# Part I

In this part, first I designed a Convolutional Neural Network model with Tensorflow and Keras in Python. Then I chose 2 numbers each for 3 hyperparameters on my model. These Hyperparameters are Layer number, batch size, and optimizer type. The model initially has 3 convolutional layers, a batch size of 32, and an Adam optimizer.

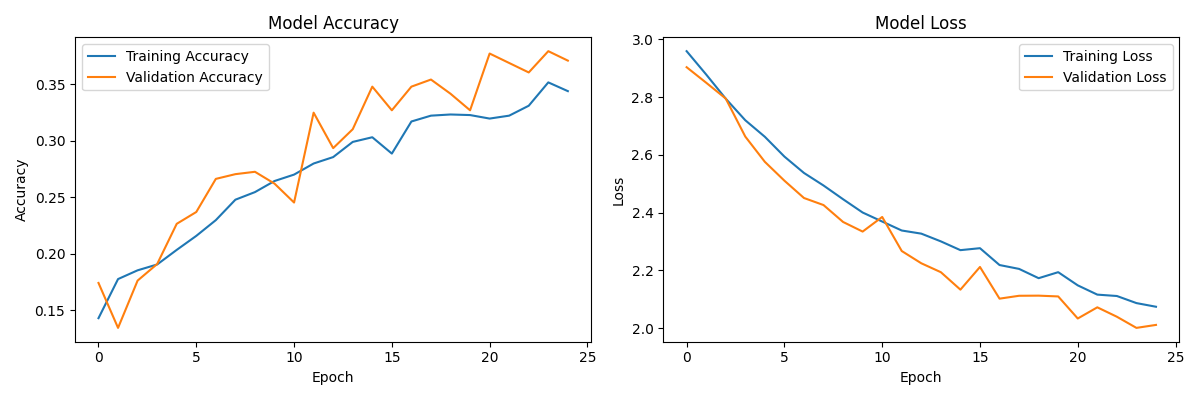
Model 1 is the initial model

Model 2 has 2 convolutional layers

Model 3 has a batch size of 16

Model 4 has SGD as the optimizer

Figure of the models:

 Fig.1 : Accuracy and Loss plots of the initial model.

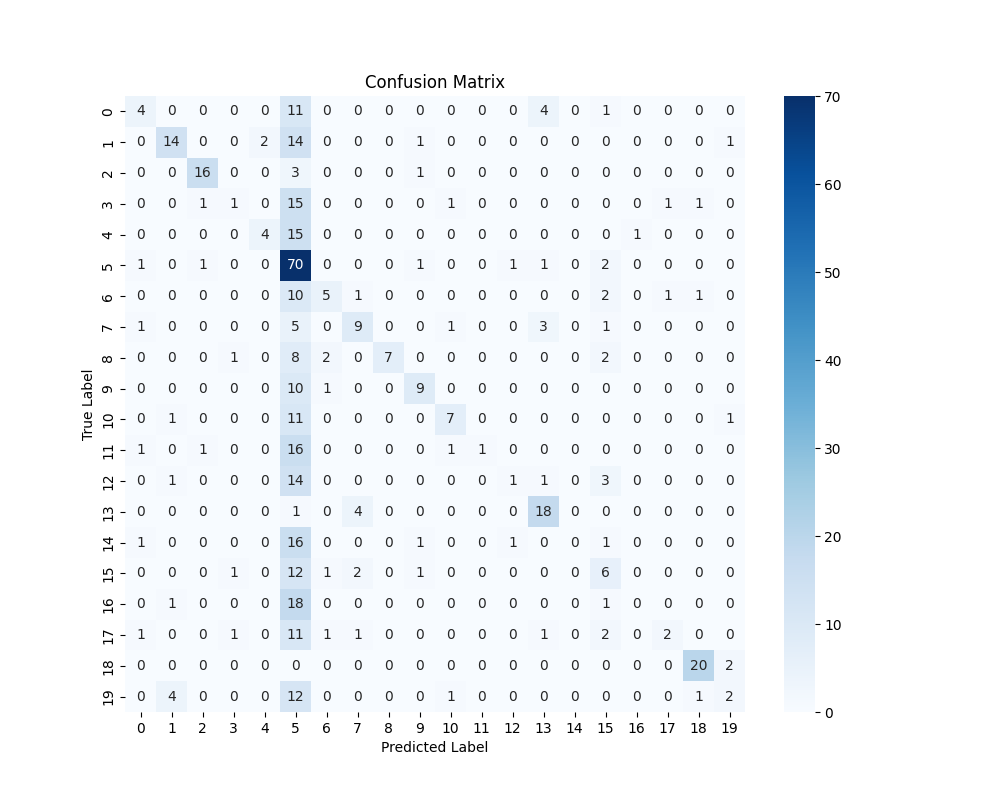


Fig. 2 Confusion matrix of the initial modal

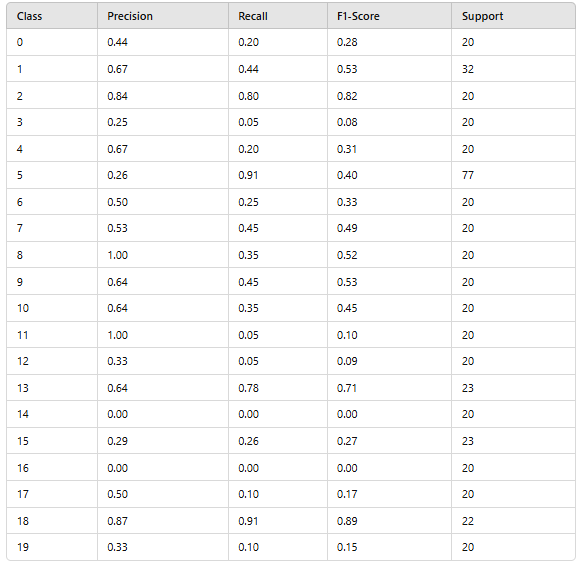


Fig. 3 statistics for each classes

Weighted Average Metrics on the best modal:

Precision: 0.4931

Recall: 0.4109

F1-score: 0.3691

Tables of the models:

|  |  |
| --- | --- |
| Model Numbers | Overall Accuracy |
| Model 1 (Initial) | 0.4109 |
| Modal 2 (2 layers) | 0.3795 |
| Modal 3 (SGD optimizer) | 0.2411 |
| Modal 4 (Batch size 16) | 0.4004 |

Table 1: Accuracies of the models.

Results & Conclusions: For this model, the best hyperparameters are 3 convolutional layers, batch size 32, and Adam optimizer. And there is a big drop in the accuracy with the Stochastic Gradient Descent optimizer.

# Part II

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Accuracy | Epoch time avg. | Precision | Recall | F1-Score |
| VGG 16 | 0.8784 | 120 seconds | ≈ 0.879 | ≈ 0.879 | ≈ 0.879 |
| EfficientNetB0 | 0.9581 | 43 seconds | 0.9617 | 0.9581 | 0.9580 |
| Resnet50 | 0.9266 | 141 seconds |  |  |  |
| Inception v3 | 0.9371 | 41 seconds | 0.9400 | 0.9371 | 0.9359 |
| MobileNet | 0.9371 | 18 seconds | 0.9379 | 0.9371 | 0.9353 |

Table 2: Showing the evaluation metrices of transfer learning models.

Results & Conculusions: EfficientNetB0 gave the best results on accuracy. Also Mobilenet has good results evan though it is much more faster. The results shows it is possiable that for 25 epochs faster models which has fewer paramteres can give better results on image recognition.

# Part III

In this part the performance CNN will be used for feature descriptor. I will use the three good CNN modals at the previous parts. They are MobileNet, EfficientNetB0, ResNet50. And I will test them at first layers, middle layers, last layers in this order.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | Percision | Recall | F1-Score |
| MobileNet 1 | 0.1761 | 0.1432 | 0.1761 | 0.1054 |
| MobileNet 2 | 0.2977 | 0.3579 | 0.2977 | 0.2306 |
| MobileNet 3 | 0.9455 | 0.9484 | 0.9455 | 0.9450 |
| EfficientNetB0 1 | 0.2537 | 0.2647 | 0.2537 | 0.1973 |
| EfficientNetB0 2 | 0.2872 | 0.2471 | 0.2872 | 0.2092 |
| EfficientNetB0 3 | 0.9665 | 0.9682 | 0.9665 | 0.9665 |
| ResNet50 1 | 0.1950 | 0.0892 | 0.1950 | 0.0909 |
| ResNet50 2 | 0.3249 | 0.3464 | 0.3249 | 0.2586 |
| ResNet50 3 | 0.9563 | 0.9577 | 0.9563 | 0.9559 |

Table 3: Comparing the results of Layer choose.

Results & Conclusions: Later, the feature descriptor used better results been on this part.

# Results

The best results are achieved when transfer learning is used. Transfer Learning had a huge impact on accuracy. Initiating the transfer learning at the first layers didn’t seem to be a good approach. However, using it at an earlier layer from the very last layer proved to be a better performer.

Models have a bias towards the class 5 DogHead. The Reason is that every class has almost 100 images but the dog class has nearly 500 images, which causes this bias. This situation is especially causing high error at simple models without transfer learning.

# Conclusion

The overall best model is EfficientNetB0 a version of EfficientNet v1. Originally, it is for classifying different kinds of food. It is very good for accuracy and speed.

Transfer Learning can increase the accuracy significantly without taking much more time then a normal CNN model.