Building and Training Convolutional Neural Networks (CNN) for Image Classification

Objective:

In this assignment, you will work with datasets from Kaggle to build, train, and evaluate a Convolutional Neural Network (CNN) for image classification. The goal is to apply the concepts of CNN architecture, data preprocessing, model training, and performance evaluation.

Instructions:

1. **Dataset Selection**:
    - Visit Kaggle Datasets and select an image classification dataset. Below are some suggestions:
        - **CIFAR-10**: A dataset of 60,000 32x32 color images in 10 classes (airplane, car, bird, etc.).
        - **Fashion MNIST**: A dataset of fashion product images.
        - **Cats and Dogs**: A popular dataset of 25,000 images of cats and dogs.
        - **Plant Seedlings Classification**: Dataset of 12 crop species seedlings images.
    - **Each student** must select a different dataset. Coordinate with your peers to ensure no duplicates.
2. **Task**:
    - Your task is to build and train a CNN model to classify the images in your selected dataset into their respective categories.
3. **Assignment Breakdown**:

a) **Data Preprocessing**:

- Load and preprocess the dataset (resizing, normalization, etc.).
- Perform data augmentation to improve model generalization (e.g., rotations, flips, etc.).
- Split the data into training, validation, and test sets (80% train, 10% validation, 10% test).

b) **CNN Model Design**:

- Design a CNN architecture suitable for your dataset.
- Use common layers like convolutional layers, max-pooling, batch normalization, and dropout (if necessary).
- Ensure the output layer matches the number of classes in your dataset.

c) **Model Training**:

- Compile your model with an appropriate loss function (e.g., categorical cross-entropy for multi-class classification) and optimizer (e.g., Adam or SGD).
- Train the model on the training data and validate it on the validation set.
- Use relevant performance metrics like accuracy and loss during training.

d) **Model Evaluation**:

- Evaluate your model on the test set and report the accuracy, confusion matrix, precision, recall, and F1-score.
- Plot training and validation accuracy/loss curves.
- Analyze model performance and discuss any issues (e.g., overfitting or underfitting).

e) **Hyperparameter Tuning (Bonus)**:

- Experiment with different hyperparameters like learning rate, batch size, number of epochs, and model depth to improve model performance.

4. **Report**: Submit a report with the following sections:
   - **Introduction**: Briefly describe the dataset and its challenges.
   - **CNN Model Architecture**: Explain your CNN architecture, the choice of layers, and any design decisions.
   - **Training Process**: Describe how you trained the model, including hyperparameters and data augmentation techniques used.
   - **Results**: Present and interpret the model's performance using the test set. Include tables or charts for accuracy, loss, confusion matrix, etc.
   - **Conclusion**: Summarize the model's performance and suggest potential improvements.

5. **Code Submission**:
    - Submit your Python code as a Jupyter notebook (.ipynb file).
    - Ensure that your code is well-documented with comments explaining each section.

Resources:

- **Keras/TensorFlow**: Use these libraries for building and training CNNs.
- **Matplotlib/Seaborn**: For plotting graphs and visualizing data.
- **Scikit-learn**: For generating the confusion matrix and performance metrics.

Grading Criteria:

- **Dataset selection and preprocessing**: 15%
- **CNN architecture design**: 25%
- **Training and validation**: 20%
- **Model performance and evaluation**: 25%
- **Report quality and presentation**: 15%

Deadline:

- Submit your code and report by 14/10/2024.