Image Classification with CIFAR-10

## Title Page:

* **Project Title:** Image Classification on CIFAR-10 Dataset
* **Group Members:** Qian Yi Wang (40211303), Andrew Chebli (ID)
* **Course:** COMP472 Artificial Intelligence
* **Instructor:** Ali Ayub
* **Date:** November 25, 2024

## 1. Introduction:

#### The objective of this project is to apply various machine learning models for image classification on the CIFAR-10 dataset, which consists of 32x32 RGB images belonging to 10 different object classes. The following models were implemented and evaluated:

#### **Naive Bayes**

#### **Decision Tree**

#### **Multi-Layer Perceptron (MLP)**

#### **Convolutional Neural Network (CNN)**

#### The goal was to explore the effectiveness of each model and its variants by analyzing performance metrics such as accuracy, precision, recall, and F1-score. The project also includes comparison between different configurations for each model type.

## 2. Methodology:

##### **2.1. Dataset Preprocessing:**

The CIFAR-10 dataset consists of 32x32 pixel RGB images, which were preprocessed as follows:

* **Resizing**: The images were resized to 224x224 pixels to match the input size expected by the pre-trained **ResNet-18** model.
* **Feature Extraction**: Feature vectors of size **512x1** were extracted from the ResNet-18 model, pre-trained on ImageNet. These feature vectors were reduced to **50x1** using **Principal Component Analysis (PCA)** for dimensionality reduction.
* **Final Dataset**: The 50-dimensional feature vectors were used as input to train and test the various machine learning models.

##### **2.2. Model Architectures and Training:**

* **Naive Bayes:**
  + Implemented Gaussian Naive Bayes from scratch using Python and NumPy, followed by Scikit-learn’s implementation for comparison.
  + Trained on feature vectors of the CIFAR-10 dataset.
* **Decision Tree:**
  + A decision tree was implemented using Python and NumPy with a maximum depth of 50. Gini impurity was used for splitting.
  + The tree depth was varied to observe the effects on performance.
* **Multi-Layer Perceptron (MLP):**
  + The MLP consists of three layers: Linear(50, 512), ReLU, BatchNorm(512), Linear(512, 512), ReLU, and a final Linear(512, 10) output layer
  + Trained using cross-entropy loss and SGD with momentum=0.9.
  + Experimented with varying the number of layers and layer sizes to optimize performance.
* **Convolutional Neural Network (CNN):**
  + The VGG11 architecture was implemented, consisting of multiple convolutional layers followed by fully connected layers and dropout for regularization.
  + Trained using cross-entropy loss and SGD with momentum=0.9.

##### **2.3. Training Methodology:**

* **Epochs:** 20-50 epochs for MLP and CNN based on performance convergence.
* **Learning Rate:** 0.01 for MLP and CNN.
* **Optimizer:** SGD with momentum of 0.9.

## 3. Results:

##### **3.1. Evaluation Metrics:**

For each model and its variants, the following evaluation metrics were computed: - **Accuracy** - **Precision** - **Recall** - **F1-measure**

A table of data with numbers and letters

Description automatically generated with medium confidence

##### **3.2. Confusion Matrices:**

Confusion matrices were generated to evaluate the performance of the models, highlighting misclassifications.

* **Naive Bayes:**

A graph of a number of numbers

Description automatically generated with medium confidenceA graph with numbers and a graph

Description automatically generated with medium confidence

* **Decision Tree**

A graph of a graph with numbers

Description automatically generated with medium confidence

A graph of a graph

Description automatically generated with medium confidence

A graph of a graph

Description automatically generated with medium confidence

A graph of a diagram

Description automatically generated with medium confidence

A graph of a tree

Description automatically generated with medium confidence

A graph of a tree

Description automatically generated with medium confidence

* **Multi-Layer Perceptron (MLP)**

A graph of a network

Description automatically generated

A graph of a network depth

Description automatically generated

A graph of a network depth

Description automatically generated with medium confidence

A graph of a network

Description automatically generated with medium confidence

* **Convolutional Neural Network (CNN):**

A graph of numbers and a graph

Description automatically generated with medium confidence

A screenshot of a graph

Description automatically generated

A graph of numbers and a graph

Description automatically generated with medium confidence

A graph of numbers and a graph of numbers

Description automatically generated with medium confidence

## 4. Discussion:

* **Model Comparison:**
  + **Naive Bayes:** Performed well with a relatively simple architecture but struggled with high-dimensional image data.
  + **Decision Tree:** Depth was crucial for model performance, with deeper trees leading to better feature learning, but also overfitting.
  + **MLP:** Showed strong performance but required careful tuning of network depth and layer sizes. Smaller networks underperformed in comparison to deeper models.
  + **CNN:** The VGG11 architecture outperformed other models, demonstrating the power of convolutional layers in image classification.
* **Hyperparameter Tuning Insights:**
  + Depth (Decision Tree, MLP, CNN): Depth generally improved performance but increased the risk of overfitting in non-CNN models.
  + Layer Size (MLP): Larger hidden layers provided better performance but at the cost of increased computational complexity.
  + Kernel Size (CNN): Larger kernels helped in capturing broader features, but smaller kernels offered better spatial resolution.

## 5. Conclusion:

* sdfsdfsdfsdf