Ensemble Convolution Neural Networks for Malaria Cell Detection using Transfer Learning.

Student ID: 36004026
Dept. of Computer Science and Communication
Lancaster University
Lancaster, England

***Abstract*—Aim of this project is to develop methods using deep learning models to automate the process in medical diagnostics. Using convolutional neural network (CNN) models, we demonstrated how fast and accurate it is to develop models using transfer learning, in this case for the diagnosis of malaria from microscopical blood smear images. EfficientNet and VGG19 are the two architectures discussed in the paper, and how they can be ensembled to produce better predicting models. By stacking these deep learning-based models, it is possible to detect malarial parasites with an accuracy of 95% from microscopic images. A "Flask" based application is developed to demonstrate how these models can be used in practice.**

***Index Terms*—Deep Learning, CNN, Transfer Learning, EfficientNet, VGG19**

I. Introduction

The use of imaging is vital to the diagnosis of many medical conditions. Human analysis is typically required for these conditions, so the diagnosis may be incorrect. To identify diseases more accurately and reliably, this process can be automated. By applying deep neural networks to blood smear images, we can attempt to detect malaria and identify parasitized cells using a computational approach. Traditionally, convolutional neural networks are trained on image data to make predictions, but they may be limited in their ability to quickly adapt to making predictions on images for completely different applications [10]. This creates algorithms that are much less generalizable since models trained on particular kinds of medical images are less likely to work well for diagnosing other diseases. We attempt to overcome this problem by the use of Transfer Learning, where previously built neural network model architectures with barely any modifications, are used to create newer more flexible models.

This paper looks at the use of models based on transfer learning, specifically using EfficientNet and VGG19 architectures, which when subsequently trained on ImageNet data, yielded high performing individual models (base models) with high rates of out-of-box accuracy. These kinds of models have previously been used for various medical image classification problems, but little research has been done to determine whether ensembles of these base models could lead to more accurate final models. The paper also attempts to analyse the effects of different types of ensemble techniques on the accuracy scores of the eventual models. Bagging and stacking ensemble models in particular are used to make comparisons between the ensemble and individual models. To get a complete understanding of their performance, we compare the models using a variety of evaluation metrics rather than just the accuracy scores. Furthermore, the paper attempts to simulate real-world applications using Flask in the background in order to deploy the models for making predictions [11].

II. Literature Review

Based on [2], ensemble techniques can enhance accuracy and develop more robust models by combining multiple machine learning techniques. The concept can be applied specifically to the detection of malaria cells using ensembles of convolutional neural networks for overcoming limitations and deficiencies of individual models.

According to [3], by analysing chest x-rays, convolutional networks can be used to detect if a person is infected with COVID-19. Convolution based architecture is used for image classification, and furthermore, they use a new CNN-based architecture called ECOVNet, which takes advantage of the powerful EfficientNet family of CNN algorithms. The high rate of accuracy that pre-trained EfficientNet tends to provide means it is beneficial in the initial extraction of features as well as using model snapshots for detecting COVID in images. A class activation map is also used to find critical regions using x-rays and better inform predictions.

A large-scale analysis of the effect of the depth of convolutional neural networks on accuracy was undertaken in [4]. By adding layers to a model, they find that better performance can be achieved by gradually increasing the size of the network. ConvNet is used to train and test the available image data, using more and more dense networks, and small (3x3) filters in the layers. The result is a more powerful and generalizable architecture.

As the need for high-performance learning models using more easily accessible data emerges, [5] and [1] takes a look at the development and potential applications of the concept of transfer learning. As well as examining the concept of transfer learning, they discuss multiple scenarios where it would be necessary to adopt learnable solutions from one model to another. Furthermore, they discuss a wide range of learning solutions for homogeneous and heterogeneous transfer.

The aim of [6] is to examine deep learning solutions for training models to analyze blood smear images. In the study, they used Convolutional Neural Networks, as well as Autoencoders, augmenting the data and eventually classifying using Support Vector Machines (SVMs) or K-nearest neighbours in order to detect Malaria parasites from blood smears. Compared to several other models, the autoencoder-based training method provided the best results, and a validation procedure was conducted using mobile phones with different amounts of processing power.

In addition, researchers from [7] work on detecting malaria parasites from stained blood samples and develop a deep neural network model to identify the various stages of the blood smear images. Using a Deep transfer graph NN, and "transferring" knowledge that the model learns from original images, makes for a good model for detecting the presence of malaria parasites. First, learning the convolutional features using a ResNet model using transfer graph-building learning, and using a K-means supervised algorithm on the final graph representations. Moreover, when tested on other malaria parasite image datasets, the multi-stage process yielded good results, and it even generalized well to another malaria like parasite, Babesia.

III. Methodology

A. Data

This dataset is taken from the official NIH Website. And downloaded from Kaggle. The data consists of 27.6K files having two classes of Uninfected and Parasitized malaria cell images. With each class consisting of 13.8K images.

B. Pre-processing

Pre-processing is an important stage in machine learning. In this stage, we ensure the data is in the right format that is numeric before training models on it. But it is not just associated with having the data in the right format but also transforming the data according to the application. Scaling, normalizing splitting data are a few examples for processing.

When it comes to datasets containing images it is a good practice to store the images according to their respective classes. Segregating the data this way helps to later use pre-processing packages to directly ingest data. This is the reason why the data was organized as two folders consisting of parasitized cells and uninfected cells. To make sure all the images are of the same size we need to rescale images. Here we check the average dimensions of the images which are height and width, based on the dimension we choose the rescaling dimensions. Hence the images were rescaled to dimensions of 130 and 130. Since the images are coloured they have a third dimension which denotes the 'number of channels' which is 3 as each image has colours in RGB (red, green, blue). This gives the final image shape which will be 130 in height, 130 in width and 3 colour channels (130,130,3). Once the shape is decided the images are then converted to image arrays which are numeric, consisting of the colour information of all the pixels in the image. Each of these pixel values can take a value within the range 0-255, a specific value in this range tells if that pixel is red, green, or blue. Once all the pixel values are obtained they need to be normalized to bring them in the range of 0 to 1. But the data had images which very normalized. Hence there was no need to normalize.

For augmenting the data as above Keras library 'ImageDataGenerator' was used. It provides a host of different augmentation techniques like standardization, rotation, shifts, flips, brightness change, and many more. Only standardization was implemented as there was no need to create more images since the data was relatively huge and had lots of different sorts of images. Because of having a variety of images no artificial augmentation was required. The data was then split into training and testing sets with 80% of images used for training, 10% images for validation and 10% for testing. The splitting decision was taken based solely on the intention to provide maximum data to the model. The splitting was done by the 'flow_from_directory' function in 'ImageDataGenerator'. The image data was grouped in batches for 32 in both training and testing sets.

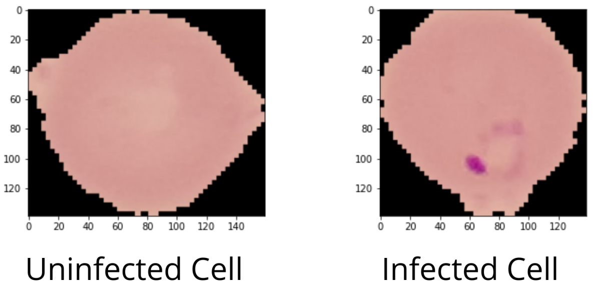


Fig. 1. Cell Images

Figure 1 shows how the infected and uninfected cell look.

C. Models

Since the data has images, convolution neural networks were used to build the models. Here the aim is to see if the ensembles perform better than the individual models. The experiments were done in regard to ensembling configuration. Two ensembling configurations were evaluated which are stacking and bagging. The reason for conducting these experiments was to see if ensembling results in better predicting and does implementing different ensembling types results in different or comparable results. There were four models generated. First, the individual models were tested, then the stacked ensemble, and finally the bagging ensemble model.

For the training of the models two base networks, Efficient Net and VGG19 are used. The outputs are then flattened and provided to two dense layers. The output layer has 1 neuron with activation function as sigmoid as this is a two-class classification. The optimisers and loss functions were chosen using grid search. For losses, binary cross-entropy and hinge loss was taken into consideration as they seem to be a logical choice for the problem. Three optimizers Adam, RMSprop, and SGD were taken. For the first model, with the efficient net as the base model binary cross-entropy with RMSprop gave the best results. For the second mode the binary cross-entropy with RMSprop was the best fitting but when training, was a bit unstable hence adam optimizer was chosen.

The technique to build models is called transfer learning. The Efficient Net and VGG19 are the state of the art CNN architectures. These two models act as individual models.

Let's discuss how transfer learning works and how the individual models were built.

Transfer Learning

Deep learning algorithms and machine learning algorithms work in isolation meaning they are made for a specific task. Every time the distribution of feature spaces changes, the models must be rebuilt from scratch. In transfer learning, knowledge acquired for one task is used to solve related tasks instead of being isolated in a learning paradigm. Therefore, transfer learning refers to the reuse of a model or knowledge for something else. [8] In predictive modelling problems involving image data, transfer learning is common. EfficientNet and VGG19 are used in this case. Using this approach makes sense since the images were trained with a large corpus of images, and the model has to predict a wide range of classes, which means the model must learn to extract features efficiently in order to be successful on the problem. A few advantages include improved baseline performance, reduced modelling time, improved final performance, ease to re-use and tune, and can be used as a base model.

EfficientNet

The EfficientNet architecture employs a compound coefficient to scale all dimensions of depth, width, and resolution uniformly. As opposed to conventional scaling, which scales network width, depth, and resolution arbitrary, EfficientNet uniformly scales them using a set of fixed scaling coefficients. As the input image grows bigger, the network requires more layers to increase the receptive field and more channels to capture the finer patterns. By reducing parameters and Floating-Point Operations Per Second, EfficientNet not only improves accuracy but also improves the efficiency of the models.

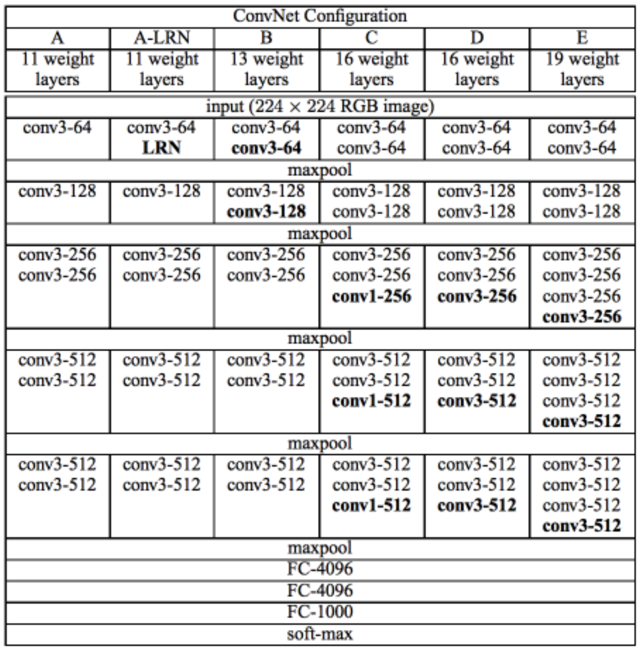
This network is based on MBConv, which is optimized through squeeze-and-excitation. Similar to MobileNet v2, MBConv uses inverted residual blocks. They connect the beginning and end of a convolutional block. As a first step, the input activation maps are expanded with 1x1 convolutions to deepen the feature maps. There are then three 3-by-3 depth-wise and 3-by-3 point-wise convolutions, which reduce the number of channels in the feature map. Shortcut connections connect the narrow layers, whereas skip connections connect the wider layers. In addition to reducing the overall number of operations needed, this structure also helps to reduce. [9]

Architecture of EfficientNet:

conv, MBconv (3x3), MBconv(3x3), MBconv(3x3), MBconv(5x5), MBconv(5x5), MBconv(3x3), MBconv(3x3), MBconv(3x3), MBconv(5x5), MBconv(5x5), MBconv(5x5), MBconv(5x5), MBconv(5x5), MBconv(5x5), MBconv(5x5), MBconv (3x3)

VGG19

The VGG19 architecture shown in figure 2, column E, has 16 layers of CNNs, three fully connected layers, and a final layer for softmax. Each network architecture will have a fully connected layer and a final layer.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig. 2. VGG19 Architecture. Source: [4]

A: Have 8 CNN layers so a total of 11 layers including the fully connected layers (FC). A-LRN: Column B is also similar to column A, but has an additional step called Local response normalization (LRN) that allows for lateral inhibition in the layer, which means that it creates a local maximum that increases sensory perception, which we prefer in CNNs. However, in this specific case, ILSVRC, the network trained slower and accuracy was not increasing. C: The CNN layers are 13 here, and the FC layers are 16. As a result of using a convolution filter of (1 * 1), the authors have improved discrimination. B and D: These columns offer additional CNN layers and consist of 13 layers and 16 layers, respectively.

The layers in the VGG19 model are as follows: Conv3x3 (64), Conv3x3 (64), MaxPool, Conv3x3 (128), Conv3x3 (128), MaxPool, Conv3x3 (256), Conv3x3 (256), Conv3x3 (256), Conv3x3 (256), MaxPool, Conv3x3 (512), Conv3x3 (512), Conv3x3 (512), Conv3x3 (512), MaxPool, Conv3x3 (512), Conv3x3 (512), Conv3x3 (512), Conv3x3 (512), MaxPool, Fully Connected (4096), Fully Connected (4096), Fully, Connected (1000), SoftMax

Ensembling

To build ensemble models the two base models built in the previous sections are used. Each of the base models is trained and then they are either set up in bagging configurations or stacking configurations.

For bagging same model was fit multiple times and the predictions were stored. Once the models were fitted the average of the predictions from the models was calculated. Here bagging is only considered for the base model that predicted well individually. EfficientNew base model was better at predicting hence the bagging was done using EfficientNet as the base model.

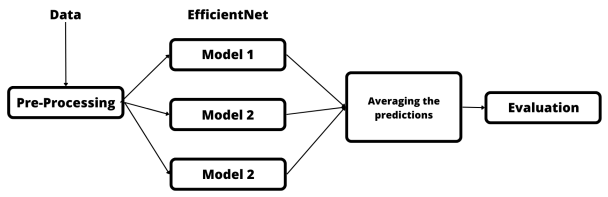


Fig. 3. Bagging Ensemble

The figure 3 explains how bagging works in this case.

Stacking works the same way as bagging but here instead of using the same base model we use a combination (mixing) of two or more different base models. Here Efficient Net and VGG19 are combined to create a stacking model and the predictions are made by the stacking model. This is the main difference between bagging and stacking. To create the stacking ensemble model Keras was used which combines the two models and calculates the predictions as an average.

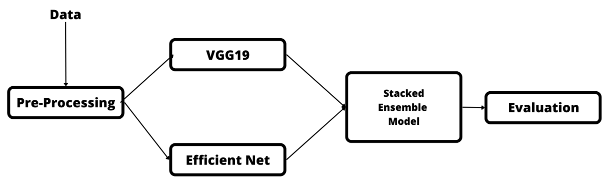


Fig. 4. Stacked Ensemble

Figure 4 represents a block diagram used in the paper for stacking models.

Flask Implementation

To display how the model would work, a simulations app was created using “Flask”. In this app, the training models were saved while modelling. For the application, just the best performing models were taken. Here the user of the app can select an image to predict. The image is then pre-processed, and given to the model to predict. The model predicts and returns a class. Based on the class a message is given out to the user in the text indicating where the cell is uninfected or parasitized. The block diagram is given in figure 5 The app runs on a browser in a locally and python code needs to be executed. Since this is just built to test an executable app was not built. However, this can be used in a similar fashion for use in real life. This can significantly help doctors or health professionals analyse and diagnose the patients.

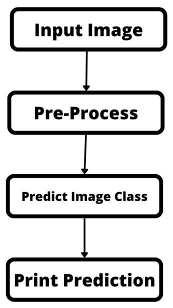


Fig. 5. Flask Implementation

IV. Results

A. Evaluation of Classification Models

To evaluate how the models are performing different metrics were taken into consideration. Accuracy was taken as primary metrics to gauge the performance of the model. The classification report gives the information about the accuracy, precision, recall, and f1_scores.

Accuracy: A model’s accuracy is determined by the number of correctly classified data over its total number of classifications.

$Accuracy = \frac{Numberofcorrectpredictions}{Totalnumberofprediction}$

Precision: Machine learning models are often judged by their precision - the quality of the positive predictions they make. The precision is calculated by given formula

$Precision = \frac{TruePositives}{TruePositives+FalsePositives}$

Recall: The recall metric measures how many correct predictions have been made out of all possible positive predictions.

$Recall = \frac{TruePositives}{TruePositives+FalseNegative}$

F1-score: A model’s F-score, also called an F1-score, measures the accuracy of that model over a dataset. The F1-score is the harmonic mean of precision and recall.

$F1-score = \frac{TruePositives}{TruePositives+\frac{1}{2}(FalsePositive+FalseNegative)}$

ROC curve: A receiver operating characteristic curve, or ROC curve, illustrates a binary classifier system's diagnostic ability as the threshold for discrimination is varied.

Confusion Matrix: The confusion matrix describes a classification model's performance on a set of test data for which the true values are known. It displays the values of the true positives, true negatives, false positives and false negative in a tabular format.

All of the data was evaluated on unseen holdout data which was not seen by the model during training phase. This holdout data accounts for 10 percent of the total data.

Below are the results of all the models.

B. EfficientNet Model

The performance of the EfficientNet Model is given in the table I below.

TABLE I
PERFORMANCE METRICS

Metric	EfficientNet
Accuracy	0.93
Precision	Parasitised: 0.98 and Uninfected: 0.89
Recall	Parasitised: 0.89 and Uninfected: 0.98
F1 Score	Parasitised: 0.93 and Uninfected: 0.94

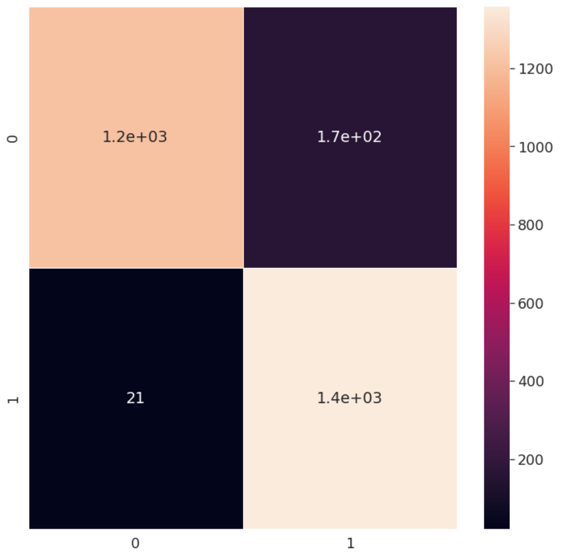


Fig. 6. Confustion Matrix of EfficientNet Model

From the confusion matrix in figure 6 and the classification report, it is observed that the model is performing well with an accuracy of 93% but the class zero that is the parasitized cell class is misclassified the most.

C. VGG19 Model

The performance of the VGG19 Model is given in the table II.

Looking at the confusion matrix in figure 7 and the classification report, we can say the VGG19 performs in a similar fashion as EfficientNet model with slightly less accuracy of 91%.

TABLE II
PERFORMANCE METRICS

Metric	VGG19
Accuracy	0.91
Precision	Parasitised: 0.98 and Uninfected: 0.86
Recall	Parasitised: 0.84 and Uninfected: 0.98
F1 Score	Parasitised: 0.90 and Uninfected: 0.92

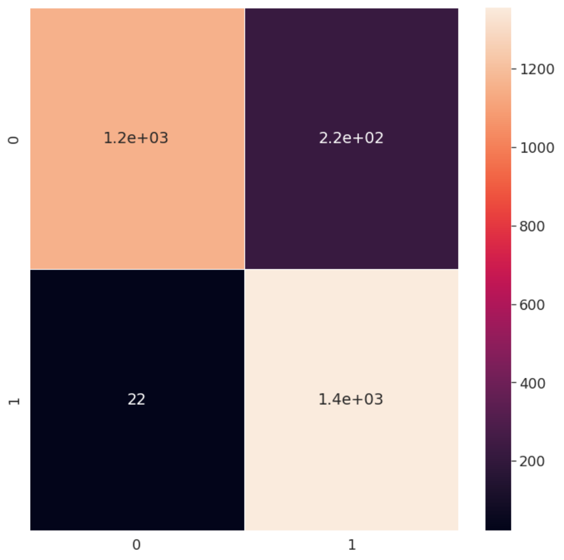


Fig. 7. Confustion Matrix of VGG19 Model

D. Stacked Ensemble Model

The performance of the Stacked Ensemble Model is given in the table III.

TABLE III
PERFORMANCE METRICS

Metric	Stacked Ensemble
Accuracy	0.95
Precision	Parasitised: 0.95 and Uninfected: 0.95
Recall	Parasitised: 0.95 and Uninfected: 0.95
F1 Score	Parasitised: 0.95 and Uninfected: 0.95

From the confusion matrix in figure 8 it can be seen that the performance is improved significantly. With the accuracy of 95%. Which is greater than both base models.

Figure 9 shows the ROC of the stacked models.

From the results the individual models gave a high accuracy with EfficientNet performing slightly better than VGG19. It can be observed from the base models that parasitised cell are misclassified the most even though the dataset is balanced. Yet the percent of misclassification is very low. The classification report, confusion matrix and ROC curve all suggest a good performance from the models. The bagging ensemble model gives more accuracy than the individual base models i.e. 93.5%. When models are stacking they perform even better and among all the models this was the best performing model. Misclassifications are reduced and not a specific class is misclassified as the base models. This stacked model was later

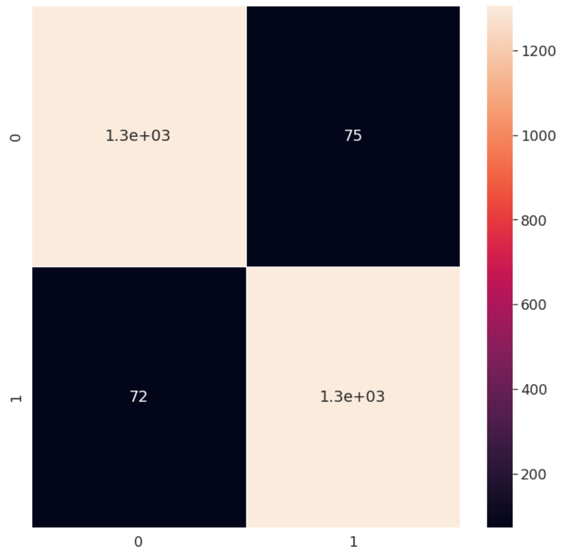


Fig. 8. Confustion Matrix of Stacked Ensemble Model

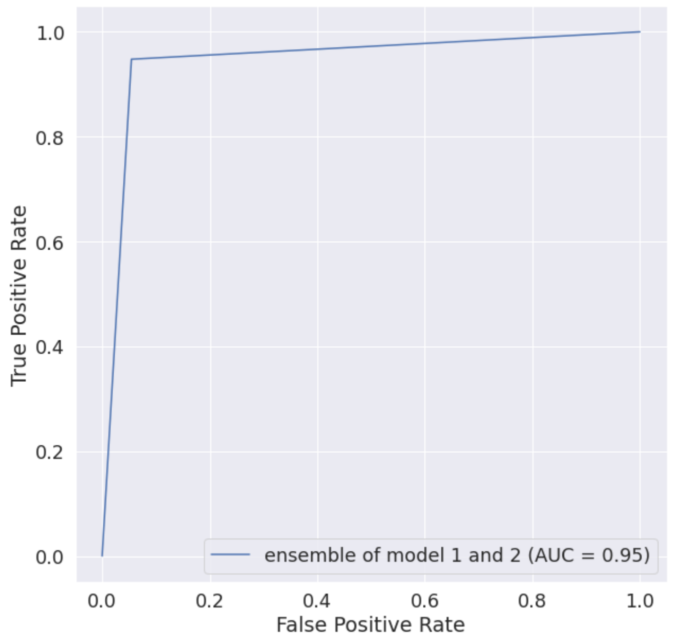
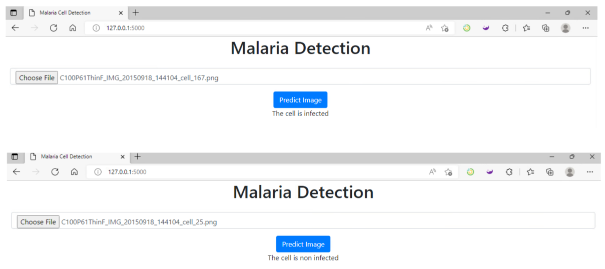


Fig. 9. ROC curve of Stacked Model

used in simulations and as can be seen from the images it classifies the images properly.

E. *Flask Outputs*

The figure 10 shows the outputs of flask, two random images were fed one uninfected cell and one infected cell. As can it be observed the the model was able to correctly predict the classes of images. The two images are same as figure 1.


Malaria Cell Detection
127.0.0.1:5000
Malaria Detection
Choose File C100P61ThinF_IMG_20150918_144104_cell_167.png
Predict Image
The cell is infected

Malaria Cell Detection
127.0.0.1:5000
Malaria Detection
Choose File C100P61ThinF_IMG_20150918_144104_cell_25.png
Predict Image
The cell is non infected

Fig. 10. Outputs of Simulations in Flask

V. Validity and Bias

A. *Pre-processing*

Even though pre-processing is done on the images, a huge number of images is needed to train really good models. Additionally, data augmentation such as brightness, contrast, height, width, rotation, and zooming can be applied to data in order to obtain more images. In this way, a model can be created that is compatible with any image. The model in this case can handle a variety of images since the dataset is built that way and not a lot of augmentation is required.

B. *Modelling*

Since these models are used for training image data the training time required for this model is huge hence this creates limitations on training, as less data needs to be given and the number of epochs also needs to be controlled. Currently, models are run for only 10 epochs. The models are only optimized with the best loss functions and optimizers, but they may also be optimized based on the number of neurons, layers, activation function, learning rate, batch size, and epochs. However, optimizing each parameter would be computationally intensive.

Modeling is done using transfer learning. Although this makes deployment easier, as seen in the literature review, there are higher-performing models as well. But those models are very specific and can't be used for other applications. On the other hand, transfer learning models can be used for other applications without requiring much coding. The models EfficientNet and VGG19 were chosen since they performed better than other models. For instance, InceptionNet performed poorly. EfficientNet in particular is the state of the art architecture hence it was used. VGG19 on the other hand had been a popular choice for image classification among the data science community.

In some cases, bagging may not be a practical solution because it is computationally expensive, despite being highly accurate. Bagging also does not add to the explanation of why it performs better. The cross-validation in the bagging ensembles is the right way to perform bagging but since it was computationally heavy and the data was in the image format a simple proof of concept is presented. Stacking Ensembles provide with a lot of explanation as there is a model involved at the end which can be evaluated.

VI. Conclusion

The purpose of this paper was to implement ensemble models using transfer learning, this was done using convolution neural network architectures namely EfficientNet and VGG19. In the process, a series of experiments were conducted including that of training individual base models, creating three week models for bagging ensembles and stacking the base models. The two base models performed well on their own but gave an higher accuracy when ensembled. The bagging ensemble with 93.5% accuracy and stacking ensemble gave an accuracy of 95% and was the best model among the four models built. This proves that ensembling the models result in better performance than the individual models. The paper also explains how easy it is to implement transfer learning using pre-pretrained weights. This helps models to generalize more as with a slight modification the models can be adjusted for a different image classification model. The simulations also show that the model was able to classify the images successfully.

The models can be further improved by hyperparameter tuning and by feeding a larger data set where the data is augmented and the model gets even more variety of data. Other architectures can be used to see how they perform. When talking about ensembles the data can be K-fold cross validated where it is trained on different sets of data and prediction of these k models can be ensembled. Stacking a different combinations of models might result in better models. With more computing power the model can perform better than the present results.

These inferences coupled with the high classification accuracy can play a part towards building a fully automated system for malaria parasite detection.

Acknowledgment

I would like to acknowledge,

1) The creators of data set. Link to Data
2) Creators of numpy, pandas and tensorflow libraries.
3) Dr Leandro Soriano Marcolino for explaining the concepts of ensembling.

References

[1] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, Qing He, “A Comprehensive Survey on Transfer Learning”, arXiv:1911.02685, 23 Jun 2020.

[2] Alok Kumar, Mayank Jain, ”Ensemble Learning for AI Developers: Learn Bagging, Stacking, and Boosting Methods with Use Cases”, Apress, June 2020.

[3] Nihad Karim Chowdhury, Muhammad Ashad Kabir, Md. Muhtadir Rahman, Noortaz Rezoana, ”ECOVNet: An Ensemble of Deep Convolutional Neural Networks Based on EfficientNet to Detect COVID-19 From Chest X-rays”, arXiv:2009.11850, 16 Oct 2020

[4] Karen Simonyan, Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv:1409.1556

[5] Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. J Big Data 3, 9 (2016)

[6] Fuhad, K M Faizullah et al. “Deep Learning Based Automatic Malaria Parasite Detection from Blood Smear and its Smartphone Based Application.” Diagnostics (Basel, Switzerland) vol. 10,5 329. 20 May. 2020, doi:10.3390/diagnostics10050329

[7] Sen Li, Zeyu Du, Xiangjie Meng, Yang Zhang, Multi-stage malaria parasite recognition by deep learning, GigaScience, Volume 10, Issue 6, June 2021, giab040, https://doi.org/10.1093/gigascience/giab040

[8] Dipanjan Sarkar, Raghav Bali, Tamoghna Ghosh, ”Hands-On Transfer Learning with Python”, Packt Publishing, August 2018

[9] Mingxing Tan, Quoc V. Le, ”EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”, arXiv:1905.11946, 11 Sep 2020.

[10] J. Latif, C. Xiao, A. Imran and S. Tu, ”Medical Imaging using Machine Learning and Deep Learning Algorithms: A Review,” 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019, pp. 1-5, doi: 10.1109/ICOMET.2019.8673502.

[11] Singh, P. (2021). Machine Learning Deployment as a Web Service. In: Deploy Machine Learning Models to Production. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-6546-8_3

APPENDIX

A. *Model 1*

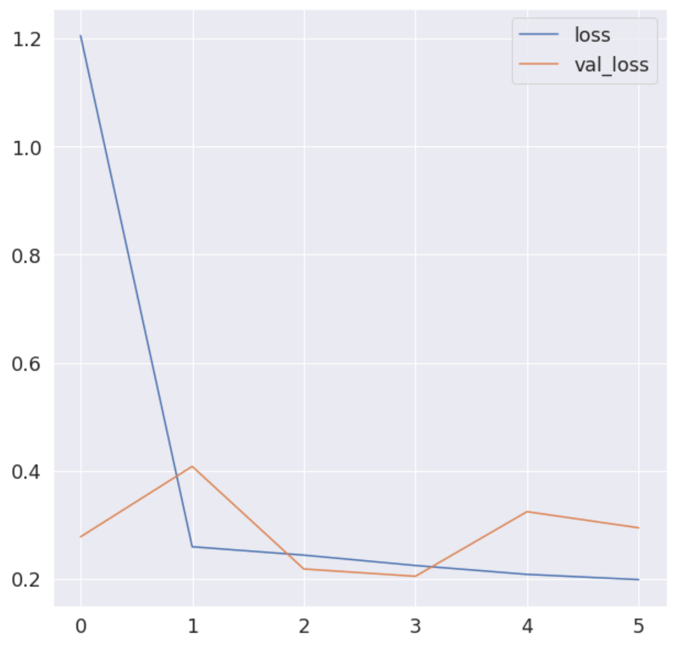


Fig. 11. Model1 loss

Fig 11 explains the loss during the training for EfficientNet Model

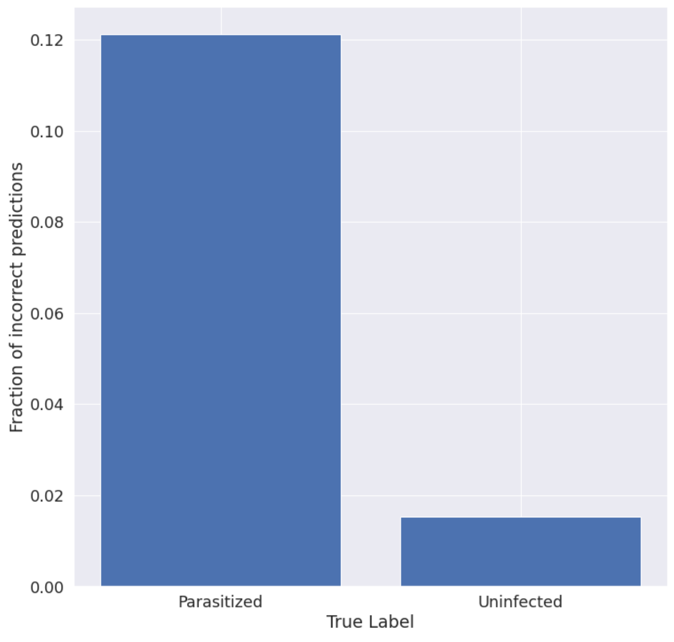


Fig. 12. Model1 incorrect classification

Fig. 12 show that the number of misclassification for Parasitized class are more as compared to Uninfected class of cells.

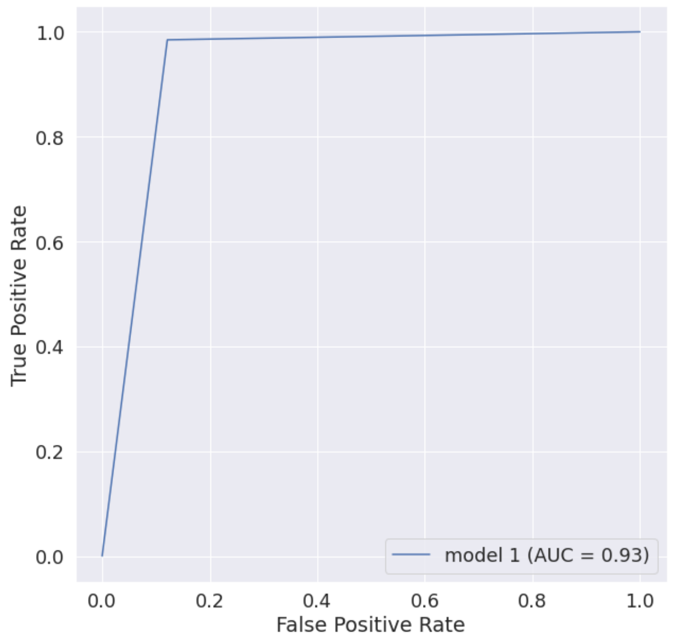


Fig. 13. Model1 loss

Fig. 13 shows the ROC for EfficientNet Model.

B. *Model 2*

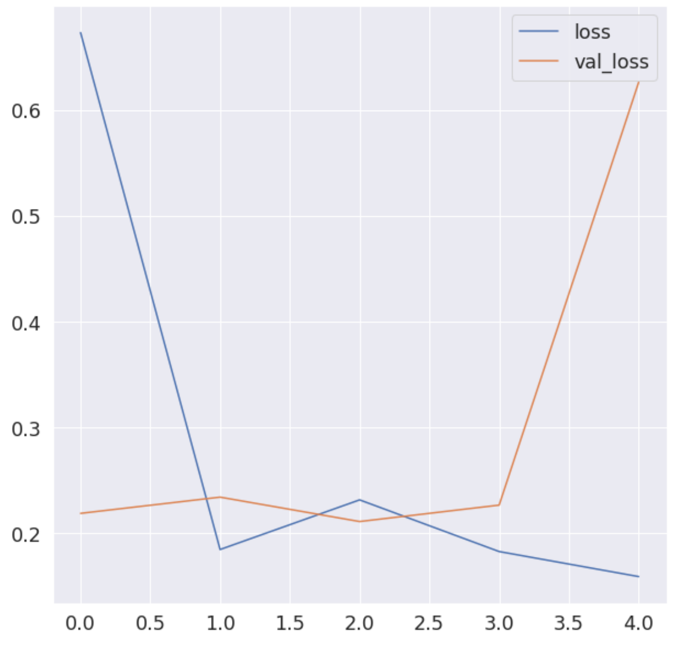


Fig. 14. Model2 loss

Fig. 14 explains the loss during the training for VGG19 Model

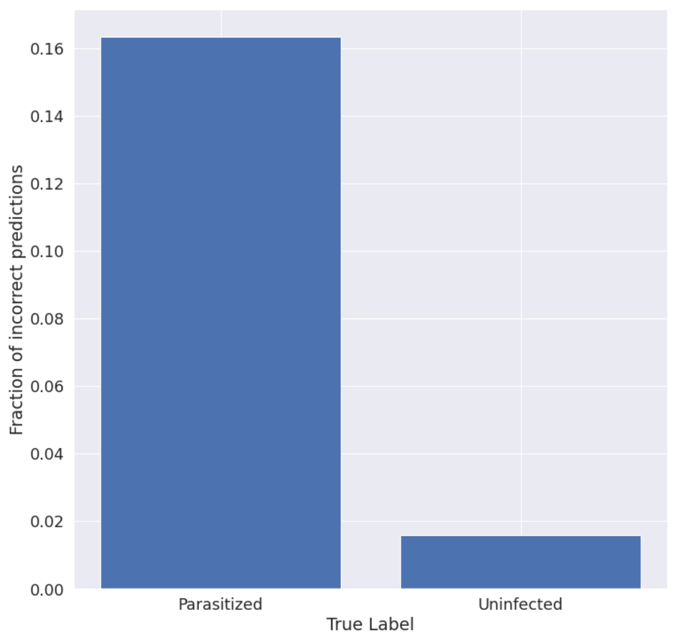


Fig. 15. Model2 incorrect classification

Fig. 15 show that the number of miss classification for Parasitized class are more as compared to Uninfected class of cells.