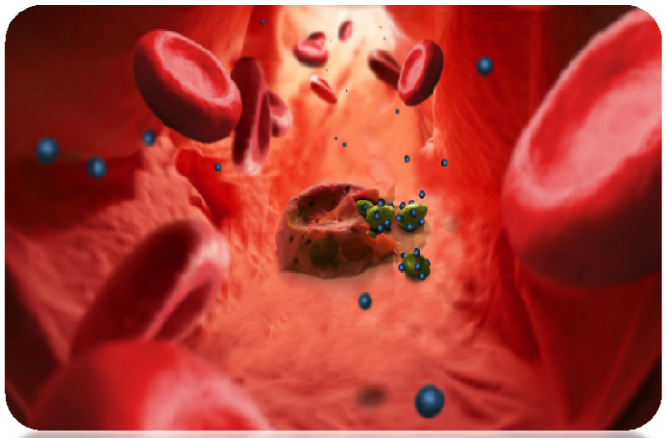




Deep Learning for Malaria Detection

Ashish Kumar

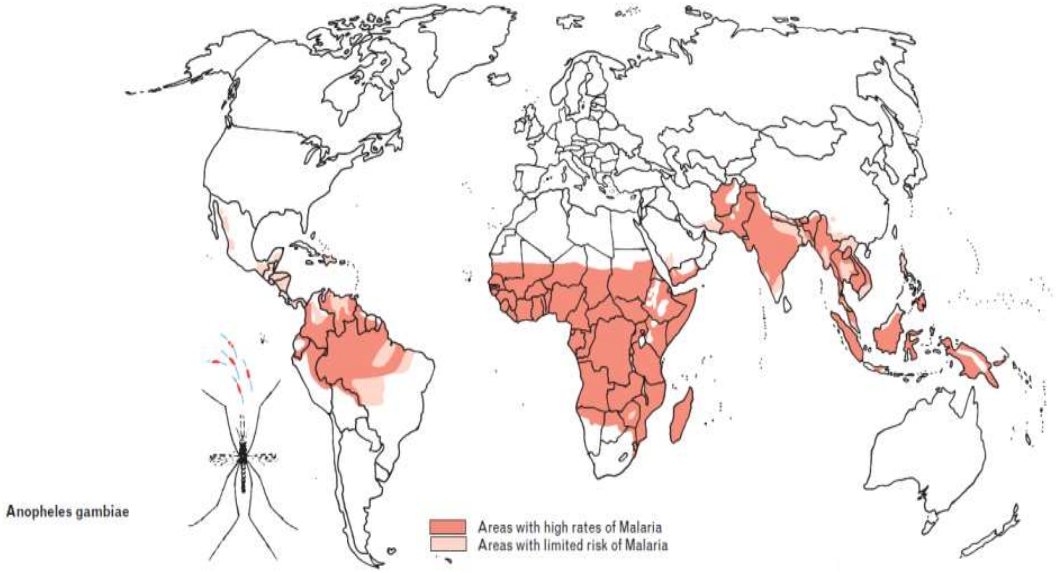
Objective

Problem Statement

- *Malaria remains a global health concern*
- *Microscopic diagnosis, though highly accurate, is tedious, time consuming, and highly dependant on technician's expertise*
- *Early and accurate detection can dramatically improve morbidity and mortality rates*

Solution

- *Develop a CNN (Convolution Neural Network) model to analyze blood smear images; and improve efficiency, accuracy, and scalability of malaria diagnosis*



Source - https://www.iamat.org/risks/malaria

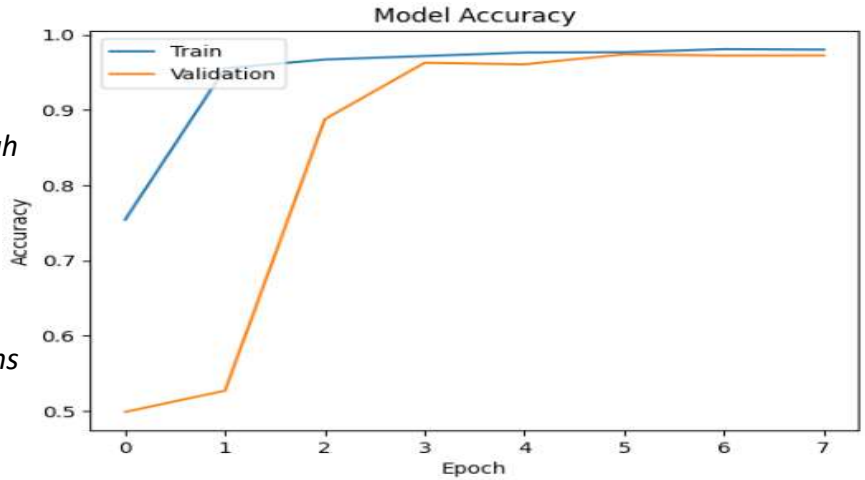
Observations & Next Steps

Observations:

- *High Accuracy – Compared to the other models, this model achieved high accuracy (f1-score = 0.98, recall = 0.99), and classified only 8 false positives (predicted = uninfected, actual = parasitized)*
- *Generalization – The convergence between training and validation accuracy (98%) indicates the model has generalized well to the variations in the dataset*
- *Training time – The model took only 8 epochs to train, even with 4 convolution layers (557K parameters to learn)*

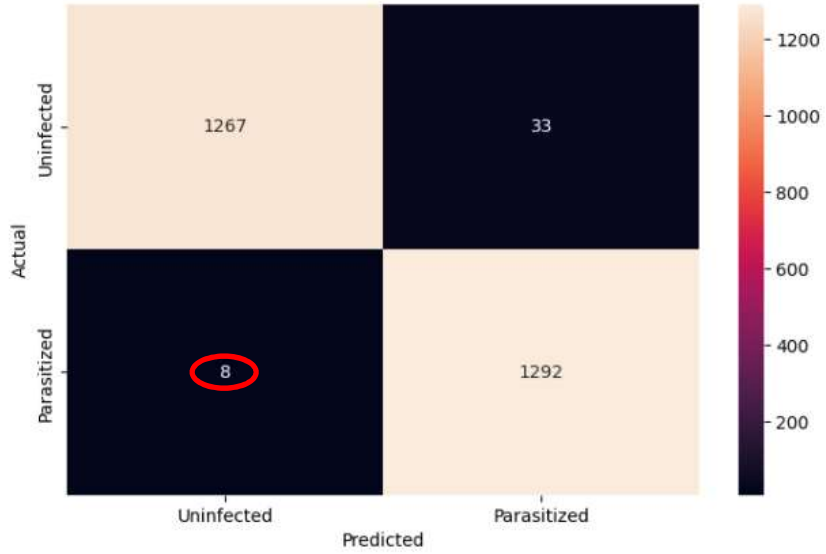
Next Steps

- *Data Quality/Bias – Training the models with high-quality labeled datasets that have diversity in geographic regions, patient demographics, or the prevalence of specific malaria species would be essential to limit bias and improve generalization of the model*
- *Explainability – In order to make the model more interpretable explore what features are being learnt and how they influence the classification. This can be done through SHAP and/or plotting what features are being learned by each layer of the CNN*



	precision	recall	f1-score	support
0	0.99	0.97	0.98	1300
1	0.98	0.99	0.98	1300
accuracy			0.98	2600
macro avg	0.98	0.98	0.98	2600
weighted avg	0.98	0.98	0.98	2600

Confusion Matrix



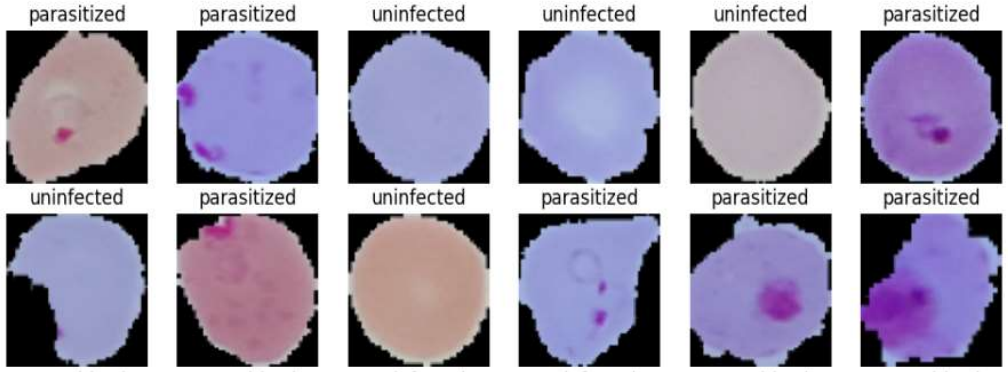
Solution Architecture

Dataset:

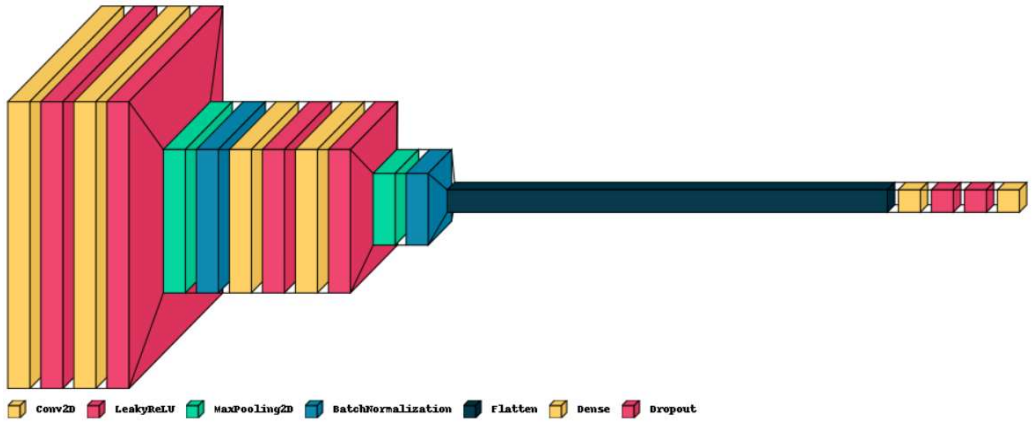
- *Train – 25K labeled RGB images (12.6K parasitized, 12.4K uninfected)*
- *Test – 2.6K labeled RGB images (1.3K parasitized, 1.3K uninfected)*

Model:

- *4 convolution 2D layers of 16, 32, 32, 64 filters respectively to learn features such as what species of parasite is present, what is the stage of infection, what are the changes in RBC morphology, etc.*
- *LeakyReLU activation function with a negative slope of 0.1 to capture complex relationships within the data*
- *Max pooling with filter size 2x2 to extract dominant features*
- *Regularization –*
  - *Batch Normalization to fasten training and reduce sensitivity to initialization*
  - *Dropout rate of 0.5 to reduce overfitting*
- *Output layer – sigmoid activation for binary classification*



Input Dataset



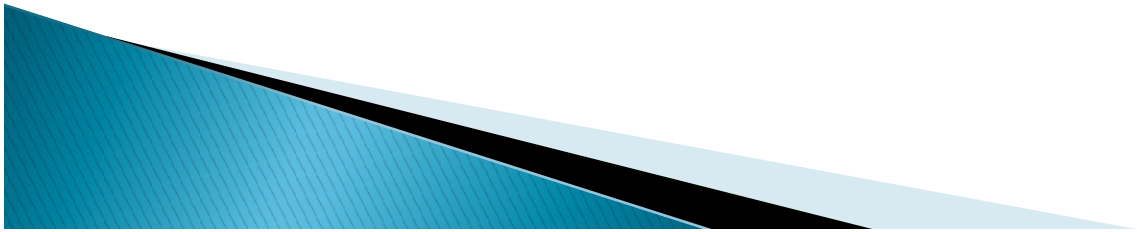
Malaria Detection CNN Model Architecture

Things to Consider

- *Deployment –*
  - *Model conversion – Convert the model to a format compatible with the desired deployment environment. This may involve converting to TensorFlow Lite for mobile deployment or ONNX for cloud deployment*
  - *API development – Develop an API that allows users to submit blood smear images and receive predictions from the model*
  - *Integration – Integrate the API into the desired platform (web app, mobile app, etc.)*

Other factors:

- *Computation resources – Training deep CNN is computationally expensive and can be a challenge if hardware resources are limited*
- *Maintenance – Keeping up with new parasites strains and lab practices would be key to the long term efficacy of the CNN model, which could be cost prohibitive*
- *Accessibility – CNN model when deployed in areas with constrained healthcare personnel resources, can increase accessibility and provide significant public health benefits*



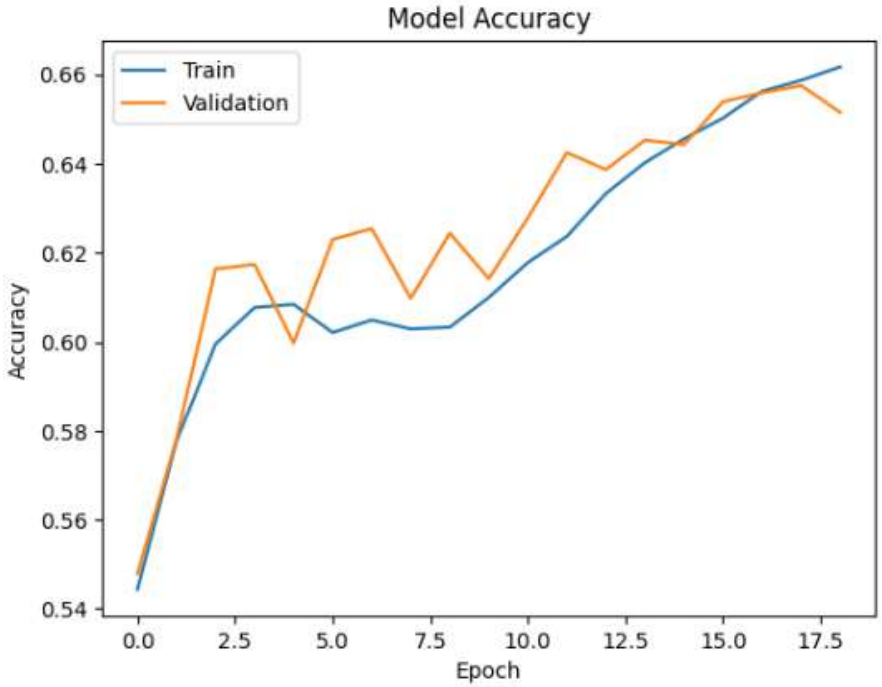
Appendix

Model Comparison

Model Name	Model Technique	Activation	Optimizer	Epoch	Batch Size	Image type	Trainable Params	Non-Trainable Params	Ttl Params	Train-Time (# of Epochs)	Train Accuracy	Val Accuracy	Test Accuracy	F1 score	Precision	Recall
model_base	2 CONV2D + Maxpooling + Flatten + Dense + Output	relu	SGD	20	64	BRG normalized	1,053,762	0	1,053,762	20	66%	65%	60%	0.65	0.58	0.72
model1	2 CONV2D + Maxpooling + 2 CONV2D + Maxpooling + Flatten + Dense + Output	relu	Adam	30	128	BRG normalized	557,218	0	557,218	12	98%	98%	99%	0.99	0.98	0.99
model2	2 CONV2D + Maxpooling + BatchNormalization + 2 CONV2D + Maxpooling + BatchNormalization + Flatten + Dense + Droput (0.5) + Output	LeakyReLU(0.1)	Adam	30	128	BRG normalized	557,410	192	557,602	8	98%	97%	98%	0.98	0.98	0.99
model3	2 CONV2D + Maxpooling + BatchNormalization + 2 CONV2D + Maxpooling + BatchNormalization + Flatten + Dense + Droput (0.5) + Output	LeakyReLU(0.1)	Adam	30	128	Image Data Generator	557,410	192	557,602	8	92%	96%	96%	0.96	0.94	1.00
model4	VGG16 + Flatten + Output		Adam	30	128	BRG normalized	1,026	14,714,688	14,715,714	30	94%	94%	91%	0.91	0.91	0.92
model_hsv	2 CONV2D + Maxpooling + 2 CONV2D + Maxpooling + 2 CONV2D + Maxpooling + Flatten + Dense + Droput (0.4) + Output	LeakyReLU(0.1)	Adam	30	128	HSV normalized	926,242	0	926,242	4	98%	98%	99%	0.99	0.98	0.99
model_gb	2 CONV2D + Maxpooling + BatchNormalization + 2 CONV2D + Maxpooling + BatchNormalization + 2 CONV2D + Maxpooling + BatchNormalization + Flatten + Dense + Droput (0.4) + Output	LeakyReLU(0.1)	Adam	30	128	Gausian Blur normalized	926,946	704	927,650	22	98%	98%	98%	0.98	0.97	0.99

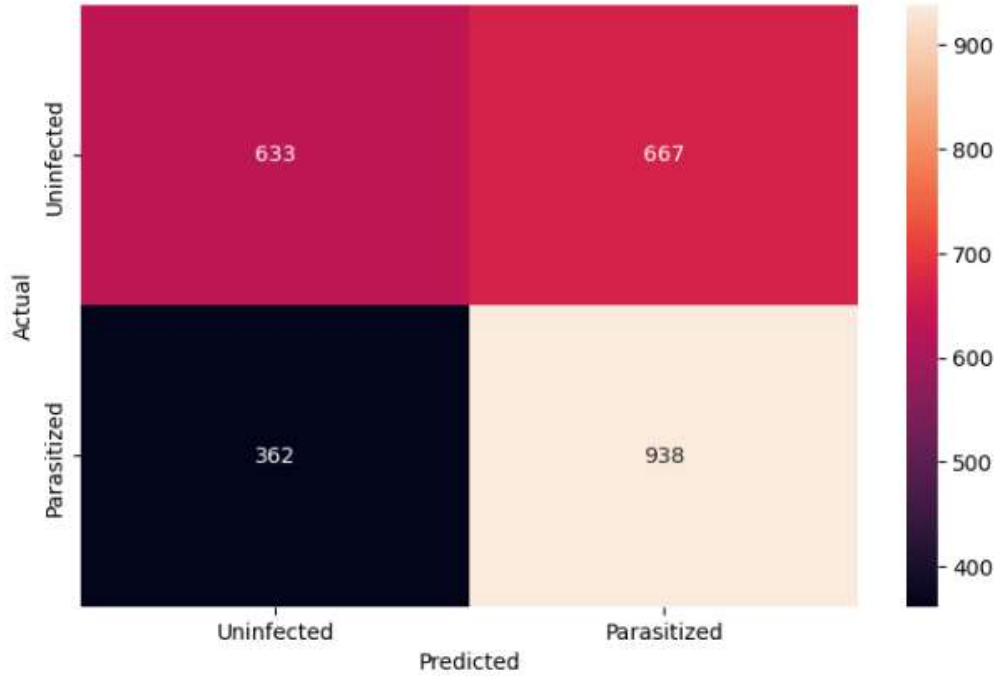
Final Model

Base Model – Training Accuracy & Confusion Matrix



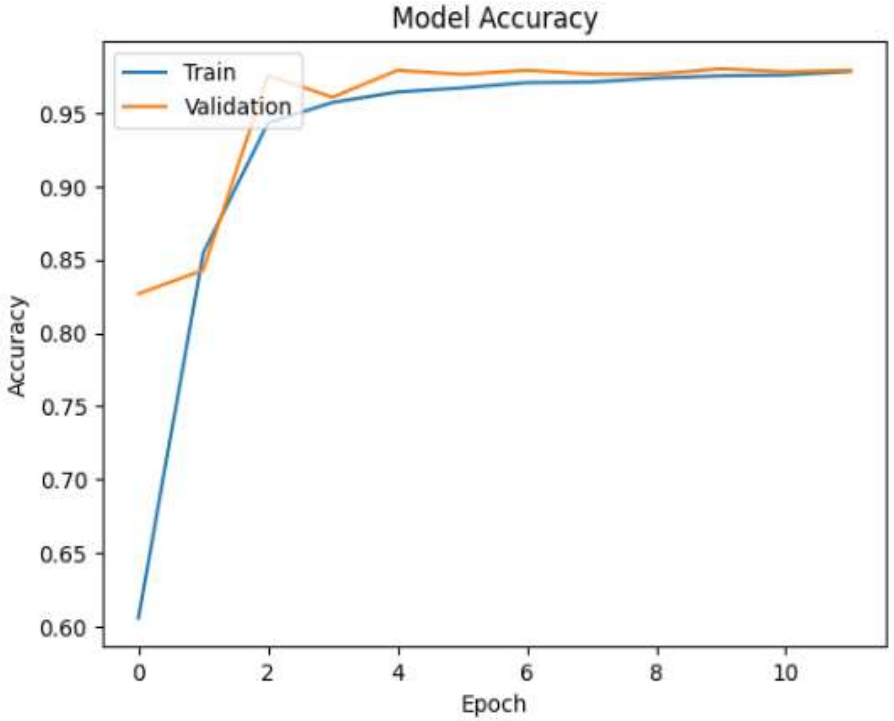
Confusion Matrix

	precision	recall	f1-score	support
0	0.64	0.49	0.55	1300
1	0.58	0.72	0.65	1300
accuracy			0.60	2600
macro avg	0.61	0.60	0.60	2600
weighted avg	0.61	0.60	0.60	2600



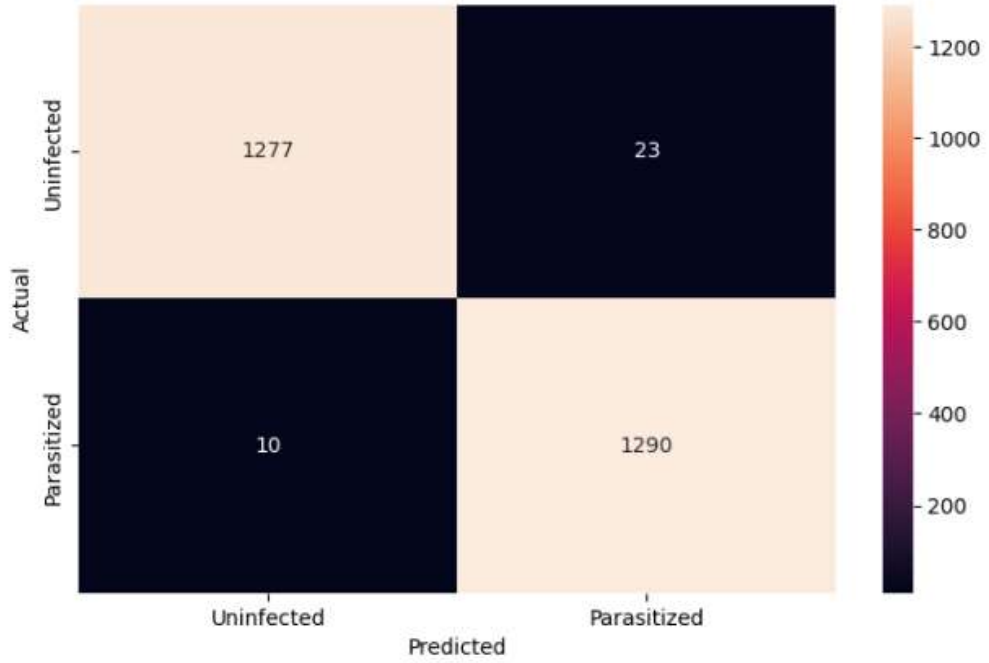
Train accuracy – 66.16%
Validation accuracy – 65.14%
Test accuracy – 60.42%

Model1 – Training Accuracy & Confusion Matrix



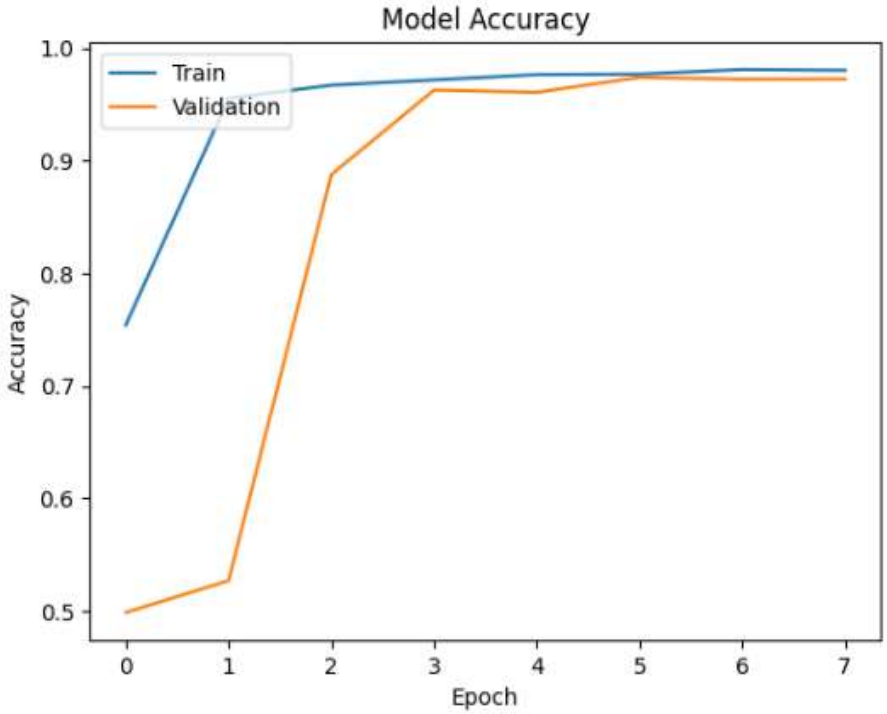
Confusion Matrix

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1300
1	0.98	0.99	0.99	1300
accuracy			0.99	2600
macro avg	0.99	0.99	0.99	2600
weighted avg	0.99	0.99	0.99	2600



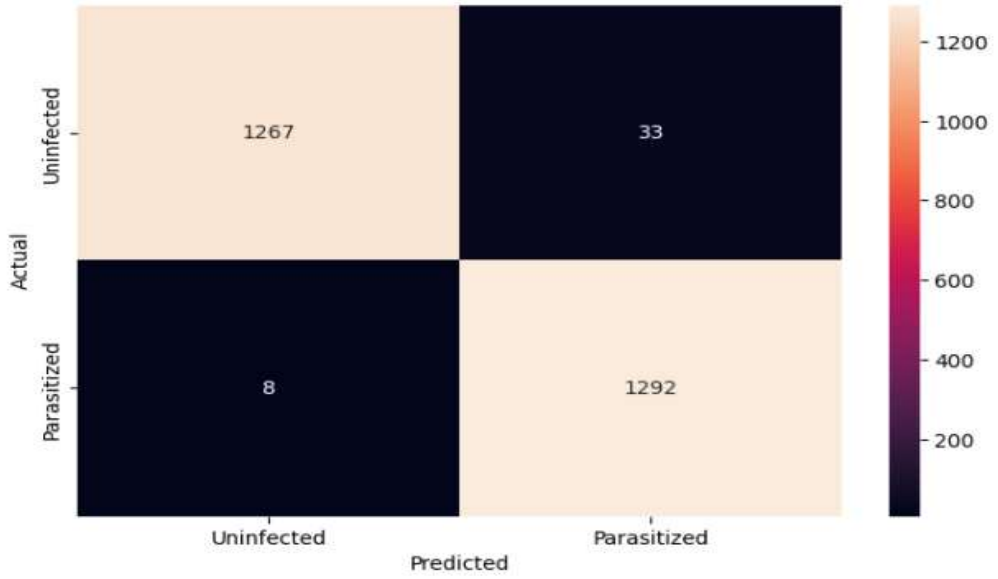
Train accuracy – 97.85%
Validation accuracy – 97.92%
Test accuracy – 98.73%

Model2 – Training Accuracy & Confusion Matrix



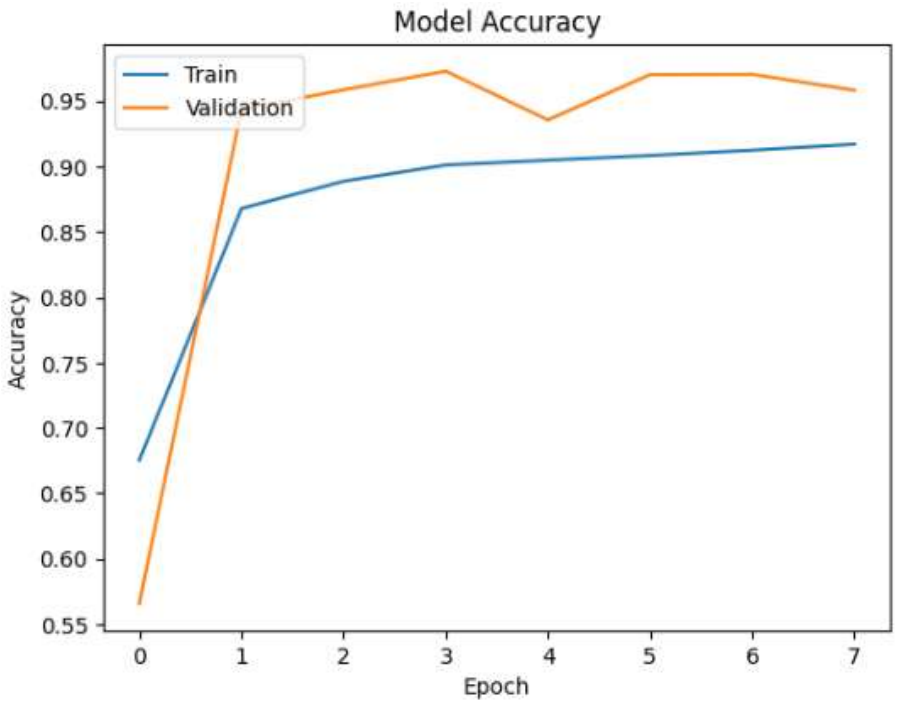
Confusion Matrix

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1300
1	0.98	0.99	0.98	1300
accuracy			0.98	2600
macro avg	0.98	0.98	0.98	2600
weighted avg	0.98	0.98	0.98	2600



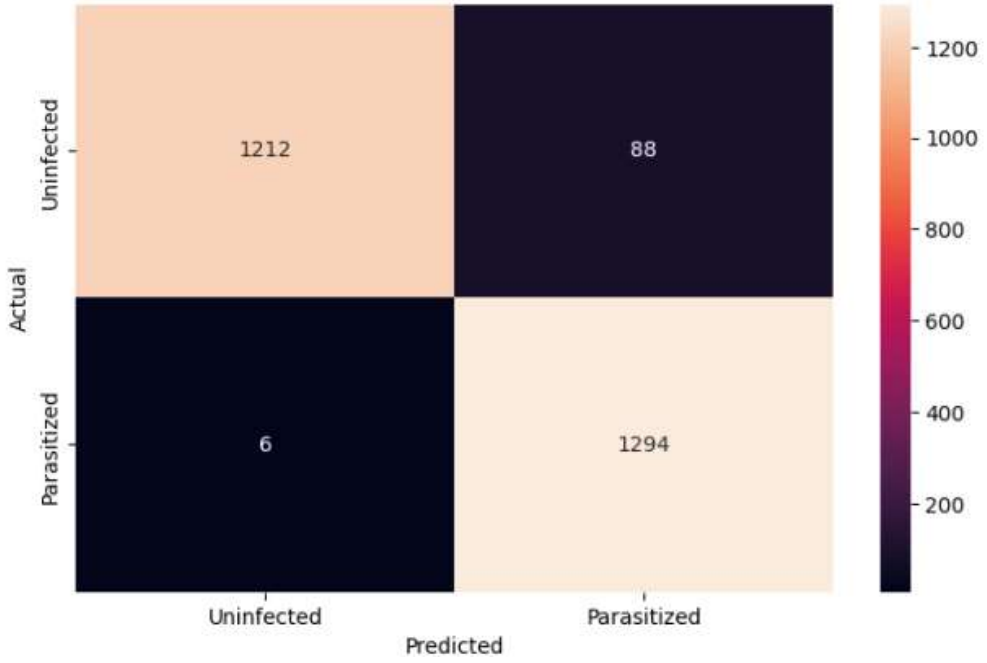
Train accuracy – 98.01%
Validation accuracy – 97.24%
Test accuracy – 98.42%

Model3 – Training Accuracy & Confusion Matrix



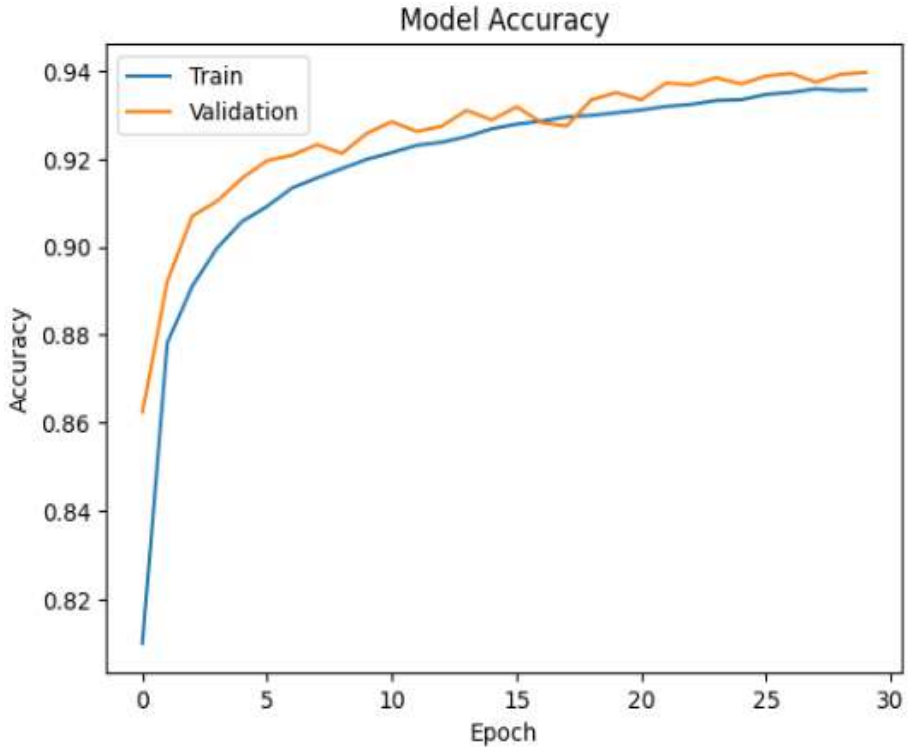
Confusion Matrix

	precision	recall	f1-score	support
0	1.00	0.93	0.96	1300
1	0.94	1.00	0.96	1300
accuracy			0.96	2600
macro avg	0.97	0.96	0.96	2600
weighted avg	0.97	0.96	0.96	2600



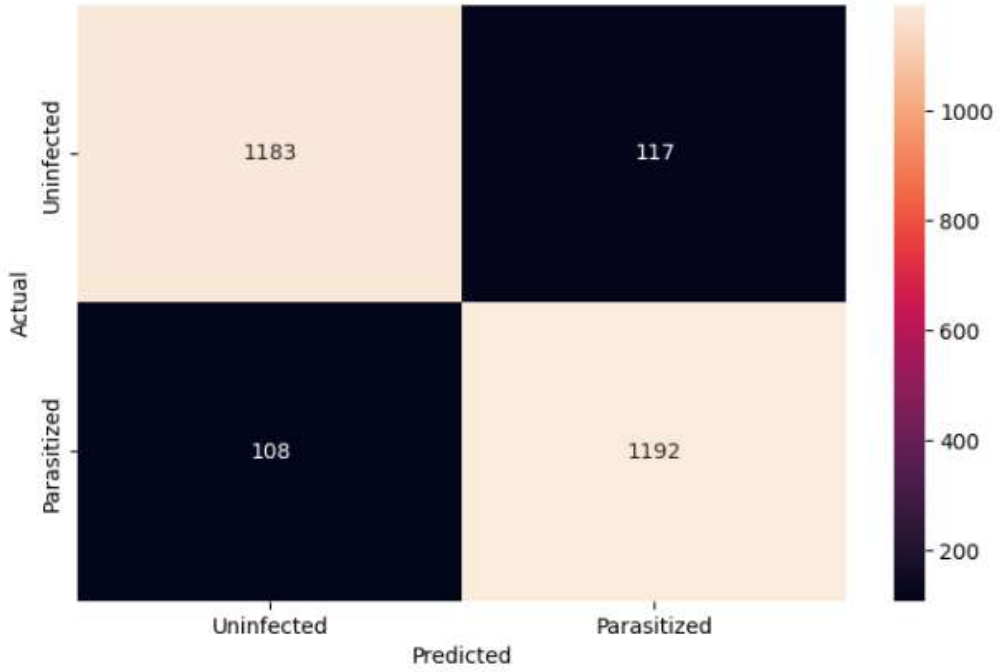
Train accuracy – 91.7%
Validation accuracy – 95.81%
Test accuracy – 96.38%

Model4 – Training Accuracy & Confusion Matrix



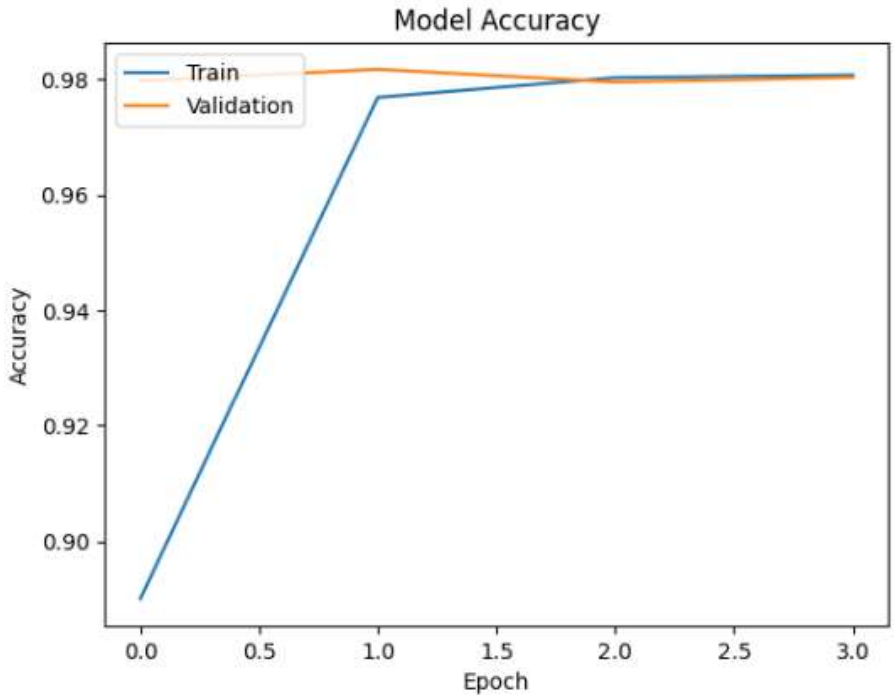
Confusion Matrix

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



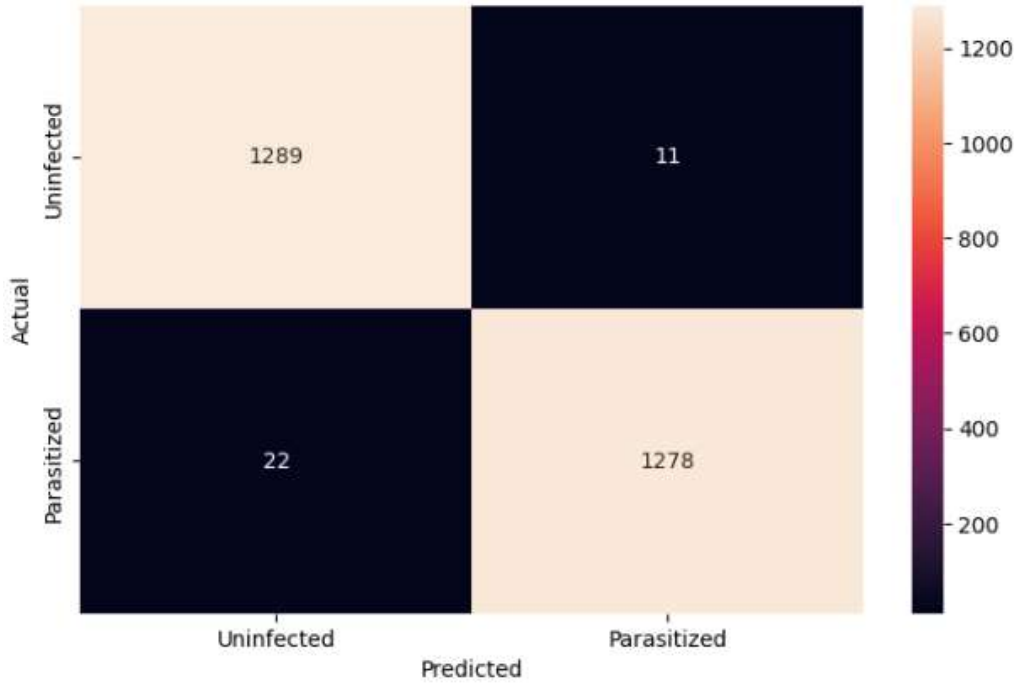
Train accuracy – 93.57%
Validation accuracy – 93.97%
Test accuracy – 91.35%

Model_HSV – Training Accuracy & Confusion Matrix



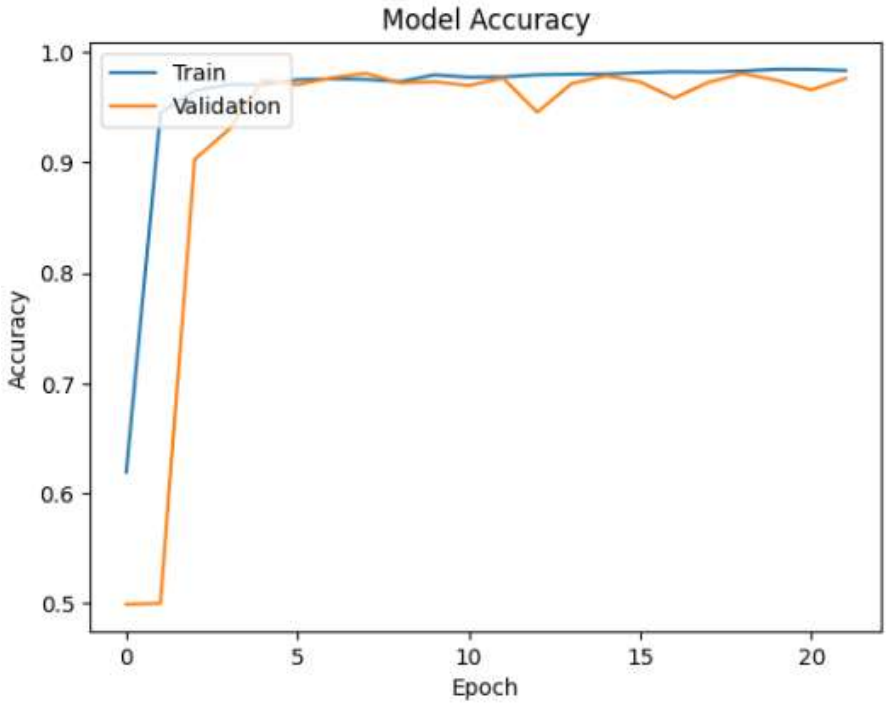
Confusion Matrix

	precision	recall	f1-score	support
0	0.98	0.99	0.99	1300
1	0.99	0.98	0.99	1300
accuracy			0.99	2600
macro avg	0.99	0.99	0.99	2600
weighted avg	0.99	0.99	0.99	2600



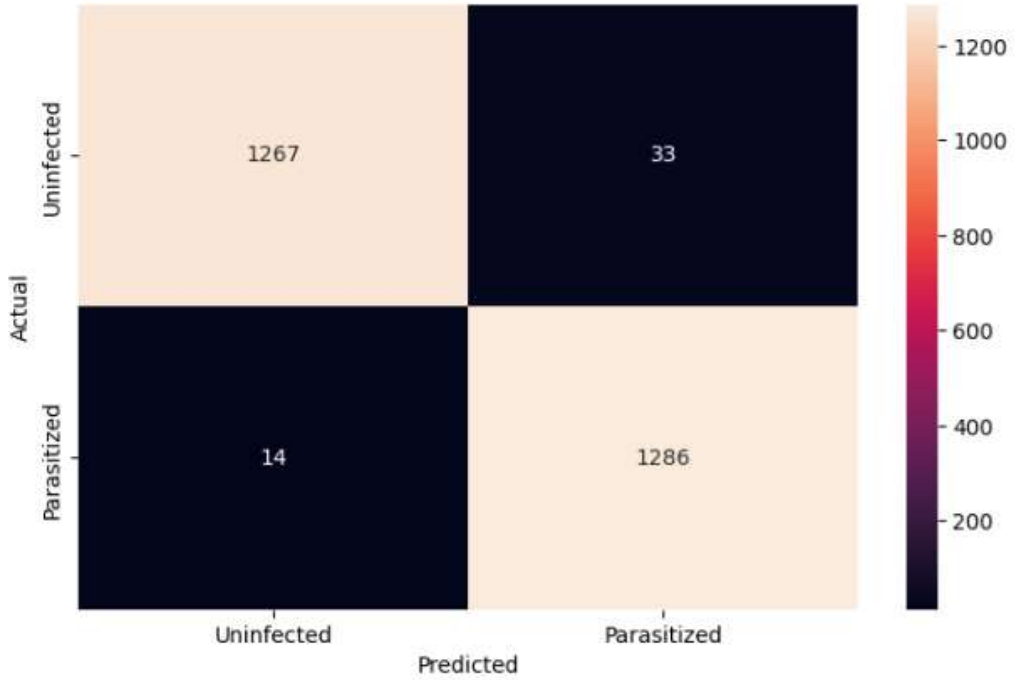
Train accuracy – 98.08%
Validation accuracy – 98.04%
Test accuracy – 98.73%

Model_GB – Training Accuracy & Confusion Matrix



Confusion Matrix

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1300
1	0.97	0.99	0.98	1300
accuracy			0.98	2600
macro avg	0.98	0.98	0.98	2600
weighted avg	0.98	0.98	0.98	2600



Train accuracy – 98.33%
Validation accuracy – 97.6%
Test accuracy – 98.19%