Connor Raymond Stewart

Project Report: Image Classification with CIFAR-10

1. Main Objective of the Analysis

- **Objective:**
  - The primary goal of this project was to develop and evaluate deep learning models for classifying images from the CIFAR-10 dataset into one of ten predefined categories. To achieve this, I focused on utilizing Convolutional Neural Networks (CNNs), given their exceptional performance in image-related tasks. My analysis involved experimenting with different CNN architectures to enhance classification accuracy and understand their performance nuances.
- **Benefits:**
  - This analysis is crucial for advancing automated image recognition systems, which have applications across various sectors including autonomous vehicles, medical imaging, and security systems. By developing a robust image classification model, I aimed to deliver a system that not only performs accurately but also has the potential to generalize well to other datasets, thereby providing valuable insights and tools for the business and stakeholders.

2. Data Set Description

- **Data Set Chosen:** CIFAR-10
- **Attributes:**
  - **Image Dimensions:** 32x32 pixels
  - **Color Channels:** 3 (RGB)
  - **Number of Classes:** 10
- **Class Labels:**
  - Airplane
  - Automobile
  - Bird
  - Cat
  - Deer
  - Dog
  - Frog
  - Horse
  - Ship
  - Truck
- **Objective:**
  - Train and evaluate CNN models for image classification.
  - Identify the best model architecture that balances accuracy and computational efficiency.

3. Data Exploration and Cleaning

- **Exploration:**
  - **Data Distribution:** Analyzed the distribution of images across the 10 classes to ensure balanced representation.
  - **Visual Inspection:** Displayed sample images from each class to inspect the dataset's quality and diversity.
- **Cleaning and Preprocessing:**
  - **Normalization:** Scaled pixel values to [0, 1] by dividing by 255.
  - **Augmentation:** Applied techniques like random cropping, horizontal flipping, and rotation to increase training data diversity and reduce overfitting.
  - **Splitting:** Divided dataset into:
    - Training: 50,000 images
    - Validation: 5,000 images
    - Test: 5,000 images

4. Model Training

- **Model 1: Simple Convolutional Neural Network (CNN)**
  - **Architecture:**
    - **Conv2D Layers:**
      - Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3))
      - Conv2D(64, (3, 3), activation='relu')
      - Conv2D(128, (3, 3), activation='relu')
    - **MaxPooling2D Layers:** MaxPooling2D((2, 2))
    - **Flatten Layer:** Flatten()
    - **Dense Layers:**
      - Dense(128, activation='relu')
      - Dropout(0.5)
      - Dense(10, activation='softmax')
  - **Hyperparameters:**
    - Learning Rate: 0.001
    - Batch Size: 256
    - Epochs: 10
  - **Performance:**
    - Accuracy: 69.32%
    - Loss: 0.8931
    - Macro avg Precision: 70.51%
    - Macro avg Recall: 69.32%
    - Macro avg F1-Score: 69.19%
    - Weighted avg Precision: 70.51%
    - Weighted avg Recall: 69.32%
    - Weighted avg F1-Score: 69.19%
- **Model 2: Deep Convolutional Neural Network (Deep CNN)**
  - **Architecture:**
    - **First Convolutional Block:**
      - Conv2D(64, (3, 3), activation='relu', padding='same')
      - BatchNormalization()

        - Conv2D(64, (3, 3), activation='relu', padding='same')
        - BatchNormalization()
        - MaxPooling2D(2, 2)
        - Dropout(0.3)
    - **Second Convolutional Block:**
        - Conv2D(128, (3, 3), activation='relu', padding='same')
        - BatchNormalization()
        - Conv2D(128, (3, 3), activation='relu', padding='same')
        - BatchNormalization()
        - MaxPooling2D(2, 2)
        - Dropout(0.3)
    - **Third Convolutional Block:**
        - Conv2D(256, (3, 3), activation='relu', padding='same')
        - BatchNormalization()
        - Conv2D(256, (3, 3), activation='relu', padding='same')
        - BatchNormalization()
        - MaxPooling2D(2, 2)
        - Dropout(0.4)
    - **Fully Connected Layers:**
        - Flatten()
        - Dense(512, activation='relu')
        - Dropout(0.5)
    - **Output Layer:** Dense(10, activation='softmax')
  - **Hyperparameters:**
    - Learning Rate: 0.001
    - Batch Size: 64
    - Epochs: 30
  - **Performance:**
    - Accuracy: 73.27%
    - Precision: 0.7567
    - Macro avg Precision: 74.05%
    - Macro avg Recall: 73.27%
    - Macro avg F1-Score: 73.21%
    - Weighted avg Precision: 74.05%
    - Weighted avg Recall: 73.27%
    - Weighted avg F1-Score: 73.21%
- **Model 3: Residual Network (ResNet)**
  - **Architecture:**
    - **Input Layer:** Input(shape=x_train[0].shape)
    - **First Convolutional Block:**
        - Conv2D(32, (3, 3), activation='relu', padding='same')
        - BatchNormalization()
        - Conv2D(32, (3, 3), activation='relu', padding='same')
        - BatchNormalization()
        - MaxPooling2D((2, 2))
    - **Second Convolutional Block:**

            - Conv2D(64, (3, 3), activation='relu', padding='same')
            - BatchNormalization()
            - Conv2D(64, (3, 3), activation='relu', padding='same')
            - BatchNormalization()
            - MaxPooling2D((2, 2))
        - **Third Convolutional Block:**
            - Conv2D(128, (3, 3), activation='relu', padding='same')
            - BatchNormalization()
            - Conv2D(128, (3, 3), activation='relu', padding='same')
            - BatchNormalization()
            - MaxPooling2D((2, 2))
        - **Flattening Layer:** Flatten()
        - **Fully Connected Layers:**
            - Dense(1024, activation='relu')
            - Dropout(0.2)
        - **Output Layer:** Dense(10, activation='softmax')
    - **Hyperparameters:**
        - Learning Rate: 0.0005
        - Batch Size: 128
        - Epochs: 50
    - **Performance:**
        - Accuracy: 84.11%
        - Loss: 0.9241
        - Macro avg Precision: 84.30%
        - Macro avg Recall: 84.12%
        - Macro avg F1-Score: 84.11%
        - Weighted avg Precision: 84.30%
        - Weighted avg Recall: 84.12%
        - Weighted avg F1-Score: 84.11%

5. Recommended Model

- **Recommendation:** Residual Network (ResNet-50)
- **Rationale:**
    - ResNet's residual connections address the vanishing gradient problem in deeper networks, leading to better performance compared to simpler CNN and Deep CNN models.

6. Key Findings and Insights

- **Performance Improvement:**
    - Complex models with deeper architectures and residual connections significantly improve classification accuracy.
- **Class Confusion:**
    - Difficulty in distinguishing visually similar classes such as 'deer' and 'horse,' indicating areas for further refinement.

- **Implications:**
    - Need for additional fine-tuning or more sophisticated models, potentially incorporating advanced techniques like attention mechanisms or ensemble methods.

7. Suggestions for Next Steps

- **Model Enhancement:**
    - Experiment with advanced architectures such as DenseNet or EfficientNet.
- **Feature Engineering:**
    - Incorporate additional data augmentation techniques or use transfer learning with pre-trained models on larger datasets.
- **Error Analysis:**
    - Conduct detailed error analysis to understand misclassifications and refine the model or dataset.
- **Future Research:**
    - Explore semi-supervised or unsupervised learning techniques to leverage additional unlabeled data for further performance improvements.