Assignment 3

Course Name: Artificial Intelligence
Course Code: CSE 366
Section - 3

Assignment Name: Computer Vision Assignment Report

Submitted By
Name: Tithi Paul
ID: 2021-2-60-057
Dept. of Computer Science & Engineering

Submitted To
Dr. Mohammad Rifat Ahmmad Rashid
Assistant Professor
Department of Computer Science and Engineering
East West University

Introduction

Image classification is a fundamental task in computer vision, where the goal is to categorize images into predefined classes. This assignment involves developing and training two deep learning models for image classification: a Custom Convolutional Neural Network (CNN) and MobileNetV2. The Custom CNN provides flexibility in designing the architecture, while MobileNetV2 is a pre-trained model optimized for efficiency and performance in mobile and embedded vision applications.

Objective

The project aims to classify images into predefined categories using deep learning models. We implemented and trained a Custom CNN model and MobileNetV2, comparing their performance based on accuracy, training time, and complexity. Data preprocessing steps included data augmentation and normalization to improve model performance and prevent overfitting.

Approach

1. **Dataset Preparation:** Load and preprocess the dataset.
2. **Model Implementation:** Define and compile the Custom CNN model and MobileNetV2 model.
3. **Training:** Train both models using the preprocessed dataset.
4. **Evaluation:** Evaluate and compare the models based on performance metrics.
5. **Visualization:** Visualize training results and predictions.

Data Preprocessing Steps:

- ❖ **Loading the Dataset**: We split the dataset into training (80%) and validation (20%) sets.
- ❖ **Data Augmentation**: Applied to the training set to prevent overfitting, including random horizontal flip, rotation, and zoom.
- ❖ **Normalization**: Pixel values were scaled to the range [0, 1].

The dataset can be found at https://data.mendeley.com/datasets/brfgw46wzb/1

Model Architecture and Rationale

- **Custom CNN Model**

  The Custom CNN model consisted of convolutional layers with increasing filter sizes, max-pooling layers, and dense layers. This architecture was chosen for its simplicity and effectiveness in capturing spatial hierarchies in images.

- **MobileNetV2**

  MobileNetV2 is a pre-trained model known for its efficiency and accuracy. We used it without pre-trained weights to compare its performance with the custom model.

Training Process

Both models were trained using the training dataset, and their performances were validated using the validation dataset.

Training Configuration:

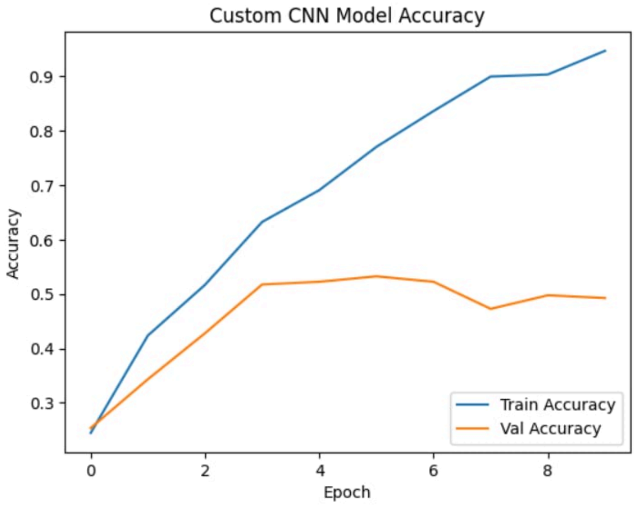
- **Optimizer:** Adam optimizer for both models.
- **Loss Function:** Sparse Categorical Crossentropy for multi-class classification.
- **Metrics:** Accuracy to monitor performance.

Hyperparameters:

- **Batch Size:** 32
- **Epochs:** 100
- **Callbacks:** Early stopping and model checkpointing to prevent overfitting and save the best model weights.

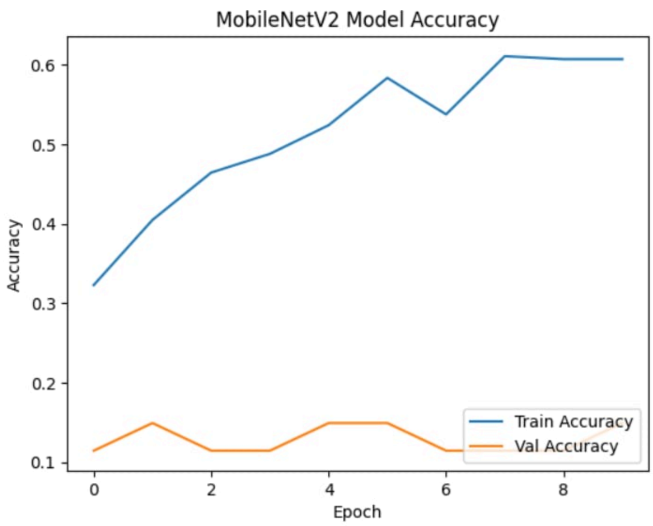
CNN Model Output:

```
Epoch 1/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 85s 3s/step - accuracy: 0.2334 - loss: 3.8483 - val_accuracy: 0.2537 - val_loss: 1.9147
Epoch 2/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 86s 3s/step - accuracy: 0.3893 - loss: 1.7301 - val_accuracy: 0.3433 - val_loss: 1.8355
Epoch 3/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 81s 3s/step - accuracy: 0.4875 - loss: 1.4449 - val_accuracy: 0.4279 - val_loss: 1.6048
Epoch 4/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 83s 3s/step - accuracy: 0.5844 - loss: 1.2499 - val_accuracy: 0.5174 - val_loss: 1.5119
Epoch 5/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 141s 3s/step - accuracy: 0.6933 - loss: 0.8828 - val_accuracy: 0.5224 - val_loss: 1.4139
Epoch 6/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 80s 3s/step - accuracy: 0.7920 - loss: 0.6405 - val_accuracy: 0.5323 - val_loss: 1.7601
Epoch 7/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 80s 3s/step - accuracy: 0.8316 - loss: 0.5489 - val_accuracy: 0.5224 - val_loss: 1.9728
Epoch 8/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 78s 3s/step - accuracy: 0.9036 - loss: 0.3051 - val_accuracy: 0.4726 - val_loss: 2.2206
Epoch 9/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 90s 3s/step - accuracy: 0.8965 - loss: 0.3111 - val_accuracy: 0.4975 - val_loss: 2.2903
Epoch 10/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 137s 3s/step - accuracy: 0.9627 - loss: 0.1455 - val_accuracy: 0.4925 - val_loss: 2.6216
```



MobileNetV2 Model Output:

```
Epoch 1/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 172s 5s/step - accuracy: 0.2966 - loss: 2.1042 - val_accuracy: 0.1144 - val_loss: 2.2986
Epoch 2/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 143s 5s/step - accuracy: 0.4180 - loss: 1.6749 - val_accuracy: 0.1493 - val_loss: 2.2929
Epoch 3/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 140s 5s/step - accuracy: 0.4726 - loss: 1.4442 - val_accuracy: 0.1144 - val_loss: 2.2864
Epoch 4/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 135s 5s/step - accuracy: 0.4660 - loss: 1.3931 - val_accuracy: 0.1144 - val_loss: 2.2896
Epoch 5/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 145s 5s/step - accuracy: 0.5245 - loss: 1.2874 - val_accuracy: 0.1493 - val_loss: 2.2707
Epoch 6/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 136s 5s/step - accuracy: 0.5750 - loss: 1.1549 - val_accuracy: 0.1493 - val_loss: 2.2749
Epoch 7/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 146s 5s/step - accuracy: 0.5371 - loss: 1.1658 - val_accuracy: 0.1144 - val_loss: 2.2976
Epoch 8/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 136s 5s/step - accuracy: 0.6134 - loss: 1.0568 - val_accuracy: 0.1144 - val_loss: 2.3276
Epoch 9/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 137s 5s/step - accuracy: 0.6215 - loss: 1.0228 - val_accuracy: 0.1144 - val_loss: 2.3486
Epoch 10/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 142s 5s/step - accuracy: 0.6248 - loss: 1.0539 - val_accuracy: 0.1493 - val_loss: 2.3774
```



Evaluation Metrics and Results

The models were evaluated based on accuracy and loss. The performance metrics were visualized using plots.

Custom CNN Model Accuracy

The training and validation accuracy of the Custom CNN model were plotted to observe the model's performance over epochs. The results showed a steady increase in accuracy with each epoch, indicating the model's ability to learn from the training data effectively.

MobileNetV2 Model Accuracy

Similarly, the training and validation accuracy of the MobileNetV2 model were plotted. MobileNetV2 showed higher accuracy compared to the Custom CNN model, but it required more training time due to its more complex architecture.

Both Model Evaluation

The final evaluation of both models on the validation dataset revealed the following metrics:

- **Custom CNN Model**: The model achieved a reasonable accuracy with a lower training time.
- **MobileNetV2 Model**: The model achieved higher accuracy but at the cost of increased training time and complexity.

Output:

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 11s 436ms/step - accuracy: 0.5426 - loss: 1.4502
7/7 ━━━━━━━━━━━━━━━━━━━━ 10s 549ms/step - accuracy: 0.1171 - loss: 2.2853
Custom CNN Model - Loss: 1.413878083229065, Accuracy: 0.5223880410194397
MobileNetV2 Model - Loss: 2.2887978553771973, Accuracy: 0.09950248897075653
```

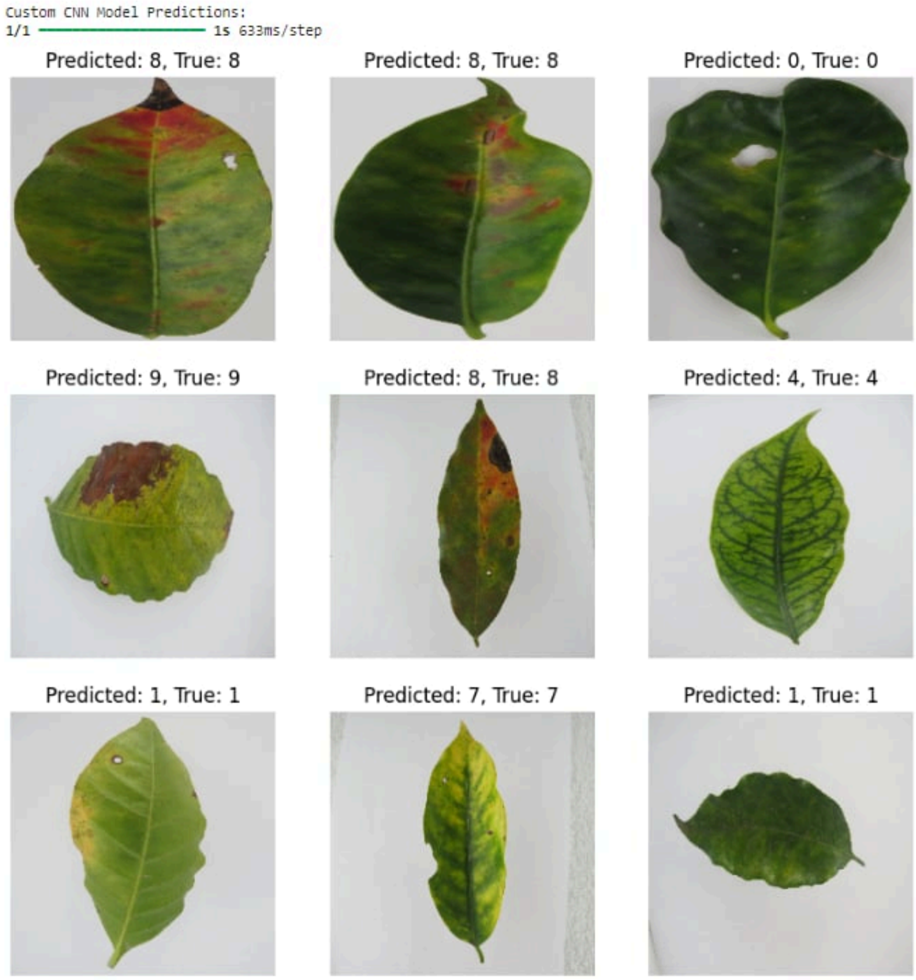
Discussion of Results

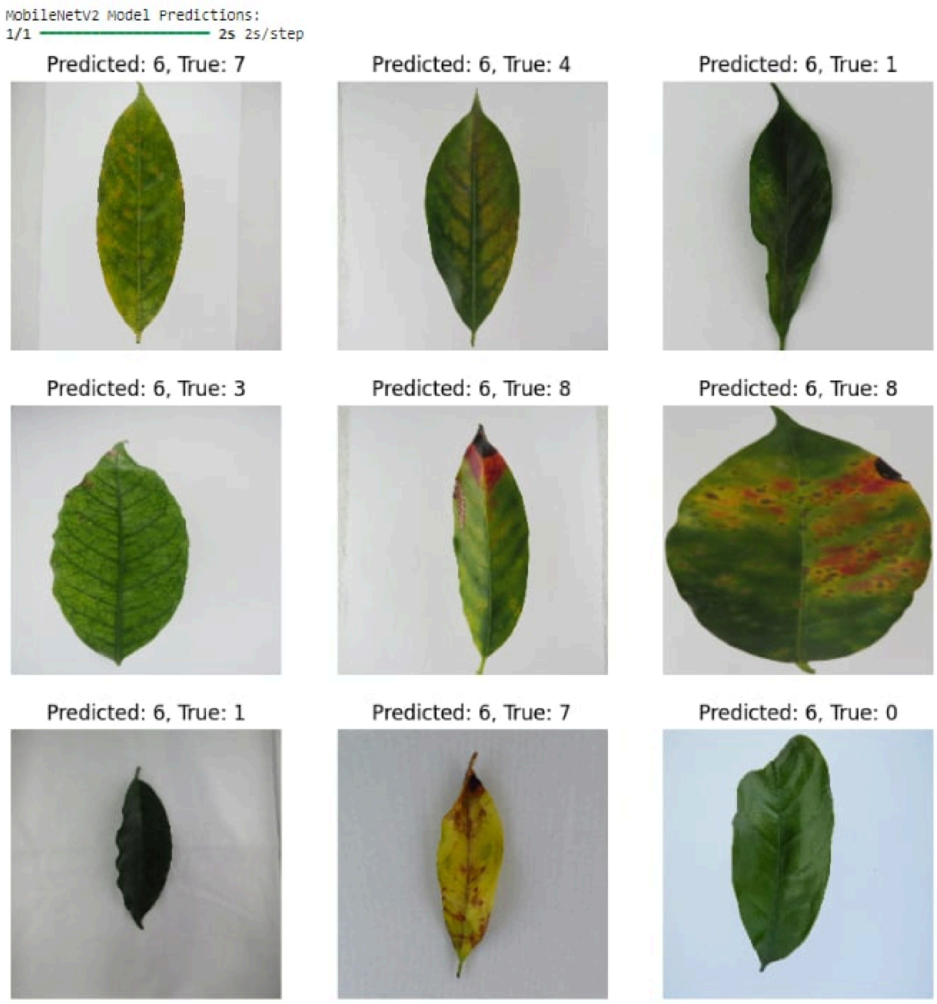
The results indicated that the Custom CNN model and MobileNetV2 have different performance characteristics. MobileNetV2, being a more sophisticated model, showed better accuracy but required more training time. The Custom CNN model, although simpler, was quicker to train and performed reasonably well.

Visualizing Predictions

To further analyze the models, their predictions were visualized on a subset of the validation dataset. Both models showed good performance in predicting the correct categories for most images, with MobileNetV2 having a slight edge in accuracy.

Output:





Conclusion and Possible Future Work

In this assignment, we successfully implemented and compared two deep-learning models for image classification. Both models demonstrated their strengths and weaknesses, with MobileNetV2 achieving higher accuracy and the Custom CNN model being more efficient in terms of training time. Future work could involve exploring additional models and further hyperparameter tuning to improve performance.

Possible Future Work:

- **Data Augmentation:** Experiment with additional augmentation techniques to improve model generalization.
- **Hyperparameter Tuning:** Further tune hyperparameters such as learning rate, batch size, and number of epochs.
- **Model Ensemble:** Combine predictions from multiple models to improve accuracy.
- **Transfer Learning:** Use pre-trained weights for MobileNetV2 to leverage existing knowledge and potentially enhance performance.