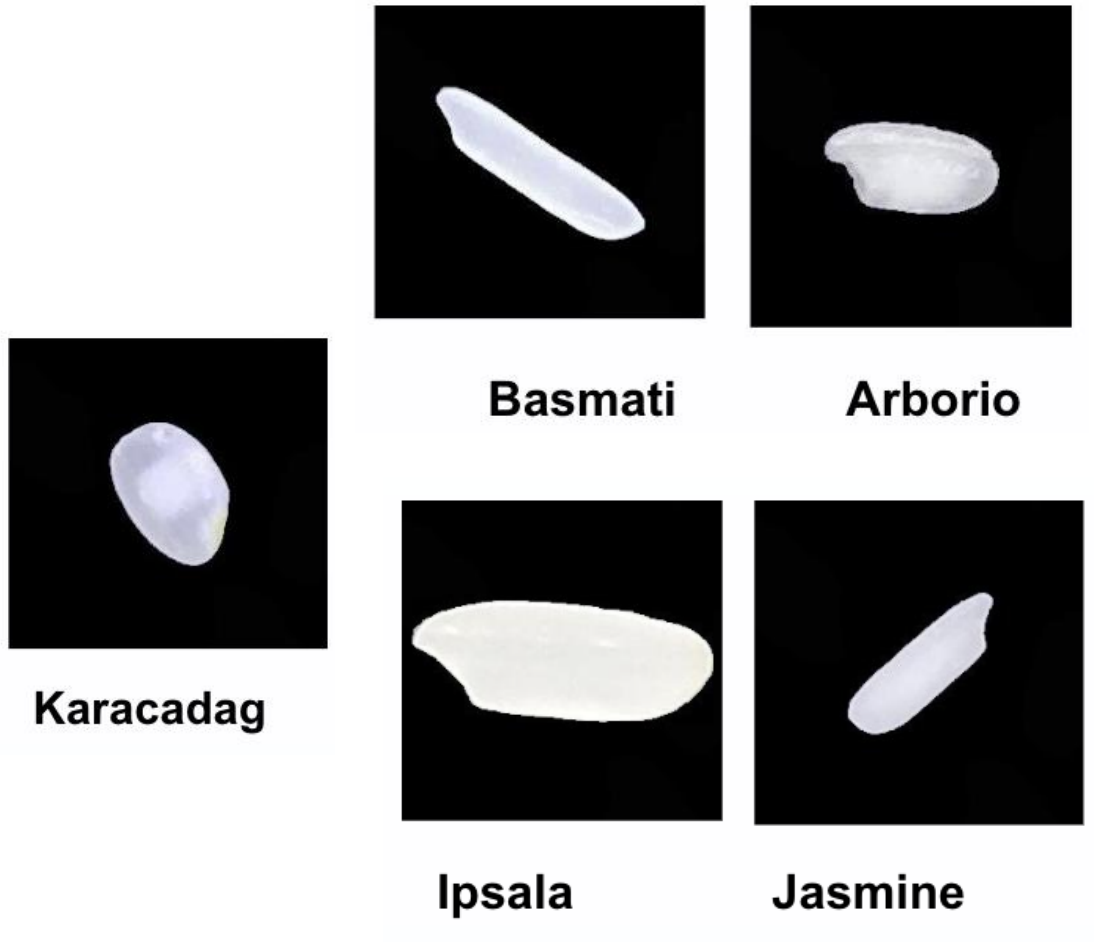
Optimizing a Convolutional Neural Network for Image Classification of Rice Grains

Alina Ahmed

Introduction

- Rice grains have genetic variations, observable by differing features such as: texture, shape and colour.
- It is possible to classify different types of rice based on these features.
- The task was to create an optimized CNN, building on publicly available pre-trained CNNs to to classify images of rice grains into 5 types: Arborio, Basmati, Ipsala and Jasmine.
- This problem was a classification problem, to be solved by a deep learning neural network as the input would be images.
- The importance of ensuring accurate classification of the rice grains is so in the food industry rice can be sorted, packaged and sold to different markets.
- A dataset of 75,00 images was available but not used due to computational limitations. Instead, 500 images from the dataset were used during building the model and then the final optimised model was trained on a dataset of 1,500 images. These numbers were decided upon a balance of time and accuracy; max time was 1hr and minimum accuracy was 0.9.

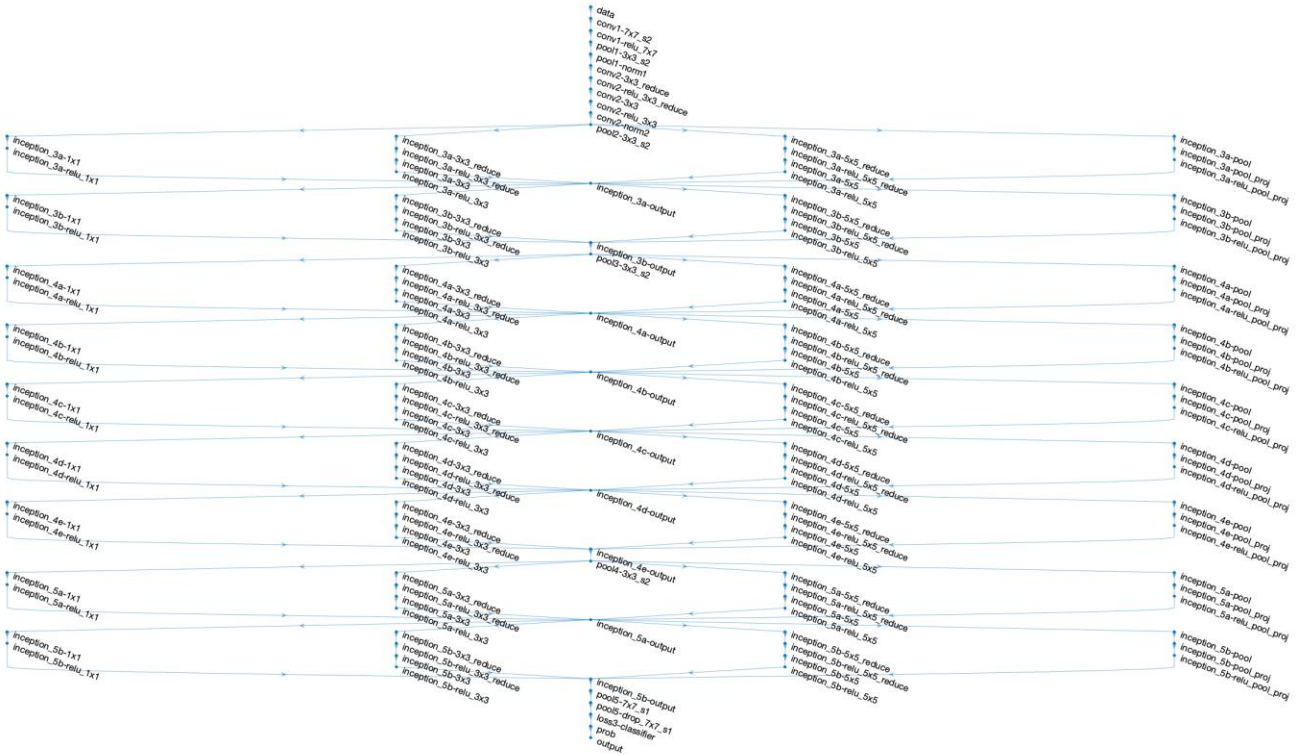


Data Pre-processing

- 3 types of data augmentation were done on the input images:
  - The images were re-sized to fit the required size of the input layer of the pre-trained neural networks
  - The images were augmented using various translation and reflection functions
  - Noise was introduced into the data – colour jittering
- Colour noise and translation augmentations were decided because logically they are the most likely to occur – e.g. introducing motion blur noise would not be logical as rice is not alive and moves, its stationary

Network Topology

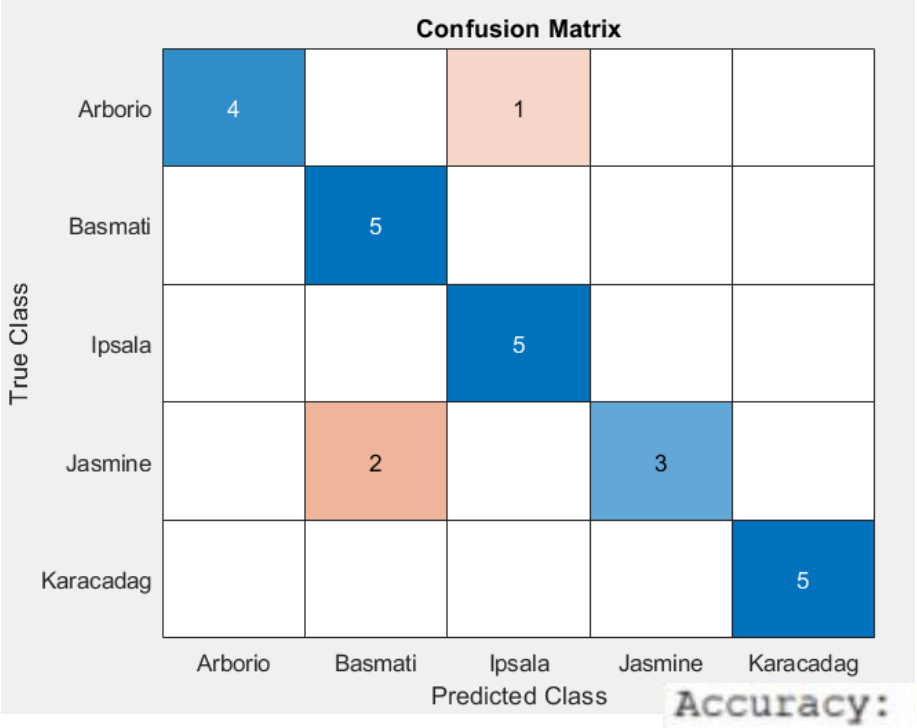
- 2 Different types of pretrained CNNs were tried: AlexNet and GoogleNet
- In the rest of this project any exploration or experimentation done on one network was also done on the other, so in the end there are 2 optimized CNNS
- Performance metrics used to assess the unoptimized models was Classification Accuracy



AlexNet

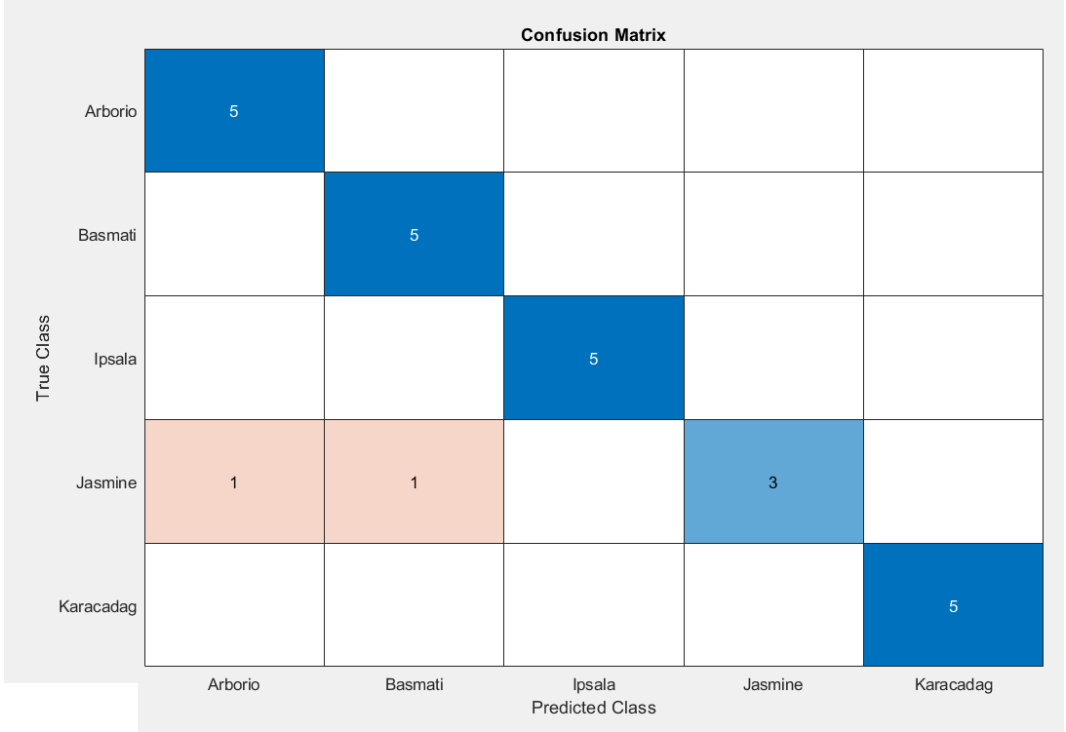
GoogleNet

AlexNet




```
Accuracy: 0.88
Confusion Matrix:
     4     0     1     0     0
     0     5     0     0     0
     0     0     5     0     0
     0     2     0     3     0
     0     0     0     0     5
```

GoogleNet




```
Accuracy: 0.92
Confusion Matrix:
     5     0     0     0     0
     0     5     0     0     0
     0     0     5     0     0
     1     1     0     3     0
     0     0     0     0     5
```

Data Splicing

- Data was split into the following 3 categories:
  - Training set: data for the network to update its weights and biases
  - Validation set: data to identify and minimize overfitting so the network does not learn the training data and can perform well with unseen data
  - Test set: data that is unseen and assesses the final performance of the trained model
- Experimented with 3 different common ratios of splitting of data:
    - 80:10:10
    - 70:15:15
    - 60:20:20
- Evaluated the performance of the model with the different ratios using Classification Accuracy
- Tried each datasplit 3 times and calculated avergae
- GoogleNet: 70:15:15
- AlexNet: 80:10:10

AlexNet:

```
Average accuracy for Data split 1: 0.97778
Average accuracy for Data split 2: 0.92533
Average accuracy for Data split 3: 0.95
Average accuracies for all Data Splits in Alexnet:
    0.9778
    0.9253
    0.9500
```

GoogleNet:

```
Average accuracy for Data split 1: 0.98148
Average accuracy for Data split 2: 0.984
Average accuracy for Data split 3: 0.96458
Average accuracies for all Data Splits in Googlenet:
    0.9815
    0.9840
    0.9646
```

Hyperparameter Tuning: Training Functions

- Through multiple iterative processes, the training function was selected to be one which produced the smallest Mean Squared Error
- Tested the following training functions:
  - Sgdm
  - Adam
  - rmsprop
- The evaluation metric used to decide the most optimal function was classification accuracy
- Best training function for Alexnet:
  - rmsprop
- Best training function for Googlenet:
  - Sgdm
  - rmsprop

```
Training with optimizer in Alexnet: sgdm
Accuracy with sgdm optimizer: 0.95
Training with optimizer in Alexnet: adam
Accuracy with adam optimizer: 0.9
Training with optimizer in Alexnet: rmsprop
Accuracy with rmsprop optimizer: 0.975
Training with optimizer in Googlenet: sgdm
Accuracy with sgdm optimizer: 1
Training with optimizer in Googlenet: adam
Accuracy with adam optimizer: 0.875
Training with optimizer in Googlenet: rmsprop
Accuracy with rmsprop optimizer: 1
```

Hyperparameter Tuning: Batch Size, Epochs, Initial Learning Rate and Learning Rate Schedule

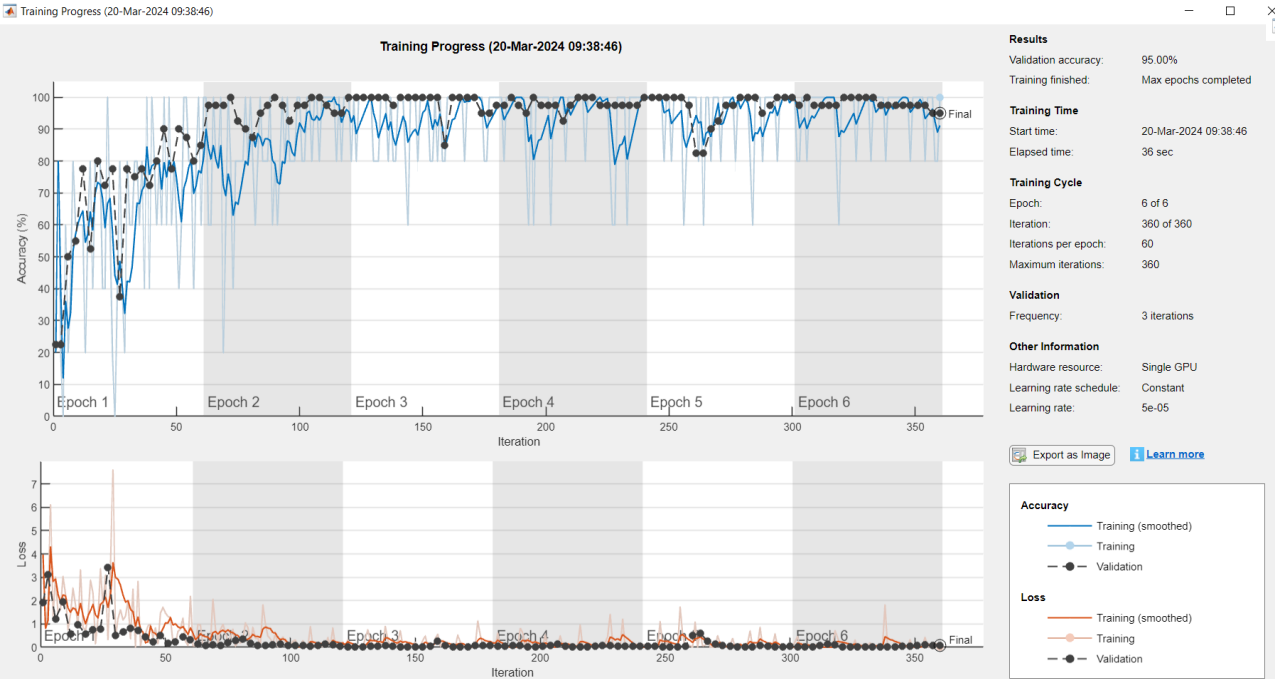
AlexNet

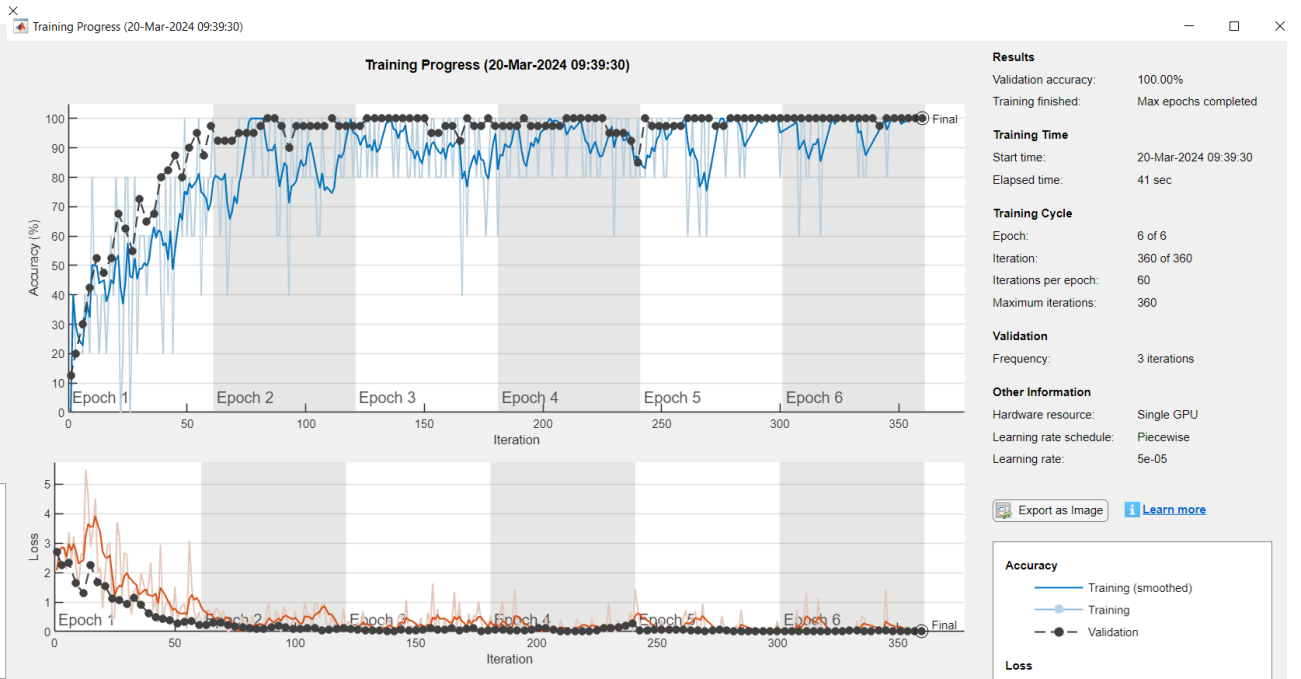
Initial Learning rate and Batch Size: First tuned the initial learning rate with a fixed mini-batch size, then tuned the mini-batch size using the best initial learning rate found. Displayed the best initial learning rate and best mini-batch size along with their corresponding classification accuracies.

Epochs: Visually observed when accuracy plateaus and doesn't improve further so training for more epochs is not useful

Learning rate schedule: Tried constant vs 'piecewise' learning rate schedule and observed graph differences (other parameters are already optimised).

```
Best initial learning rate: 5e-05, Accuracy: 1
Best mini-batch size: 5, Accuracy: 1
Accuracy WITHOUT piecwise learning rate schedule: 0.95
Accuracy with piecwise learning rate schedule: 1
```





Hyperparameter Tuning: Batch Size, Epochs, Initial Learning Rate and Learning Rate Schedule

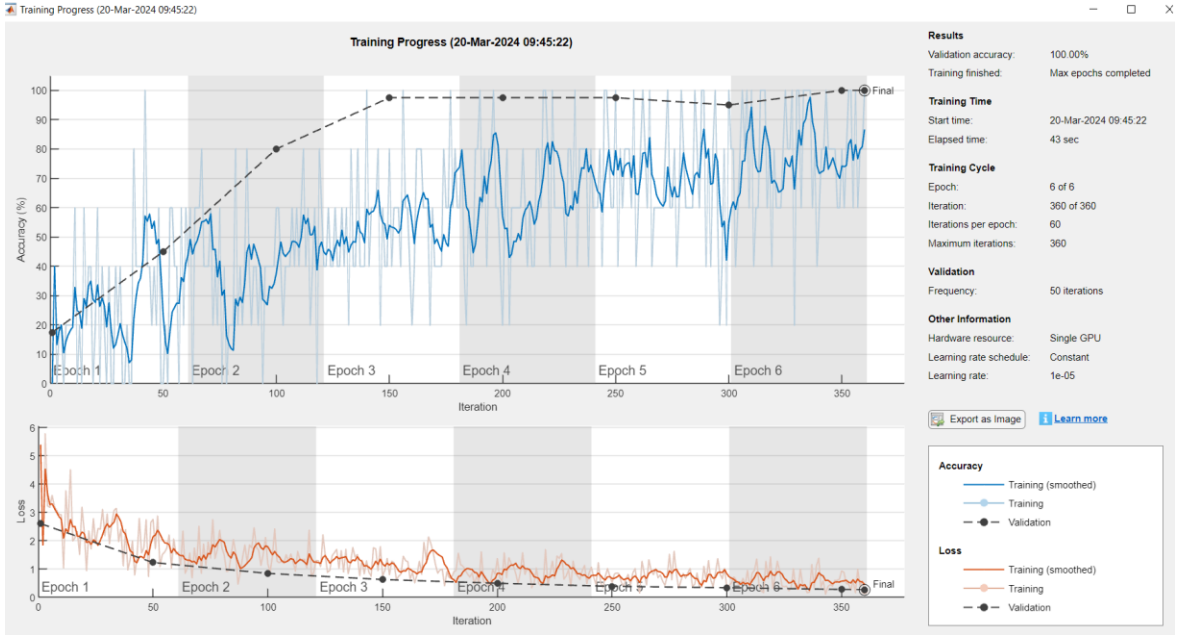
GoogleNet

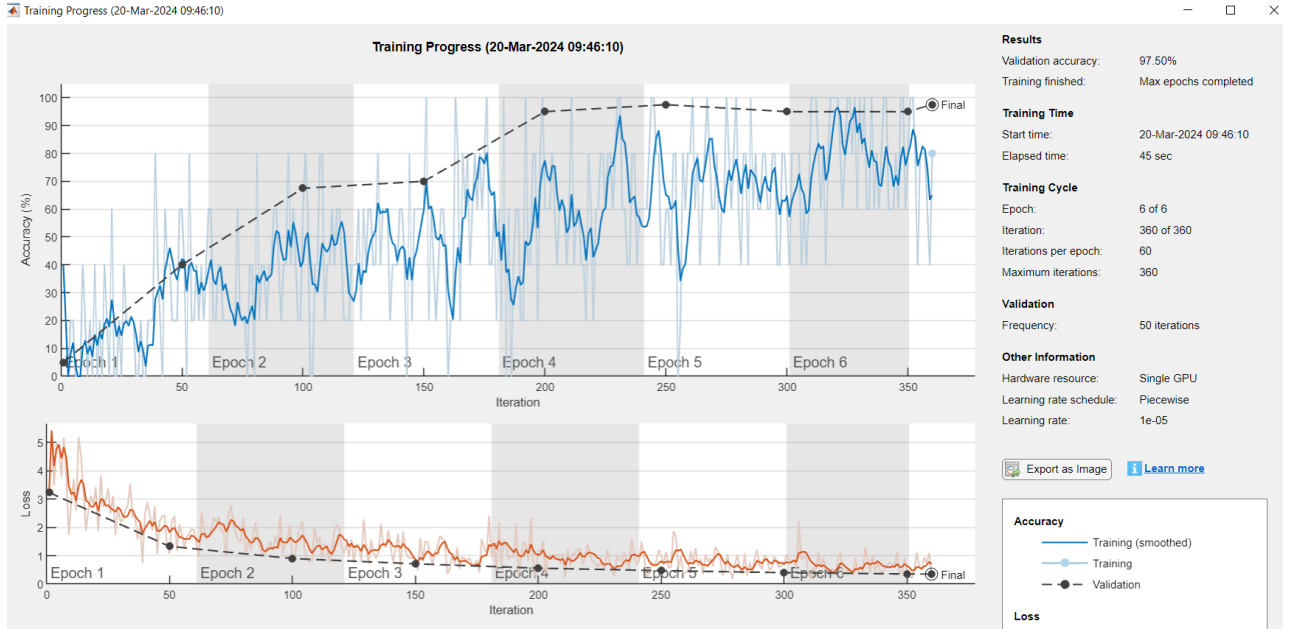
Initial Learning rate and Batch Size: First tuned the initial learning rate with a fixed mini-batch size, then tuned the mini-batch size using the best initial learning rate found. Displayed the best initial learning rate and best mini-batch size along with their corresponding classification accuracies.

Epochs: Visually observed when accuracy plateaus and doesn't improve further so training for more epochs is not useful

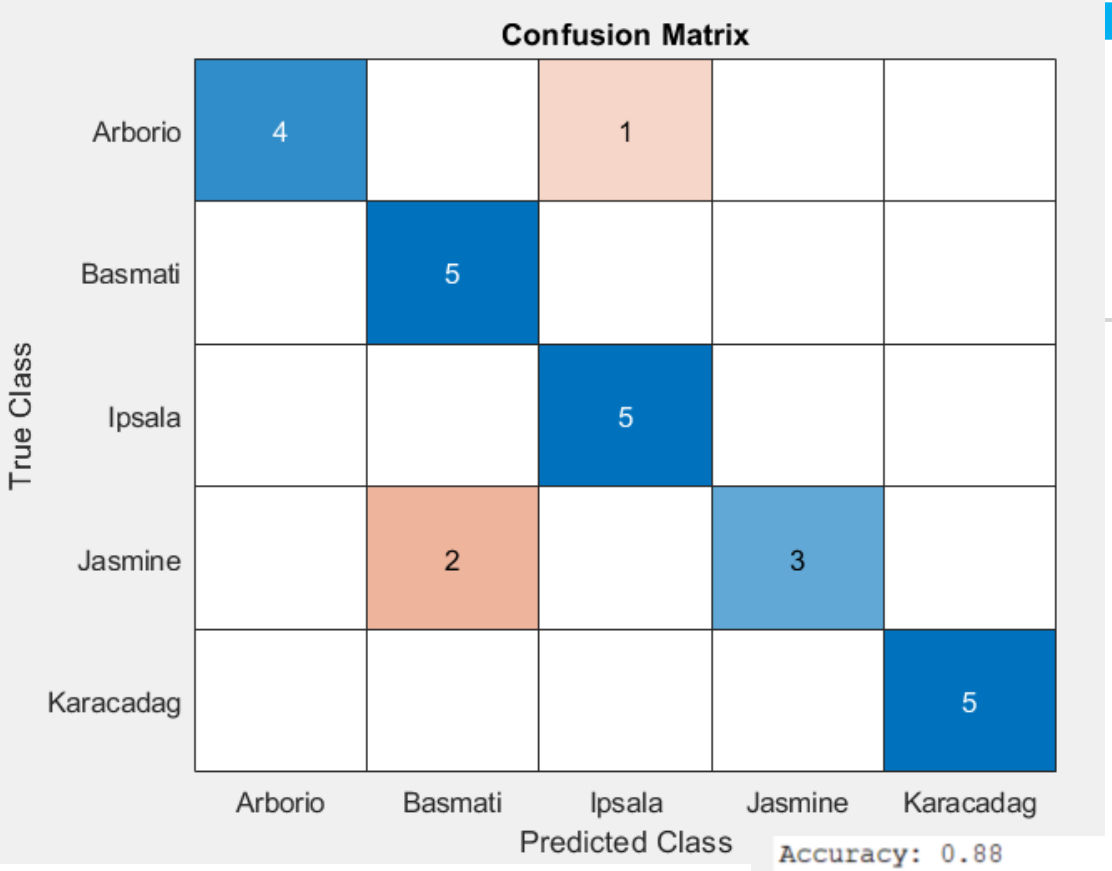
Learning rate schedule: Tried constant vs 'piecewise' learning rate schedule and observed graph differences (other parameters are already optimised).

```
Best initial learning rate: 1e-05, Accuracy: 1
Best mini-batch size: 5, Accuracy: 0.95
Accuracy WITHOUT piecewise learning rate schedule: 1
Accuracy WITH piecewise learning rate schedule: 0.975
```





AlexNet initial vs final optimized model

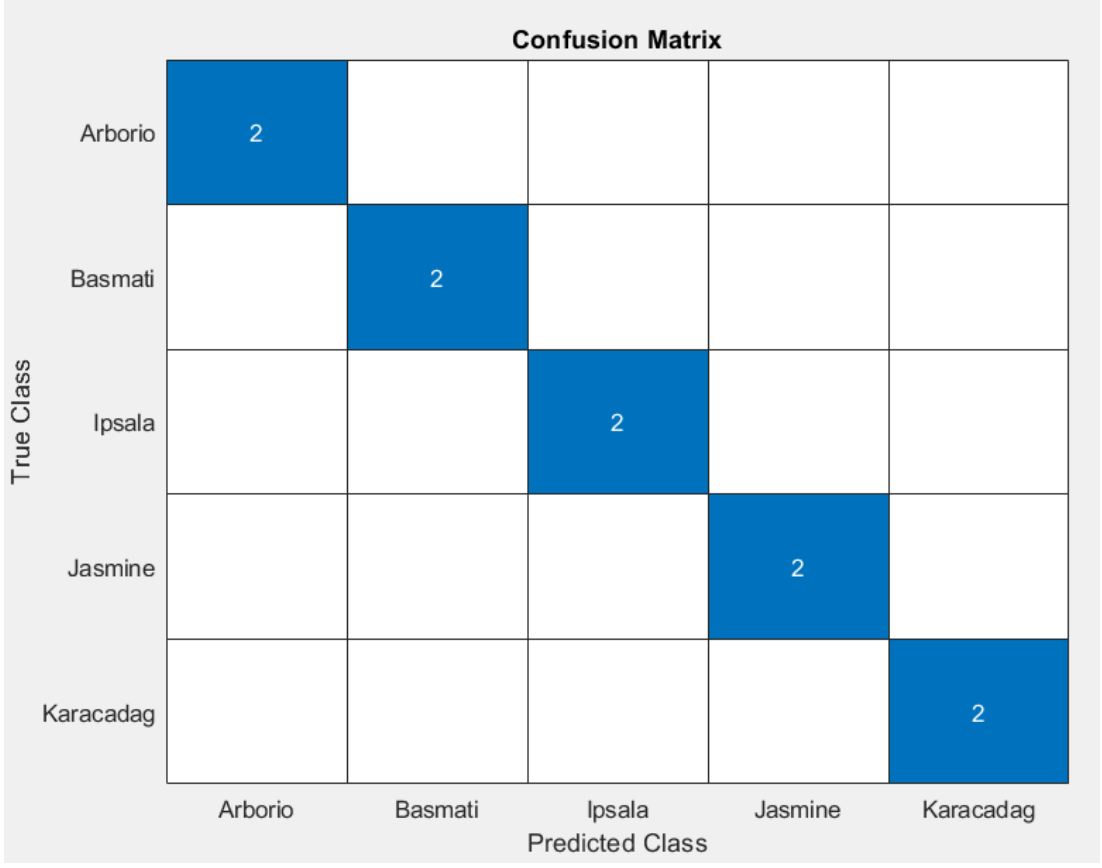


Initial

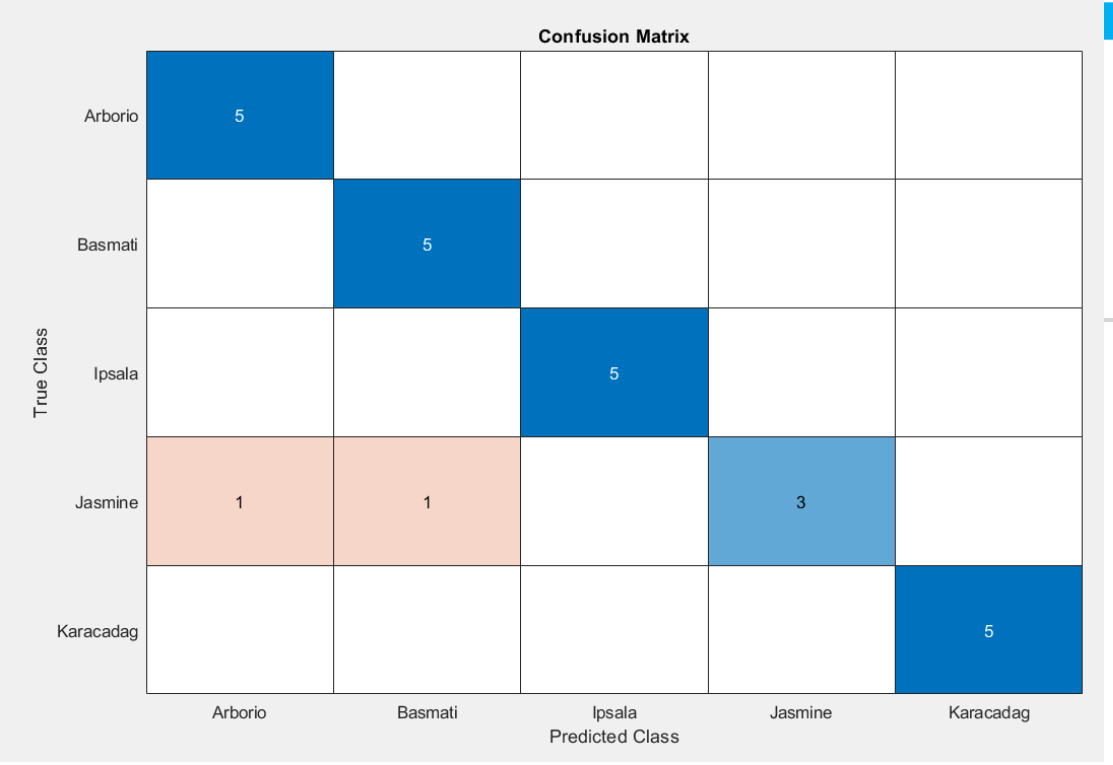
```
Accuracy: 0.88
Confusion Matrix:
     4     0     1     0     0
     0     5     0     0     0
     0     0     5     0     0
     0     2     0     3     0
     0     0     0     0     5
```

```
Accuracy: 1
Confusion Matrix:
     2     0     0     0     0
     0     2     0     0     0
     0     0     2     0     0
     0     0     0     2     0
     0     0     0     0     2
```

Final



GoogleNet initial vs final optimized model



Initial

```
Accuracy: 0.92
Confusion Matrix:
     5     0     0     0     0
     0     5     0     0     0
     0     0     5     0     0
     1     1     0     3     0
     0     0     0     0     5

Accuracy: 0.92
Confusion Matrix:
     5     0     0     0     0
     0     5     0     0     0
     0     0     5     0     0
     0     1     0     4     0
     1     0     0     0     4
```

Final