**Assessment 2:  
Machine Learning Report**

CAB320 Artificial Intelligence, Sem 1 2024

May 31, 2024

**Group 32**

*Kenzie Haigh, n10755012*

*Luke Whitton, n10814256*

*Emma Wu, n11132833*

# 1. Introduction

This report presents the implementation and evaluation of a flower classifier using transfer learning with a pretrained MobileNetV2 model. The goal is to train a classifier capable of distinguishing different classes of flowers in an image, and to evaluate its performance using various metrics.

# 2. Methodology

The methodology involves using transfer learning with a pretrained MobileNetV2 model. The last layer of the model is replaced with a Dense layer of appropriate shape for the five classes of the small flower dataset. The data is split into training, validation, and test sets, and the model is trained using an SGD optimizer with various parameters. The performance of the model is evaluated using precision, recall, F1 score, and confusion matrix.

# 3. Function Descriptions and Analysis

## 3.1 my\_team()

* + **Description**: This function returns a list of tuples, each containing the student ID, first name, and last name of the team members.
  + **Analysis**: This function is straightforward and ensures that all team members are listed. It is essential for identifying the contributors to the project.

## 3.2 load\_model()

* + **Description**: This function loads the MobileNetV2 model with weights pre-trained on ImageNet, excludes the top layers, and adds a new Dense layer with five output nodes (one for each flower class) with a softmax activation function.
  + **Analysis**: This function effectively sets up the model for transfer learning by reusing the pre-trained MobileNetV2 layers to extract features and adding custom layers for the specific classification task.

## 3.3 load\_data(path)

* **Description**: This function loads the dataset from the specified directory, resizes the images to 224x224 pixels, and returns the images and labels as NumPy arrays.
  + **Analysis**: This function ensures that the dataset is correctly loaded and pre-processed, which is crucial for training and evaluating the model. It also manages the resizing of images and batching.

## 3.4 split\_data(X, Y, train\_fraction, randomize=False, eval\_set=True)

* **Description**: This function splits the dataset into training, testing, and optionally evaluation sets. It can also randomize the data before splitting.
* **Analysis**: This function provides flexibility in splitting the dataset and ensures that the splits are appropriate for training, validation, and testing. It is critical for creating balanced and unbiased datasets for model evaluation.

## 3.5 confusion\_matrix(predictions, ground\_truth)

* + **Description**: This function computes the confusion matrix given the predictions and ground truth labels.
  + **Analysis**: The confusion matrix is a valuable tool for evaluating the performance of the classifier by showing the true positive and false positive rates for each class.

## 3.6 plot\_confusion\_matrix(cm, classes)

* + **Description:** This function plots the confusion matrix using matplotlib, providing a visual representation of the classifier's performance. It includes labels for the classes and color coding to differentiate between higher and lower values in the matrix.
* **Analysis:** This function visually represents the confusion matrix, making it easier to understand the performance of the classifier at a glance. It highlights the true positives and false positives clearly, which helps in identifying which classes are being confused by the model.

## 3.7 precision(predictions, ground\_truth)

* **Description**: This function calculates the precision for each class by dividing the number of true positives by the sum of true positives and false positives.
* **Analysis**: Precision measures the proportion of positive identifications that were actually correct. It is crucial for understanding the classifier's performance in identifying true positives without too many false positives. This metric is particularly important in scenarios where the cost of false positives is high.

## 3.8 recall(predictions, ground\_truth)

* **Description**: This function calculates the recall for each class by dividing the number of true positives by the sum of true positives and false negatives.
* **Analysis**: Recall measures the proportion of actual positives that were correctly identified. It is important for understanding the classifier's ability to identify all relevant instances. This metric is particularly important in scenarios where the cost of false negatives is high.

## 3.9 f1(predictions, ground\_truth)

* **Description**: This function calculates the F1 score for each class, which is the harmonic mean of precision and recall.
* **Analysis**: The F1 score provides a single metric that balances both false positives and false negatives, making it useful for overall performance evaluation. It is especially useful when the classes are imbalanced, providing a more comprehensive measure than precision or recall alone.

## 3.10 k\_fold\_validation(features, ground\_truth, classifier, k=3)

* **Description**: This function performs k-fold cross-validation, splitting the dataset into k folds, training on k-1 folds, and testing on the remaining fold. It returns the average metrics (precision, recall, F1 scores) and their standard deviations.
* **Analysis**: K-fold cross-validation provides a robust evaluation by ensuring every data point is used for both training and testing. It helps in understanding the generalizability of the model and reduces the bias associated with a single train-test split. The averaged metrics give a more accurate picture of model performance across different subsets of the data.

## 3.11 transfer\_learning(train\_set, eval\_set, test\_set, model, parameters)

* **Description**: This function implements transfer learning with given parameters (learning rate, momentum, and nesterov), trains the model on the training set, validates on the evaluation set, and tests on the test set. It evaluates the model's performance using recall, precision, and F1 scores.
* **Analysis**: This function efficiently implements transfer learning by using a pre-trained model and fine-tuning it on the flower dataset. It evaluates the performance using important metrics, providing insights into how well the model generalizes to new data. The use of SGD optimizer with different parameters helps in understanding the impact of these parameters on the model's performance.

## 3.12 accelerated\_learning(train\_set, eval\_set, test\_set, model, parameters)

* **Description**: This function implements accelerated learning with data augmentation and mixed precision training. It uses an ImageDataGenerator for data augmentation and a learning rate scheduler to adjust the learning rate during training.
* **Analysis**: Accelerated learning uses data augmentation and mixed precision to improve the training process. This function provides a way to train models more efficiently, potentially achieving better performance by leveraging techniques that enhance the model's ability to generalize from augmented data and optimized computational resources.

## 3.13 plot\_history(history, title="Training and Validation Metrics")

* **Description**: This function plots the training and validation accuracy and loss over epochs, providing a visual representation of the model's performance during training.
* **Analysis**: This function is crucial for visualizing the model's performance during training and validation. It helps in identifying issues like overfitting or underfitting and provides insights into how well the model is learning. By comparing training and validation metrics, it becomes easier to diagnose problems with the model and adjust training strategies accordingly.

# 4. Experiments and Results

## 4.1 Data Splitting

The data was split into training, validation, and test sets using an 80-10-10 split ratio. Care was taken to ensure that the split was reasonable and did not introduce class imbalance.

## 4.2 Learning Rate Experiments

Experiments were conducted with three different learning rates (0.1, 0.01, 0.001). The following graphs show the training and validation accuracy and loss for each learning rate.

## 4.3 Outcomes and Analysis

A graph of a training with blue dots

Description automatically generated with medium confidence

Figure 1: Initial Training with Learning Rate 0.1

**Caption:** This graph shows the training and validation accuracy and loss for the initial training phase with a learning rate of 0.1.

**Analysis:** The trends in accuracy and loss values over the epochs help determine the effectiveness of the chosen learning rate. Generally, increasing accuracy and decreasing loss indicate a successful training process. If validation accuracy plateaus or decreases while training accuracy continues to increase, it might indicate overfitting.

A graph of a training with a number of points

Description automatically generated with medium confidence

Figure 2: Initial Training with Learning Rate 0.01

**Caption:** This graph shows the training and validation accuracy and loss for the initial training phase with a learning rate of 0.1.

**Analysis:** The trends in accuracy and loss values over the epochs help determine the effectiveness of the chosen learning rate. Generally, increasing accuracy and decreasing loss indicate a successful training process. If validation accuracy plateaus or decreases while training accuracy continues to increase, it might indicate overfitting.

A graph of a training and training

Description automatically generated with medium confidence

Figure 3: Initial Training with Learning Rate 0.001

**Caption:** This graph shows the training and validation accuracy and loss for the initial training phase with a learning rate of 0.1.

**Analysis:** The trends in accuracy and loss values over the epochs help determine the effectiveness of the chosen learning rate. Generally, increasing accuracy and decreasing loss indicate a successful training process. If validation accuracy plateaus or decreases while training accuracy continues to increase, it might indicate overfitting.

A graph of a training

Description automatically generated with medium confidence

Figure 4: Transfer Learning - Training with Best Learning Rate 0.1

**Caption:** This graph represents the training and validation metrics for the transfer learning process using the best learning rate identified (0.1).

**Analysis:** The graph should ideally show improved performance compared to the initial training, indicating the effectiveness of transfer learning. Better training and validation accuracy and lower loss values suggest that the model is leveraging pre-trained weights effectively.

A blue squares with white squares

Description automatically generated

Figure 5: Confusion Matrix

**Caption:** This graph shows the confusion matrix for the classifier on the test dataset using the best learning rate.

**Analysis:** The confusion matrix provides a detailed breakdown of the model’s performance by showing the true positive and false positive rates for each class. High values along the diagonal indicate correct classifications, while off-diagonal values represent misclassifications. Analysing this matrix helps identify specific classes where the model is struggling and guides further improvements.

## 4.4 Evaluation Metrics

The table below summarizes the precision, recall, and F1 scores for the classifier on the test dataset using the best learning rate. Please fill in the data accordingly.

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1 Score** |
| Class 0 |  |  |  |
| Class 1 |  |  |  |
| Class 2 |  |  |  |
| Class 3 |  |  |  |
| Class 4 |  |  |  |

## 4.5 Discussion

The results from the experiments provide insights into the performance of the classifier. The learning rate experiments indicated that 0.1 was the most effective learning rate. The confusion matrix highlighted the classes where the model performed well and those where it struggled. The evaluation metrics, including precision, recall, and F1 scores, provide a comprehensive view of the model's performance. Further experiments with different momentum values and accelerated transfer learning could provide additional improvements.

# 5. Conclusion

In conclusion, the transfer learning approach with a pretrained MobileNetV2 model proved effective in building a flower classifier. The best learning rate identified was 0.1, and the evaluation metrics indicated good performance. Future work could explore fine-tuning the model and experimenting with different architectures to further improve performance.