Table of Contents

List of Figures

List of Tables

List Of Equations

Malaria Detection Using Computer Vision

1.0 Introduction

Malaria is a life-threatening disease caused by the plasmodium parasite which is transmitted via the bites of an infected female anopheles' mosquito (Mayo Clinic, 2023), and is the 5th leading cause of death in low-income countries (World Health Organization, 2020).

An effective way to reduce malaria deaths is early diagnosis and treatment (Mbanefo and Kumar, 2020), but the traditional approach of manually examining blood slides via with microscopes by expert microscopists to detect malaria misses over 25% of malaria cases (Nema et al., 2022). This happens because firstly, in the low-income countries, there is scarcity a of expert microscopists and resources (Nema et al., 2022), and secondly the process of detecting the requires a lot of time and effort and this potentially leads to errors due to fatigue (Rees-Channer et al., 2023).

Convolutional Neural Networks (CNN) can be trained on microscopic images of blood smears to identify if a person has malaria or not. CNN models can mitigate the issues with manual detection of malaria because firstly, CNN models that have been properly trained can have the same or an even greater accuracy in prediction than an expert microscopist (Alzubaidi et al., 2021), solving the issue of expert scarcity. Secondly deep learning models like the CNN can process much more images in a shorter time frame than humans (De Man et al., 2019), and this solves the issue of time consumption and fatigue that microscopists face when manually detecting malaria.

2.0 Background

Several research has been undertaken to detect malaria using machine learning techniques, for example, (Siłka et al., 2023) built a CNN model to detect malaria from blood samples that had a 99.68% accuracy on the "Malaria Bounding Boxes" dataset on Kaggle.com.

(Kumar, Priya and Kumar, 2023) built a two-layer segmentation CNN to detect malaria on the malaria Cell Image Dataset from the official NIH website and they achieved an accuracy of 95.40%.

(Hemachandran et al., 2023) compared the performance of a basic CNN model, the MobileNetV2 and the ResNet50 in detecting malaria on the malaria Cell Image Dataset from the official NIH website and they determined that the MobileNetV2 performed best with a classification accuracy of 97.06%.

(Kumar, Sarkar and Pradhan, 2019) compared the performance of the Stochastic Gradient Decent (SGD), Adaptive Moment Estimation (ADAM) and RMSprop optimizers to determine which one had the best accuracy in malaria detection. They determined that the ADAM optimizer performed best with a classification accuracy of 96.62%.

(Khaled Almezhghwi, 2022), trained and finetuned the GoogleNet model on the malaria Cell Image Dataset from the official NIH website and an accuracy of 95% was achieved.

(Vijayalakshmi A and Rajesh Kanna B, 2019) determined that a hybrid of the VGG16 model and the SVM machine learning model outperforms the individual models with an accuracy of 93.10% on a malaria digital corpus dataset.

2.0. Aims

The aim of this research was to compare the performance of several pretrained models including the Inception V3, VGG16, VGG19, ResNet101V2 and mobile net with the performance of a basic CNN model, a hybrid of a CNN and a Random Forest (CNN-RF) and a hybrid of a CNN and a Support Vector

Machine (CNN-SVM) to determine the models that performs best in identifying malaria from microscopic images of blood smears.

The goal was to first train each of the pretrained models using weights from image net and then fine tune them on the dataset to ensure that they all performed at their best before comparing with the performance of the CNN, CNN-RF and the CNN-SVM.

3.0. Methodology

This section describes the data set and the methodology used for this research. The flow of the methodology can be seen in the flow diagram in Figure 1 below.

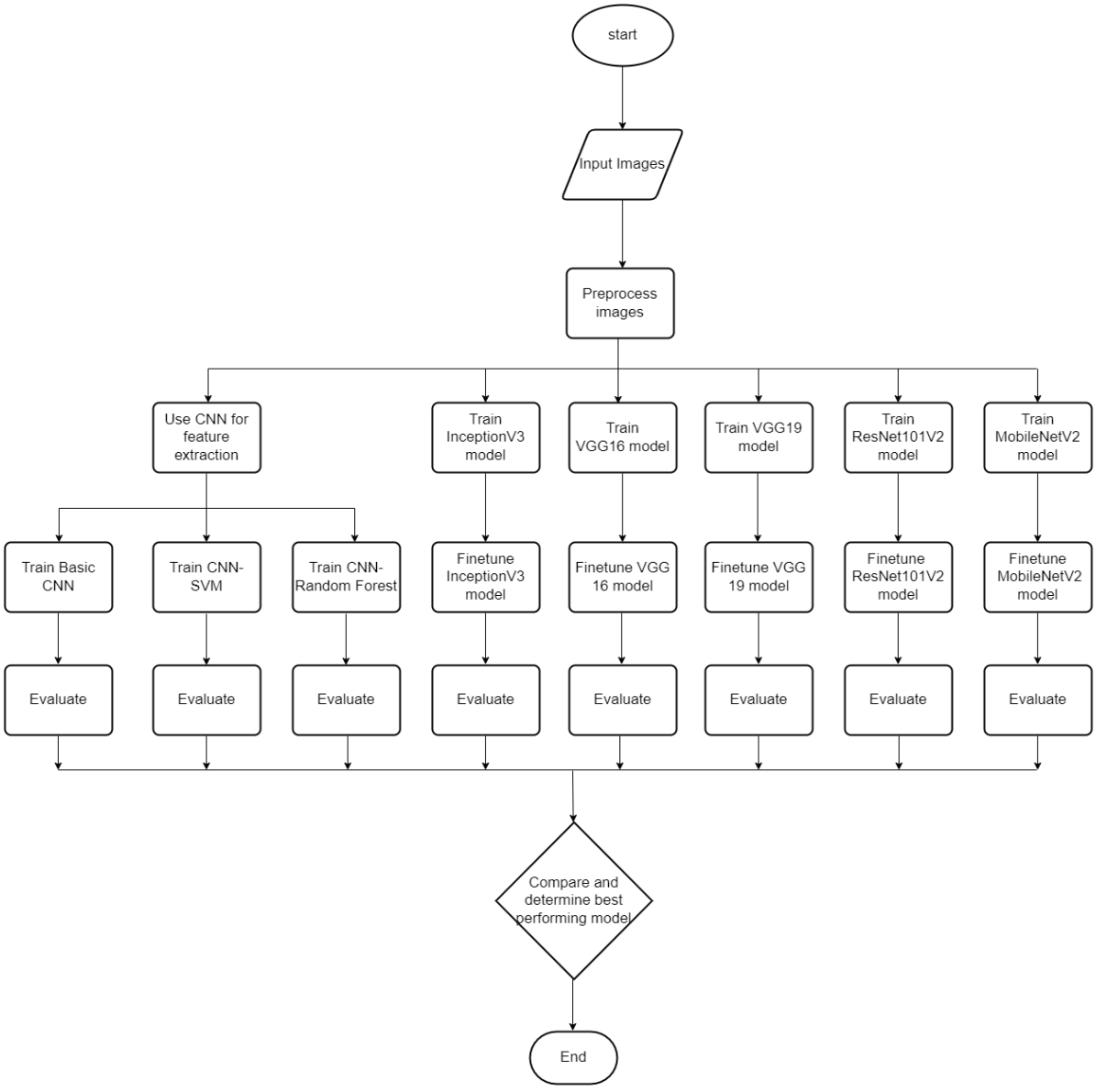

Figure 1. Flow diagram showing the methodology used for the study (drawn using the draw.io online drawing tool).

3.1. Environmental Setup

3.1.1. Hardware

All experiments were carried out using cloud computing on google Collaboratory. The cloud hardware used can be seen below.

CPU: Intel Xeon CPU with 2 vCPUs (virtual CPUs)

RAM: 89.6GB;

GPU: NVIDIA A100-SXM 40GB.

CUDA version: 12.0

3.1.2. Libraries

The libraries used include tensorflow (Abadi et al., 2015) version 1.13.0, sklearn (Pedregosa et al., 2011) version 1.2.2, matplotlib version 3.7.1 (Caswell et al., 2023) and numpy version 1.23.5.

3.2. Dataset

The dataset used for this research is the malaria Cell Image Dataset sourced from the official NIH website (https://lhncbc.nlm.nih.gov/), but it can also be found on "Kaggle.com". The dataset consists of microscopic images of blood-smears with two classes labelled as either Parasitized or Uninfected as seen in Figure 2 below.

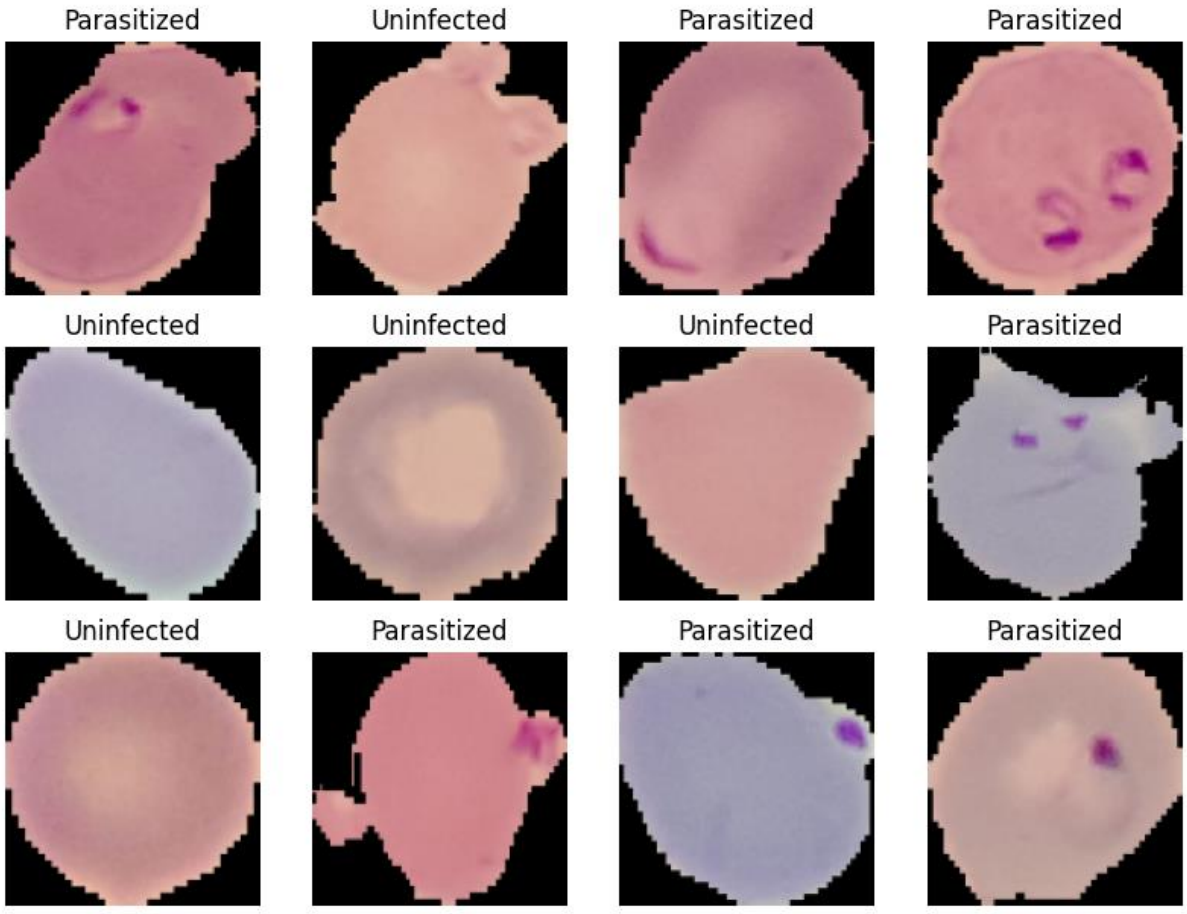

Figure 2. Figure showing samples of the parasitized and Uninfected classes present in the malaria Cell Image Dataset sourced from the official NIH website used for the study (generated from the attached Jupyter notebook).

The dataset has a total of 43,392 images with 27,558 for training and validation and 15,832 images for testing. The full distribution of the dataset can be seen in Table 1 below.

	Parasitized	**Uninfected**	**Total**
Train Data	13,779	13,779	27558
Test Data	7952	7880	15832
Single Prediction	1	1	2
Total	21,732	21,660	43,392

Table 1. Table showing the distribution of the malaria Cell Image Dataset sourced from the official NIH website used for the study.

3.3. Data Preprocessing

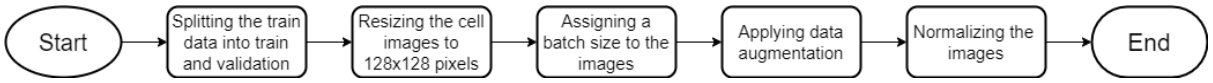

Figure 3. Flow diagram showing the steps taken to preprocess the malaria dataset before introducing them to the different models (drawn using the free drawing tool on draw.io)

The train dataset was already balanced from the onset as seen in Table 1 and so first thing that was done in the preprocessing was to split the train data into train and validation. 80% of the train data was used for training and 20% was used for validation and the train data is still balanced as seen in Figure 4 below.

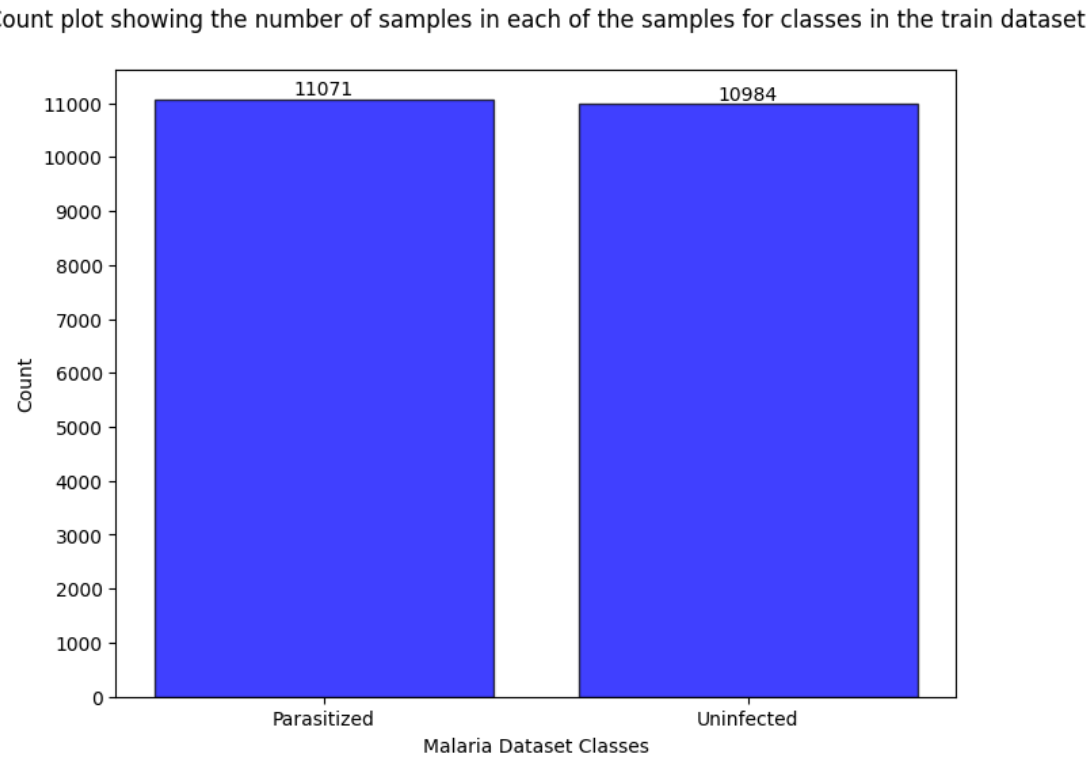

Figure 4. Count Plot showing the distribution of the parasitized and uninfected classes in the dataset for training (generated by the attached Jupyter notebook).

All the images were also resized to 128x128 pixels to firstly to ensure size uniformity because neural networks require images to be of the same size (Hashemi, 2019), and secondly if the images are too large it would cause the CNN would learn very slowly (Saponara and Elhanashi, 2022).

Following this, a batch size of 512 was selected so that the model would perform more weight updates per training epoch leading to a faster training time and a reduced number of epochs required for convergence.

Although the dataset was already large, data augmentation was employed to further improve the performance of the models (Lyashenko, 2022). The augmentation used included Randomly flipping horizontally, randomly rotating by 20% and randomly zooming in by 20%.

The data was then normalized for two different reasons. The first reason is that when the features are all on the same scale (0 to 1), the gradient step would stabilize which implies that higher learning rates can be used without the fear of overshooting the global minimum which potentially leads to faster convergence (Chng, 2022). The second reason is the Support Vector Machine (SVM) which is a used in this research is distance based and so if the images are not normalized, the features would not contribute equally when learning and the features with larger scales would bias the SVM model (Raghav Vashisht, 2021).

3.4. CNN Feature Extraction

As seen in Figure 1, the CNN was used as to extract features which were used to train a custom CNN model, an SVM model and a Random Forest model.

The CNN feature extractor consists of four convolutional layers, with each followed by a max pooling layer and a flatten layer at the end as seen in Figure 5 and Figure 6.

```
_________________________________________________________________
 Layer (type)                Output Shape
=================================================
 conv2d (Conv2D)             (None, 128, 128, 128)

 max_pooling2d (MaxPooling2  (None, 64, 64, 128)
 D)

 conv2d_1 (Conv2D)           (None, 64, 64, 32)

 max_pooling2d_1 (MaxPoolin  (None, 16, 16, 32)
 g2D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)

 max_pooling2d_2 (MaxPoolin  (None, 8, 8, 64)
 g2D)

 conv2d_3 (Conv2D)           (None, 8, 8, 128)

 max_pooling2d_3 (MaxPoolin  (None, 4, 4, 128)
 g2D)

 flatten (Flatten)           (None, 2048)

=================================================
Total params: 132832 (518.88 KB)
Trainable params: 132832 (518.88 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 5. Figure showing the summary of the CNN feature extractor used for the experiment.

From Figure 6 below, we see that the max pooling layers progressively reduced the spatial dimensions of the extracted features (in other words dimensionality reduction) and in the end, the features were flattened to become a 1-dimensional vector. This is relevant because the neural network Dense layers, the Random Forest model and the SVM model are only compatible with 1-dimensional vectors (Lv, Zhang and Wang, 2022).

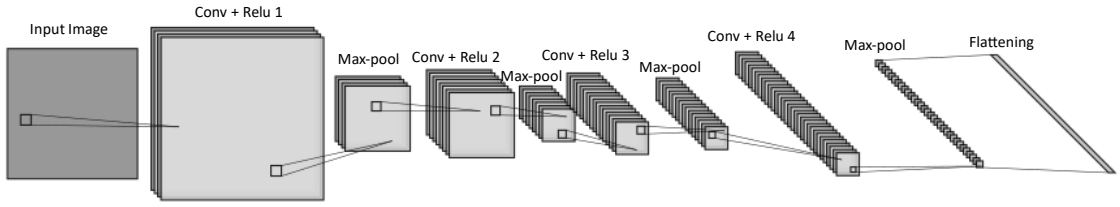

Figure 6. Diagram showing the flow of the CNN feature extractor for the malaria dataset. (Custom drawn with the free online tool on http://alexlenail.me/)

3.5. CNN Custom model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 128)	3584
max_pooling2d (MaxPooling2D)	(None, 64, 64, 128)	0
conv2d_1 (Conv2D)	(None, 64, 64, 32)	36896
max_pooling2d_1 (MaxPooling2D)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 8, 8, 64)	0
conv2d_3 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 512)	1049088
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

Total params: 1182433 (4.51 MB)
Trainable params: 1182433 (4.51 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 7. figure showing the summary custom CNN model after connecting the classification layers to the CNN feature extractor.

Following the CNN Feature extraction, three layers were added on top for the sake of the CNN Custom model. The first was one was a dense layer with 512 neurons and an activation function of relu. This was done to introduce nonlinearity to allow the neural network to develop complex functions based on the inputs (Brodtman, 2021).

The second layer was a dropout layer with a dropout value of 0.5. This layer was added because it set 50 percent of the input units to 0 with a frequency rate at each step during training time and this helped prevent mitigate overfitting while the model was being trained (Vij, 2023).

The third and final layer was the prediction layer with an activation function of sigmoid. The sigmoid activation was chosen because the dataset contained only two classes and sigmoid function converts all raw values to be between (0, 1) as seen in Figure 8 below.

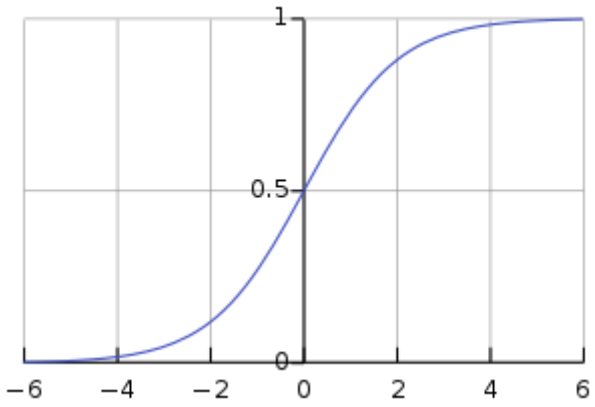

Figure 8. Diagram showing the sigmoid function (Qef, 2008)

The custom CNN model was then trained on the malaria dataset for 30 epochs with the Adaptive Moment Estimation (ADAM) optimizer and a learning rate of 0.001. The model had a training and validation accuracy of 95.41% and 96.20% respectively and training and validation loss of 0.12 and 0.11 respectively as seen in Figure 9.0 and Figure 9.1 below.

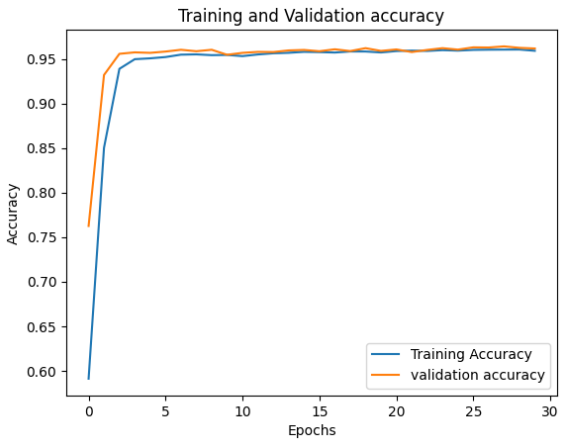

Figure 9.0. Diagram showing the Training and Validation accuracy while training the custom CNN model.

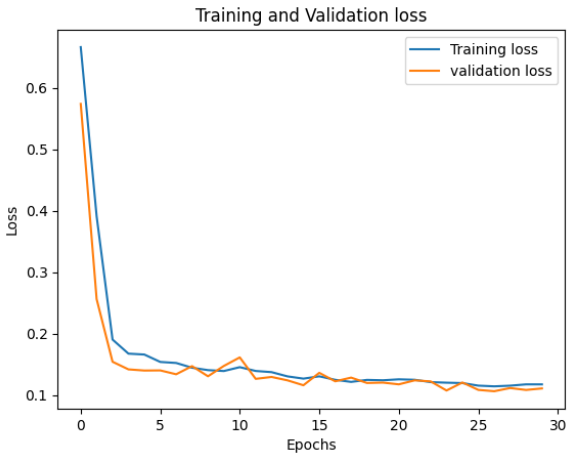

Figure 9.1. Diagram showing the training and validation loss for the custom CNN model.

3.6. CNN-Random Forest and CNN-SVM

The random forest and the SVM classifiers were utilized to make predictions using the features extracted by the CNN feature extractor discussed above. The Figure 10 and Figure 11 below demonstrates how the Random Forest and SVM made predictions on the features that were flattened by the flattening layer of the CNN feature extractor.

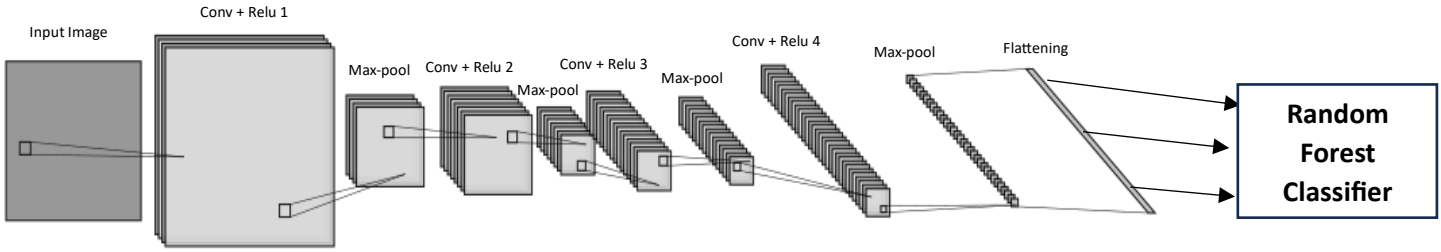

Figure 10. Diagram illustrating the architecture of the CNN-Random Forest classifier.

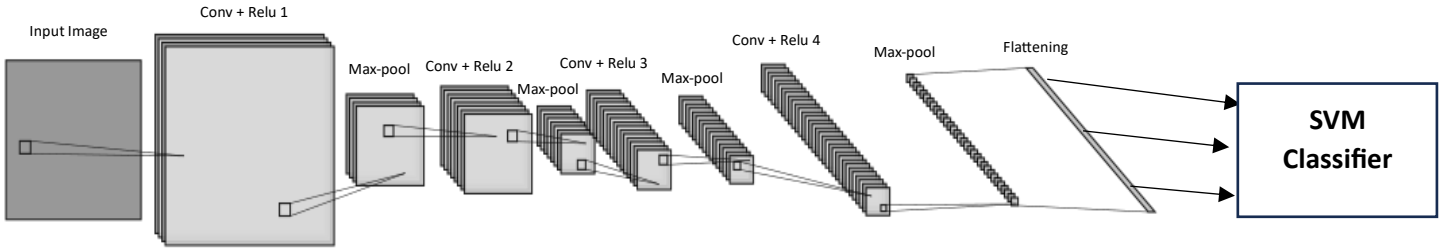

Figure 11. Diagram illustrating the architecture of the SVM classifier.

3.7. Pretrained Models

Several transfer learning models were developed to determine if they performed better than the CNN-Random Forest and the CNN-SVM. Each of the models was first trained with weights from image net before being finetuned on the dataset to for optimal performance.

3.7.1. Inception V3

The inceptionV3 model is a 48-layer deep network that has been trained with millions of images from the ImageNet and has shown accuracies of over 78.1% (Szegedy et al., 2015). The network has a preferred image size of (299 X 299) and has been proven to classify images into 1000 categories.

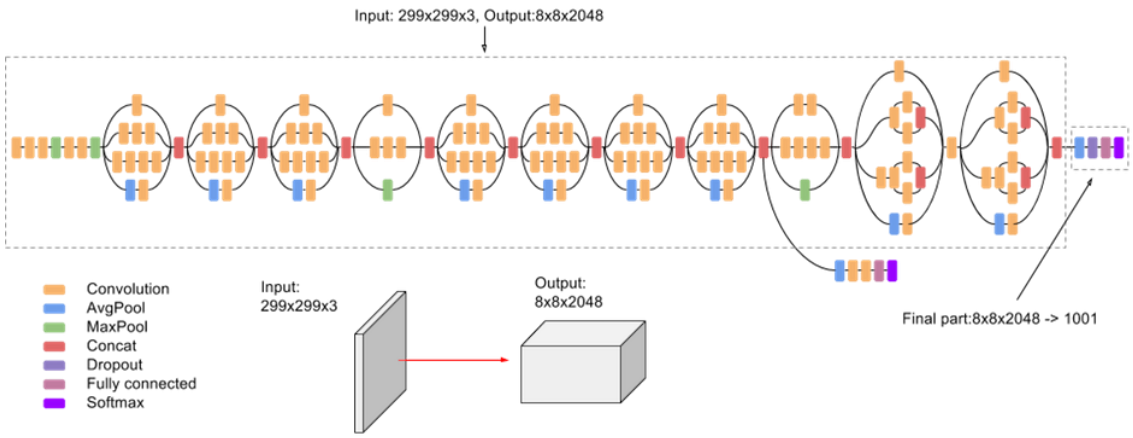

Figure 12. Diagram showing the architecture of the inception v3 model (Szegedy et al., 2015).

For the sake of this research, a custom dropout layer with a dropout of 0.2 and a dense layer with a sigmoid activation function were added on top of the Inception v3 model. The model was then trained on weights from image nets using the binary crossentropy loss function and the ADAM

optimizer and a learning rate of 0.01 for 10 epochs and this yielded a training and validation accuracy of 88.29% and 90.64% respectively and a training and validation loss of 0.2922 and 0.2398 respectively.

After this, the layers were unfrozen and finetuned on the malaria dataset. The model was finetuned on 31 epochs with the first 5 epochs using a learning rate of 1e-6 and the rest using a 1e-5. After fine tuning, the training and validation accuracies increased to 96.53% and 96.59 percent respectively and the training and validation losses dropped to 0.0925 and 0.0882 respectively. The entire training and validation loss and accuracy before and after finetuning seen in Figure 13 below.

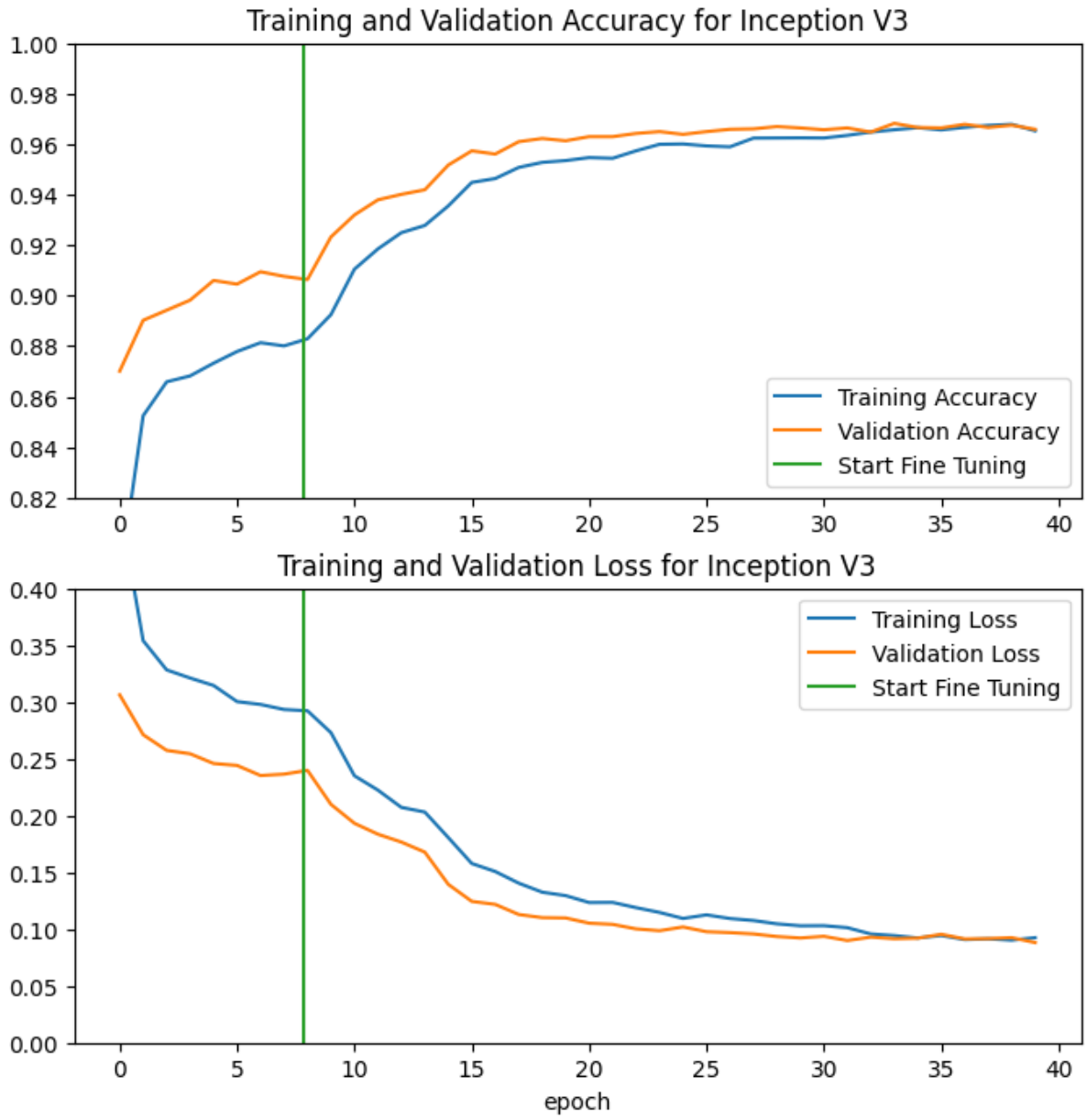

Figure 13. Plots showing the training and validation accuracies and losses for the entire training process of the InceptionvV3 model.

3.7.2. ResNet101V2 model

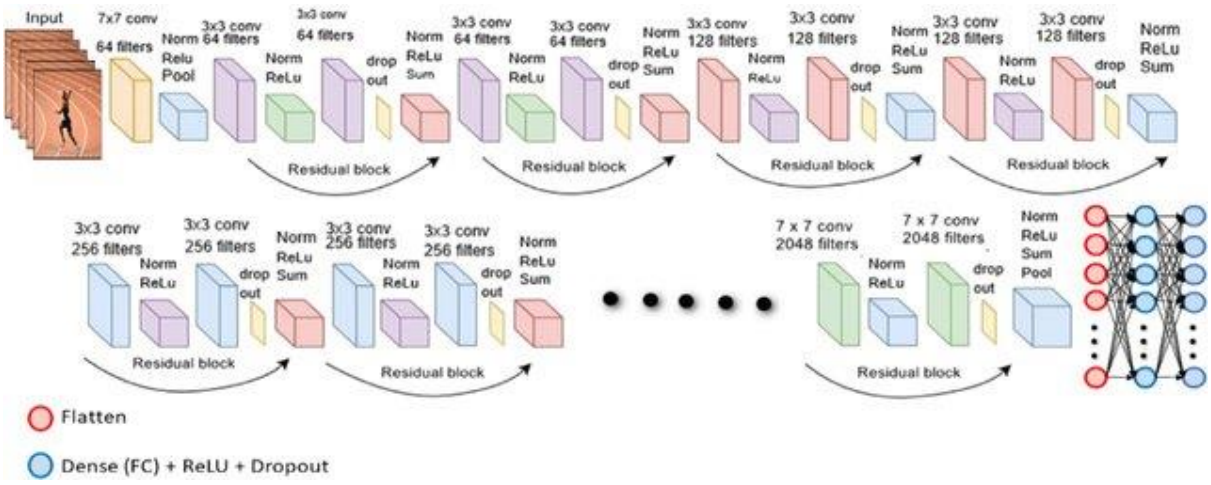

Figure 14. Architecture of the ResNet101V2 model (Bilal et al., 2021)

A dropout with a dropout value of 0.2 and a sigmoid layer were added on top of the last layer in the ResNet101V2 model. Then the layers were compiled with a the binary crossentropy loss, accuracy metric and the ADAM optimizer with a learning rate of 0.01 before being frozen and trained on weights from image for a duration of 10 epochs. This gave a training and validation accuracy of 90.70% and 92.56% respectively and a training and validation loss of 0.2419 and 0.1904 respectively.

After this, the model was finetuned on the data set by training for a duration of 30 epochs with a learning rate of 1e-6 for the first 20 epochs and 1e-5 for the rest. This increased the training and validation accuracies to 96.81% and 97.04 percent respectively and the training and validation losses dropped to 0.0897 and 0.0840 respectively. The entire training and validation loss and accuracy before and after finetuning seen in Figure 15 below.

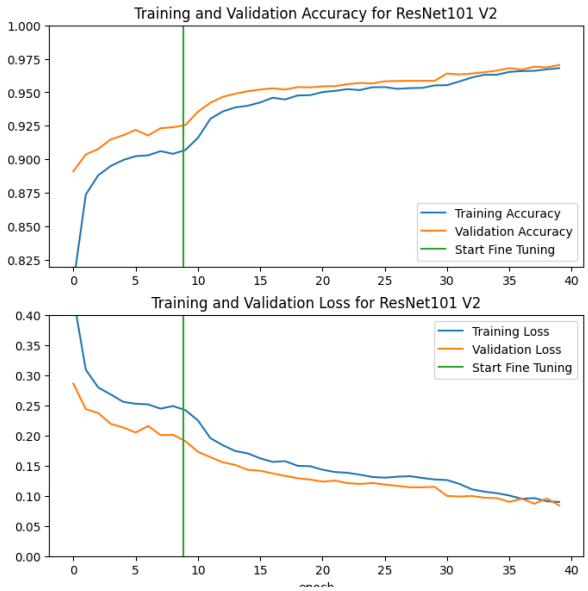

Figure 15. Plots showing the training and validation accuracies and losses for the entire training process of the ResNet101V2 model.

3.7.3. VGG 16

The VGG is a CNN architecture that makes use of convolution layers with a 3 X 3 filter and a stride of which are also in the same padding and max pooling layer of 2 X 2 filter of stride 2. This pattern remains the same throughout the entire architecture and has two fully connect layers at the end. The model is said to contain approximately 138 million parameters and the only difference between it and the VGG 19 is that the VGG 16 has 16 layers with weight while the latter has 19 layers with weights (Bilal et al., 2021).

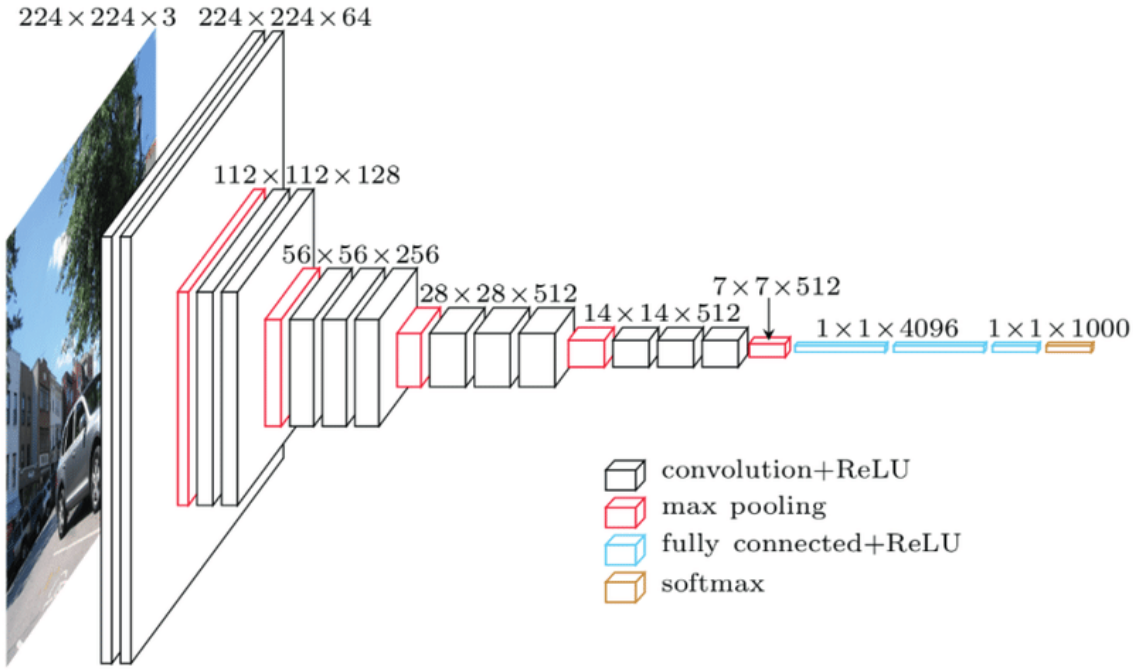

Figure 16. Architecture of the VGG16 and the VGG19 models (Bilal et al., 2021)

The layers of the VGG 16 and the VGG 19 models were frozen, and they were both trained for a duration of 10 epochs initially with the same binary crossentropy loss function, ADAM optimizer and the same accuracy metrics with weights from ImageNet. The VGG 16 had a training and validation accuracy of 80.57% and 80.26% respectively and had a training and validation loss of 0.468 and 0.4399 respectively, while, the VGG19 had a training and validation accuracy of 78.31% and 79.40% respectively and had a training and validation loss of 0.4972 and 0.4746 respectively.

After finetuning by unfreezing the layers and compiling with a learning rate of 1e-6 for the first 10 epochs and 1e-5 for the rest, the models were finetuned for 30 epochs and the VGG 16 had an improved training and validation accuracy of 96.55% and 96.79% respectively and had a training and validation loss of 0.0913 and 0.0909 respectively, while, the VGG19 had a training and validation accuracy of 96.66% and 96.37% respectively and had a training and validation loss of 0.09808 and 0.9624 respectively.

The plots of the training and validation accuracy and validation for the VGG16 and the VGG19 models can be seen below.

Figure 17. Plots showing the training and validation accuracies and losses for the entire training process of the VGG16 model.

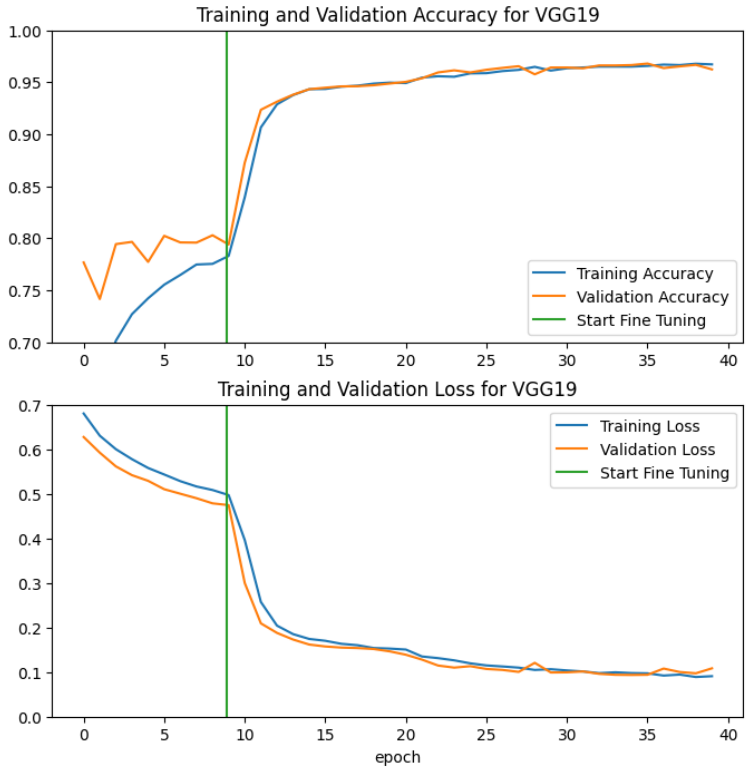

Figure 18. Plots showing the training and validation accuracies and losses for the entire training process of the VGG19 model.

3.8. Evaluation Metrics

The following evaluation metrics were used to determine the best performing model in detecting malaria on the dataset.

True positive (TP) is the model correctly predicts that blood sample is infected with malaria.

True Negative (TN) is the models correctly predict a blood sample is uninfected with malaria.

False Positive (FP) is when the models incorrectly predict that a blood sample is infected with malaria.

False Negative (FN) is when the model incorrectly predicts that a blood sample is not infected with malaria.

Accuracy: This is the overall description of how well a model predicts a both classes in the dataset, it is defined by equation (1)

Recall: This is the TP rate, and it is given by the equation (2) below

Specificity: This is the TN rate, and it is given by the equation (3) below

Precision: This is the proportion of predicted cases of malaria that are malaria

Accuracy $$\frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

Recall $$\frac{TP}{TP + FN} \quad (2)$$

Specificity $$\frac{TN}{TN + FP} \quad (3)$$

Precision $$\frac{TP}{TP + FP} \quad (4)$$

F1-Score $$\frac{precision\ x\ recall}{Precision + recall} \quad (5)$$

4.0 Evaluation and Discussion

From comparing the accuracy, recall, precision, and F1-Score of all the models, we determined that all the pretrained models used for the experiment outperformed the custom CNN model, but the hybrid of the CNN-Random Forest is the best performing model with an accuracy of 97.34% and is closely followed by the ResNet101V2 which has an accuracy of 97.11% as seen in Table 2.

Also from looking at the confusion Matrixes from Figures 19 to 24, it can be deduced that all the models are more likely to predict that a person is not infected and is infected (False Negative) than they are to predict a person is infected with malaria and they are not (False positive), and this is bad because it implies that the model is more likely to miss out on cases of malaria even though the overall accuracy is high.

	Accuracy (%)	F1-score (%)	Recall (%)	Precision (%)
CNN	95.84	95.85	95.85	95.92
CNN-SVM	92.34	92.32	92.36	92.76
CNN-RF	97.34	97.38	97.34	97.35
InceptionV3	97.01	97.01	97.02	97.03
ResNet101V2	97.11	97.11	97.11	97.12
VGG-16	96.26	96.26	96.27	96.32
VGG-19	95.36	95.36	95.38	95.59

Table 2. Table showing the average metrics of both the parasitized and the uninfected classes.

	F1-score (%)	Recall (%)	Precision (%)
CNN	95.76	93.62	98.00
CNN-SVM	91.97	87.34	97.12
CNN-RF	97.34	96.71	97.97
InceptionV3	97.00	96.04	97.97
ResNet101V2	97.09	96.09	98.11
VGG-16	98.21	94.37	96.12
VGG-19	95.21	91.78	98.92

Table 3. Table showing the metrics for the parasitized class.

	F1-score (%)	Recall (%)	Precision (%)
CNN	95.91	98.07	93.85
CNN-SVM	92.68	97.39	88.40
CNN-RF	97.35	97.98	96.72
InceptionV3	97.03	97.99	96.08
ResNet101V2	97.12	98.13	96.13
VGG-16	96.31	98.17	94.53
VGG-19	95.51	98.98	92.26

Table 4. Table showing the metrics uninfected classes.

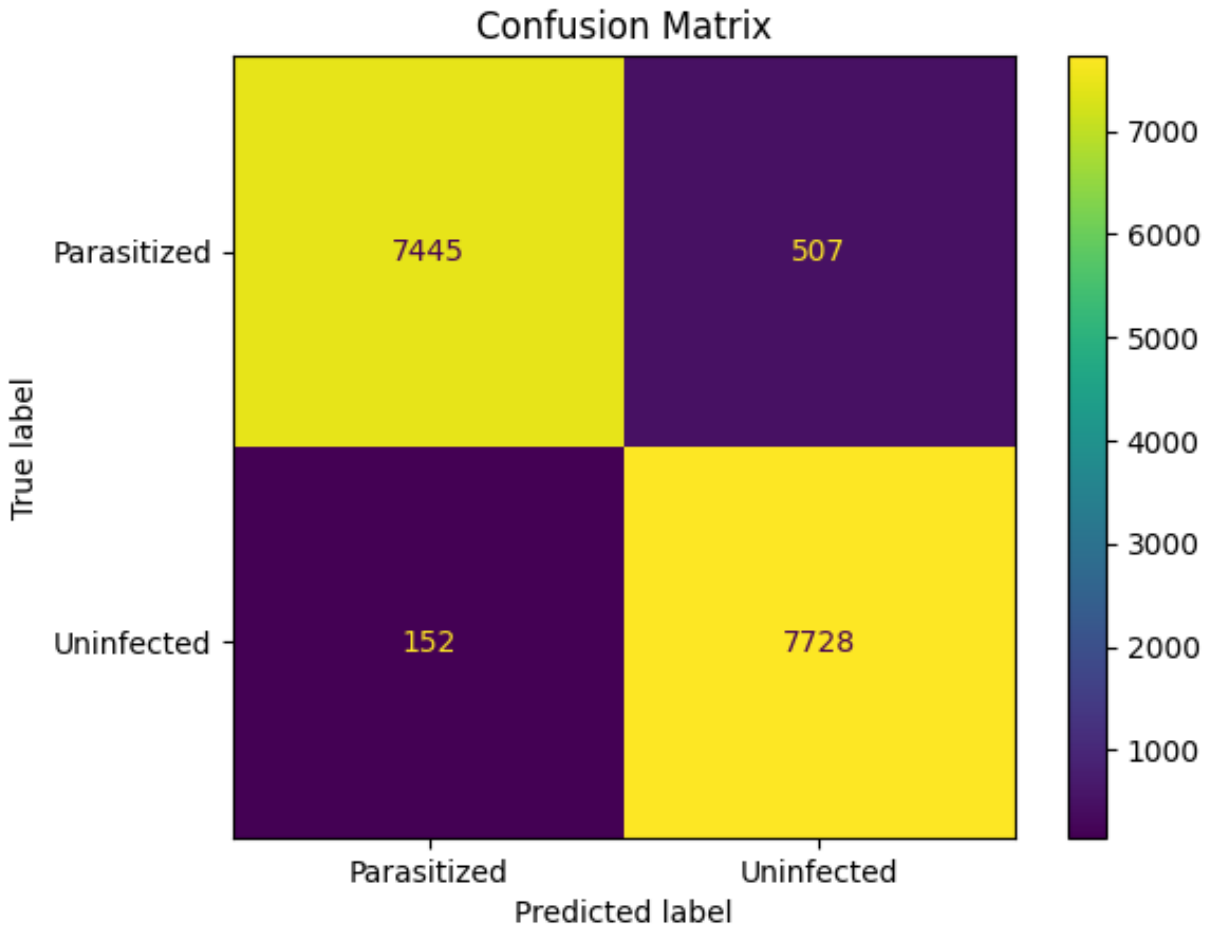

Figure 19Confusion matrix for the custom CNN on the malaria dataset

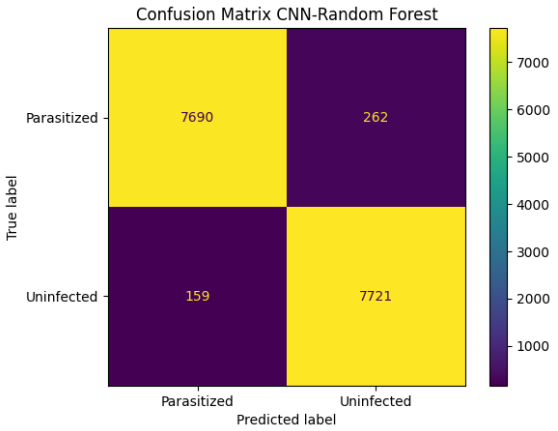

Figure 23 Confusion matrix for the CNN-RF classifier on the malaria dataset

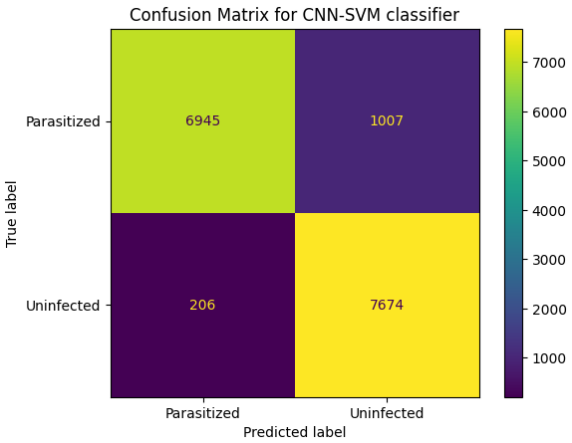

Figure 22 Confusion matrix for the CNN-SVM classifier on the malaria dataset

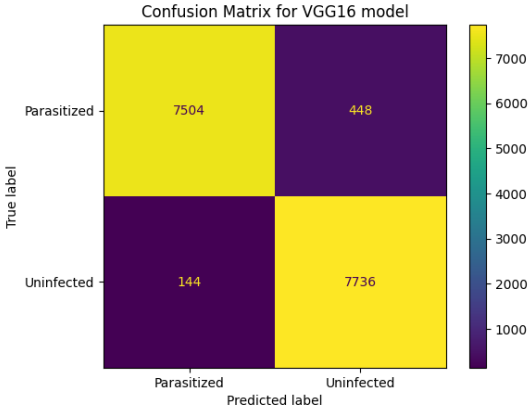

Figure 25. Confusion matrix for the VGG16 classifier on the malaria dataset

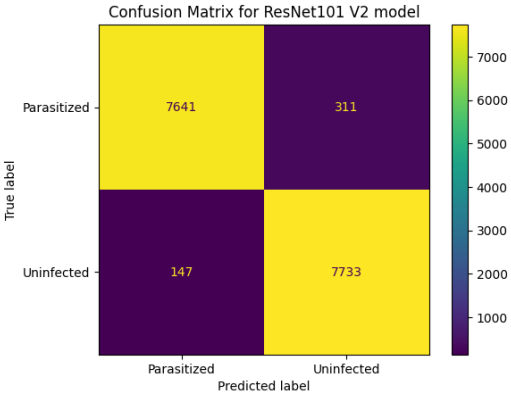

Figure 24. Confusion matrix for the ResNet101V2 model on the malaria dataset

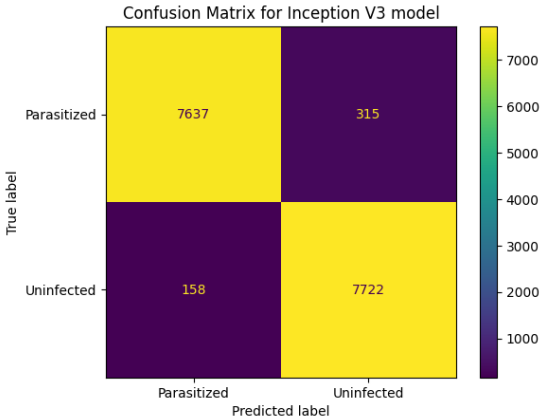

Figure 21. Confusion matrix for the InceptionV3 model on the malaria dataset

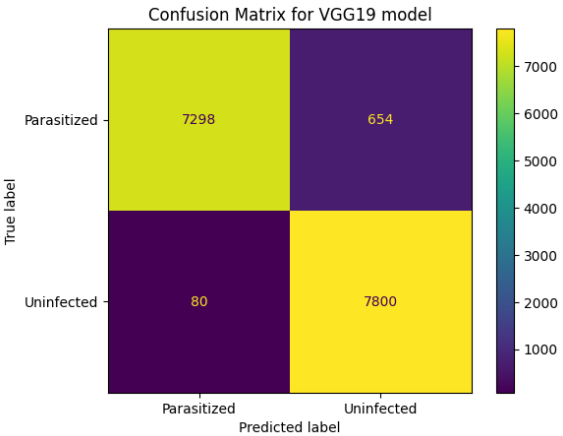

Figure 20. Confusion matrix for the VGG19 classifier on the malaria dataset

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M. and Levenberg, J. (2015) *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Available online: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf.

Alnussairi, M.H.D. and İbrahim, A.A. (2022) Malaria parasite detection using deep learning algorithms based on (CNNs) technique. *Computers and Electrical Engineering*, 103, 108316.

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M. and Farhan, L. (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, [online] 8(1). Available online: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8.

Bilal, M., Maqsood, M., Yasmin, S., Hasan, N.U. and Rho, S. (2021) A transfer learning-based efficient spatiotemporal human action recognition framework for long and overlapping action classes. *The Journal of Supercomputing*.

Brodtman, Z. (2021) *The Importance and Reasoning behind Activation Functions*. Medium. Available online: https://towardsdatascience.com/the-importance-and-reasoning-behind-activation-functions-4dc00e74db41.

Caswell, T.A., Lee, A., Sales De Andrade, E., Droettboom, M., Hoffmann, T., Klymak, J., Hunter, J., Firing, E., Stansby, D., Varoquaux, N., Hedegaard Nielsen, J., Root, B., May, R., Gustafsson, O., Elson, P., Seppänen, J.K., Lee, J.-J., Dale, D., Hannah and McDougall, D. (2023) matplotlib/matplotlib: REL: v3.7.1. *Zenodo*. [online] Available online: https://ui.adsabs.harvard.edu/abs/2023zndo...7697899C/abstract [Accessed 11 Sep. 2023].

Chng, Z.M. (2022) *Using Normalization Layers to Improve Deep Learning Models*. MachineLearningMastery.com. Available online: https://machinelearningmastery.com/using-normalization-layers-to-improve-deep-learning-models/#:~:text=Normalization%20can%20help%20training%20of.

De Man, R., Gang, G.J., Li, X. and Wang, G. (2019) Comparison of deep learning and human observer performance for detection and characterization of simulated lesions. *Journal of Medical Imaging*, 6(02), 1.

Hashemi, M. (2019) Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation. *Journal of Big Data*, 6(1).

Hemachandran, K., Alasiry, A., Marzougui, M., Ganie, S.M., Pise, A.A., Alouane, M.T.-H. and Chola, C. (2023) Performance Analysis of Deep Learning Algorithms in Diagnosis of Malaria Disease. *Diagnostics*, [online] 13(3), 534. Available online: https://www.mdpi.com/2075-4418/13/3/534 [Accessed 10 Sep. 2023].

Khaled Almezhghwi (2022) Malaria Detection Using Convolutional Neural Network. *Springer eBooks*, 116–123.

Kumar, A., Sarkar, S. and Pradhan, C. (2019) Malaria Disease Detection Using CNN Technique with SGD, RMSprop and ADAM Optimizers. *Studies in Big Data*, 211–230.

Kumar, S., Priya, S. and Kumar, A. (2023) *Malaria detection using Deep Convolution Neural Network*. Available online: https://arxiv.org/pdf/2303.03397.pdf.

Lv, Q., Zhang, S. and Wang, Y. (2022) Deep Learning Model of Image Classification Using Machine Learning. *Advances in Multimedia*, 2022, 1–12.

Lyashenko, V. (2022) *Data Augmentation in Python: Everything You Need to Know*. neptune.ai. Available online: https://neptune.ai/blog/data-augmentation-in-python#:~:text=In%20general%2C%20having%20a%20large [Accessed 11 Sep. 2023].

Mayo Clinic (2023) *Malaria - symptoms and causes*. Mayo Clinic. Available online: https://www.mayoclinic.org/diseases-conditions/malaria/symptoms-causes/syc-20351184.

Mbanefo, A. and Kumar, N. (2020) Evaluation of Malaria Diagnostic Methods as a Key for Successful Control and Elimination Programs. *Tropical Medicine and Infectious Disease*, [online] 5(2). Available online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7344938/.

Nema, S., Rahi, M., Sharma, A. and Bharti, P.K. (2022) Strengthening malaria microscopy using artificial intelligence-based approaches in India. *The Lancet Regional Health - Southeast Asia*, 5, 100054.

Pedregosa, F., Pedregosa@inria, F., Fr, Org, G., Michel, V., Fr, B., Grisel, O., Grisel@ensta, O., Blondel, M., Prettenhofer, P., Weiss, R., Com, V., Vanderplas, J., Com, A., Cournapeau, D., Varoquaux, G., Gramfort, A., Thirion, B., Dubourg, V. and Passos, A. (2011) Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos

PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot Edouard Duchesnay. *Journal of Machine Learning Research*, [online] 12, 2825–2830. Available online: https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf.

Qef (2008) *The logistic sigmoid function.* Wikimedia Commons. Available online: https://commons.wikimedia.org/wiki/File:Logistic-curve.svg?uselang=en#Licensing [Accessed 11 Sep. 2023].

Raghav Vashisht (2021) *Machine Learning: When to perform a Feature Scaling? - Atoti Community*. Atoti Community. Available online: https://atoti.io/articles/when-to-perform-a-feature-scaling/#:~:text=Distance%2DBased%20Algorithms&text=This%20is%20because%20behind%20the.

Rees-Channer, R.R., Bachman, C., Grignard, L., Gatton, M.L., Burkot, S., Horning, M.P., Delahunt, C.B., Hu, L., Courosh Mehanian, Thompson, C.M., Woods, K., Lansdell, P., Shah, S. and Chiodini, P.L. (2023) Evaluation of an automated microscope using machine learning for the detection of malaria in travelers returned to the UK. 1.

Saponara, S. and Elhanashi, A. (2022) Impact of Image Resizing on Deep Learning Detectors for Training Time and Model Performance. *Lecture Notes in Electrical Engineering*, 866, 10–17.

Siłka, W., Wieczorek, M., Siłka, J. and Woźniak, M. (2023) Malaria Detection Using Advanced Deep Learning Architecture. *Sensors*, 23(3), 1501.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2015) *Rethinking the Inception Architecture for Computer Vision*. arXiv.org. Available online: https://arxiv.org/abs/1512.00567.

Team, G.L. (2021) *What is VGG16 - Convolutional Network for Classification and Detection*. GreatLearning Blog: Free Resources what Matters to shape your Career! Available online: https://www.mygreatlearning.com/blog/introduction-to-vgg16/.

Vij, R. (2023) *Combating Overfitting with Dropout Regularization*. Medium. Available online: https://towardsdatascience.com/combating-overfitting-with-dropout-regularization-f721e8712fbe#:~:text=Let [Accessed 11 Sep. 2023].

Vijayalakshmi A and Rajesh Kanna B (2019) Deep learning approach to detect malaria from microscopic images. *Multimedia Tools and Applications*.

World Health Organization (2020) *The Top 10 Causes of Death*. World Health Organization. Available online: https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death.