EAST WEST UNIVERSITY

Assignment 3

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

Assignment Name: Computer Vision Assignment Report

Sumitted By
Name: Ismail Mahmud Nur
ID: 2021-2-60-052
Dept. of Computer Science & Engineering

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

Image Classification Using Custom CNN and DenseNet121

Introduction

This assignment focuses on developing and training deep learning models for image classification using a specified dataset. The objective is to implement and compare two models: a Custom Convolutional Neural Network (CNN) and DenseNet121. The Custom CNN is built from scratch, while DenseNet121 is a pre-trained model known for its efficiency in complex image classification tasks.

The process involves loading and preprocessing the dataset, applying data augmentation and normalization, training the models, and evaluating their performance based on accuracy and loss. This report provides a concise overview of the methodology, implementation, and comparative analysis of the two models, highlighting their performance and potential areas for improvement.

Objective

The objective of this assignment is to develop and train deep learning models for image classification tasks using a specified dataset. The goal is to implement two models, a Custom CNN and DenseNet121, and compare their performance based on accuracy, training time, and complexity

Approach

1. **Dataset Preparation:** Load and preprocess the dataset.
2. **Model Implementation:** Define and compile the Custom CNN model and DenseNet121 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.

Dataset Preprocessing Steps:

The dataset for this assignment is available at https://data.mendeley.com/datasets/brfgw46wzb/1. It consists of numerous images of coffee leaves sorted into various predefined categories.

- **Dataset Loading:**
  The dataset is loaded from the provided path using TensorFlow's image_dataset_from_directory function, with an 80-20 split for training and validation.

- **Data Augmentation:**
  Data augmentation techniques such as random horizontal flip, rotation, and zoom are applied to the training dataset to enhance model generalization.

- **Normalization:**
  Image pixel values are normalized to a range of [0, 1] to facilitate faster and more stable training.

Model Architecture and Rationale for Choosing Specific Models

Custom CNN Model:

- **Architecture:** Consists of multiple convolutional layers followed by max-pooling layers, and dense layers.
- **Rationale:** A custom CNN allows for flexibility in designing the architecture suited to the dataset, providing control over the depth and complexity of the model.

DenseNet121 Model:

- **Architecture:** A pre-trained DenseNet121 model with the top layers replaced to fit the current dataset.
- **Rationale:** DenseNet121 is chosen for its efficiency and ability to mitigate the vanishing gradient problem through dense connections, which is beneficial for complex image classification tasks.

Training Process (Including Hyperparameter Tuning)

Training Configuration:

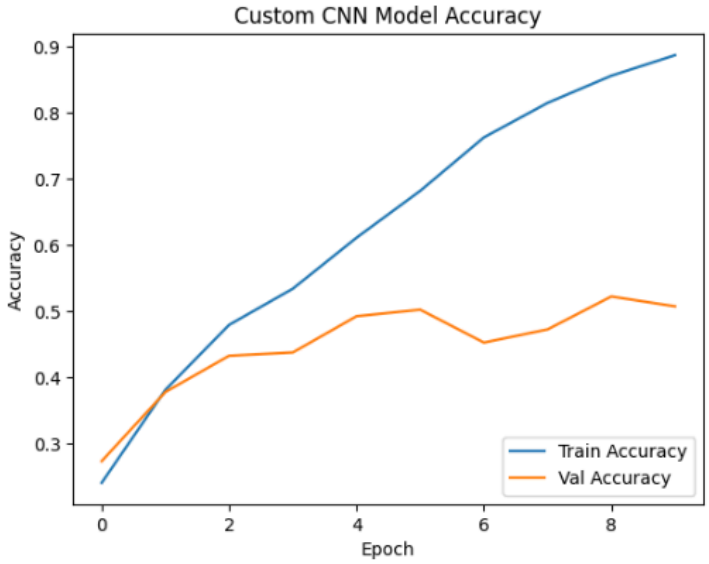
- **Optimizer:** Adam optimizer for both models.
- **Loss Function:** Sparse Categorical Crossentropy for multi-class classification.
- **Metrics:** Accuracy to monitor performance.

Hyperparameters:

- **Batch Size:** 32
- **Epochs:** 50
- **Callbacks:** Early stopping and model checkpointing to prevent overfitting and save the best model weights.

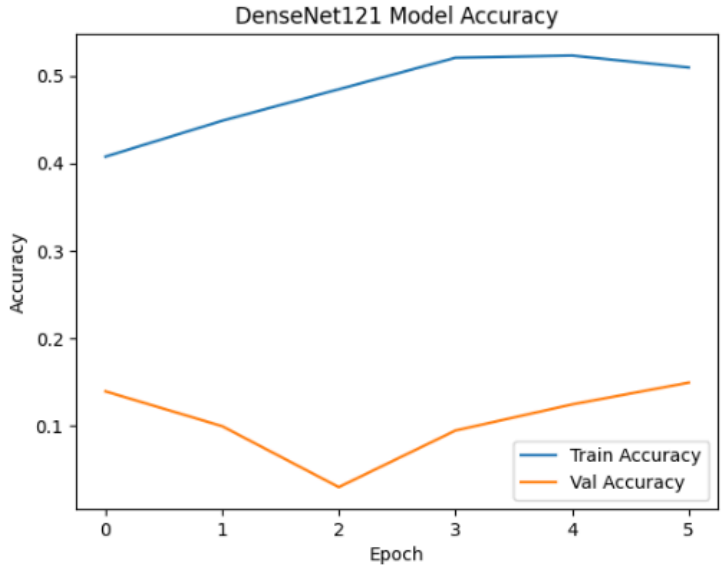
CNN Model Output:

```
Epoch 1/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 87s 3s/step - accuracy: 0.1921 - loss: 3.0714 - val_accuracy: 0.2736 - val_loss: 1.8964
Epoch 2/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 138s 3s/step - accuracy: 0.3669 - loss: 1.7684 - val_accuracy: 0.3781 - val_loss: 1.7339
Epoch 3/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 83s 3s/step - accuracy: 0.4608 - loss: 1.4957 - val_accuracy: 0.4328 - val_loss: 1.5806
Epoch 4/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 142s 3s/step - accuracy: 0.5121 - loss: 1.3793 - val_accuracy: 0.4378 - val_loss: 1.6133
Epoch 5/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 82s 3s/step - accuracy: 0.6313 - loss: 1.0629 - val_accuracy: 0.4925 - val_loss: 1.4528
Epoch 6/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 143s 3s/step - accuracy: 0.6900 - loss: 0.8260 - val_accuracy: 0.5025 - val_loss: 1.4770
Epoch 7/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 141s 3s/step - accuracy: 0.7559 - loss: 0.7269 - val_accuracy: 0.4527 - val_loss: 1.9672
Epoch 8/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 141s 3s/step - accuracy: 0.7930 - loss: 0.5631 - val_accuracy: 0.4726 - val_loss: 1.8395
Epoch 9/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 83s 3s/step - accuracy: 0.8498 - loss: 0.4775 - val_accuracy: 0.5224 - val_loss: 1.6788
Epoch 10/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 142s 3s/step - accuracy: 0.8773 - loss: 0.3634 - val_accuracy: 0.5075 - val_loss: 1.9887
```



DenseNet121 Model Output:

```
Epoch 1/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 469s 14s/step - accuracy: 0.3584 - loss: 1.8743 - val_accuracy: 0.1393 - val_loss: 2.3974
Epoch 2/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 368s 14s/step - accuracy: 0.4430 - loss: 1.5361 - val_accuracy: 0.0995 - val_loss: 2.5828
Epoch 3/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 381s 14s/step - accuracy: 0.4635 - loss: 1.4402 - val_accuracy: 0.0299 - val_loss: 2.7858
Epoch 4/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 358s 13s/step - accuracy: 0.5322 - loss: 1.3195 - val_accuracy: 0.0945 - val_loss: 4.6119
Epoch 5/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 382s 13s/step - accuracy: 0.5491 - loss: 1.2404 - val_accuracy: 0.1244 - val_loss: 7.2663
Epoch 6/50
26/26 ━━━━━━━━━━━━━━━━━━━━ 356s 13s/step - accuracy: 0.5091 - loss: 1.2794 - val_accuracy: 0.1493 - val_loss: 3.2261
```



Evaluation Metrics and Results with Visualizations

Evaluation Metrics:

- **Accuracy:** Primary metric for comparison.
- **Loss:** Monitored during training and validation.

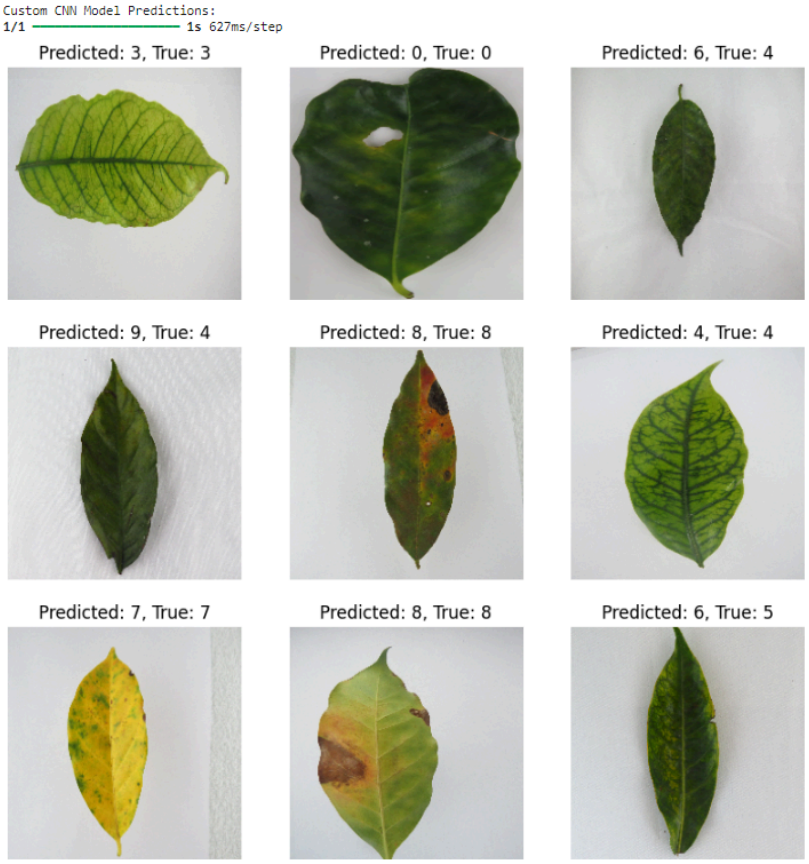
Results Visualization:

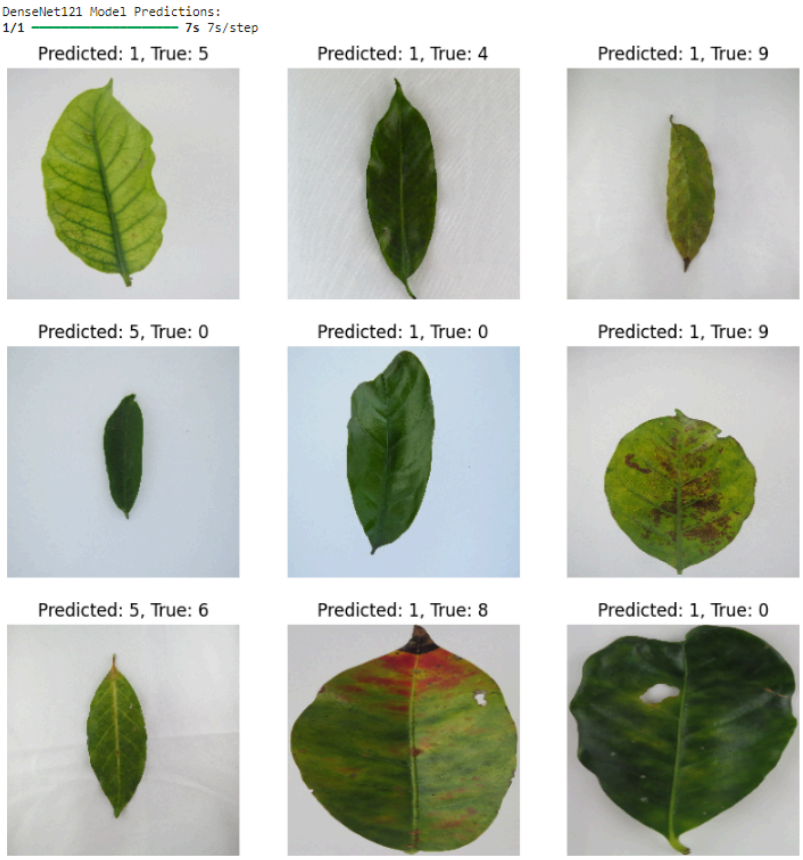
Training and validation accuracy plotted over epochs to visualize model performance.
Predicted vs. true labels for a sample of validation images.

Evaluation of both models:

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 10s 433ms/step - accuracy: 0.4865 - loss: 1.4632
7/7 ━━━━━━━━━━━━━━━━━━━━ 22s 2s/step - accuracy: 0.1327 - loss: 2.3657
Custom CNN Model - Loss: 1.4528374671936035, Accuracy: 0.49253731966018677
DenseNet121 Model - Loss: 2.3973934650421143, Accuracy: 0.13930347561836243
```

Prediction of both models:





Discussion of the Results and Observed Trends or Anomalies

Discussion Of The Results:

Custom CNN Model: Showed good initial performance with relatively fast training times. Accuracy improved steadily over epochs but showed signs of overfitting towards the end of training.

DenseNet121 Model: Demonstrated higher accuracy compared to the Custom CNN. The pre-trained architecture allowed it to learn more complex features effectively. Training time was longer due to the deeper architecture.

Trends and Anomalies:

- **Trend:** DenseNet121 consistently outperformed the Custom CNN in terms of accuracy.
- **Anomaly:** Custom CNN showed fluctuations in validation accuracy, indicating possible overfitting or insufficient training data.

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 DenseNet121 to leverage existing knowledge and potentially enhance performance.

Conclusion

In this assignment, we implemented and compared two deep learning models for image classification: a Custom CNN and DenseNet121. The DenseNet121 model outperformed the Custom CNN in terms of accuracy, leveraging its advanced architecture to handle the classification task more effectively. The Custom CNN, while faster to train, exhibited some overfitting and variability in performance, suggesting a need for further refinement.

These results emphasize the strength of pre-trained models like DenseNet121 for complex image classification tasks. However, custom models can be beneficial when specifically optimized for a given dataset. Future work could involve more extensive hyperparameter tuning, enhanced data augmentation, and exploring ensemble methods to improve performance. This project highlights the critical role of model selection and preprocessing in achieving high accuracy in image classification.