Project Title: **CIFAR-10 Image Classification Using Convolutional Neural Networks (CNN) and Distributed Computing**

Team Members and Individual Contributions:

1. Varshitha Reddy Davarapalli

- **Contribution**: Designed the CNN model architecture, implemented data augmentation strategies, and tuned hyperparameters to improve model accuracy and prevent overfitting.
- **Details**: Developed the core CNN structure, ensuring efficient layer setup and adding data augmentation to enhance generalizability. Focused on configuring dropout layers, batch normalization, and early stopping for better model performance on the CIFAR-10 dataset.

2. Saketha Kusu

- **Contribution**: Managed data preprocessing and splitting, setting up the Spark environment, and integrating the CNN model with distributed computing.
- **Details**: Handled data loading and normalization, organized the Spark configuration for efficient model distribution, and managed training on multiple VMs to accelerate processing. Optimized data handling for Spark and Python interoperability.

3. Karunakar Uppalapati

- **Contribution**: Performed the evaluation, generated performance metrics, and created visualization tools for tracking model training, validation, and testing results.
- **Details**: Implemented accuracy, confusion matrix, and loss visualization tools, developed evaluation scripts, and documented the results. Focused on monitoring model improvement across epochs and ensured reliable reporting of accuracy and other key metrics.

Project Overview

Problem Statement

This project aims to classify images in the CIFAR-10 dataset into one of 10 classes using a Convolutional Neural Network (CNN). The goal is to apply distributed computing via Spark on multiple VMs to manage computational demands and scalability for efficient model training.

Challenges

- **Training Efficiency**: Managing large image datasets with distributed computing.
- **Generalization**: Improving model generalizability to avoid overfitting.
- **Hyperparameter Tuning**: Optimizing CNN performance in a distributed environment.

Implementation Plan

Dataset and Source

- **Dataset**: CIFAR-10, containing 60,000 images across 10 classes (6,000 images per class).
- **Source**: CIFAR-10 dataset page.
- **Class Labels**: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.

CNN Model Architecture

We propose designing the CNN model with the following layers:

Convolutional Layers:

- **Layer 1**: 32 filters, 3x3 kernel, ReLU activation, followed by max pooling.
- **Layer 2**: 64 filters, 3x3 kernel, ReLU activation, followed by max pooling.
- **Layer 3**: 128 filters, 3x3 kernel, ReLU activation, followed by max pooling.

Flattening Layer:

To convert the final convolutional output into a 1D array for fully connected layers.

Fully Connected Layers:

- **Dense Layer**: 128 units, ReLU activation, followed by dropout to prevent overfitting.
- **Output Layer**: 10 units with softmax activation for classification probabilities.

Hyperparameters and Tuning

- **Initial Settings**: Learning rate of 0.001, batch size of 64, and 30 epochs.
- **Tuning Strategy**: We will use grid search to optimize dropout rates and learning rates, managing overfitting and improving convergence.

Techniques for Overfitting Prevention:

- **Data Augmentation**: Apply transformations (rotations, width/height shifts, and flips) to create diverse training samples and improve model generalization.

- **Dropout**: Use dropout in dense layers (rate 0.5) to reduce neuron dependency and avoid overfitting.
- **Early Stopping**: Monitor validation loss and halt training if no improvement occurs over five epochs.

Distributed Computing with Spark and VM Setup

We plan to set up Spark across multiple VMs to enable parallel data processing and training.

- **Spark Configuration**: Configure Spark with 2 GB executor memory and 2 cores per executor.
- **Data Distribution**: Distribute images across Spark executors to enable batch processing, reducing training time by leveraging Spark's parallel processing.

Data Preprocessing and Splitting

- **Normalization**: Scale pixel values to a range of 0-1 to improve model convergence.
- **Data Splitting**: Divide data into training (50,000 images) and testing (10,000 images) sets with an 80-20 split.
- **One-Hot Encoding**: Convert labels into one-hot encoded vectors for compatibility with the softmax layer.

Model Training and Evaluation

- **Training**: Conduct training on Spark with 30 epochs and monitor validation loss for early stopping.
- **Testing**: Aim for a target accuracy, using evaluation metrics to analyze performance on the test set.

Evaluation Metrics

- **Precision, Recall, F1-Score**: To assess model accuracy per class.
- **Confusion Matrix**: To provide insights into misclassifications for a detailed analysis.

Visualization Tools

- **Accuracy Plot**: To show training and validation accuracy over epochs.
- **Loss Plot**: To visualize training and validation loss, monitoring convergence and overfitting.

Advanced Techniques and Future Work

For future improvements, we propose the following:

- **Transfer Learning**: Utilize pretrained models to potentially improve accuracy with reduced training time.
- **Explainability**: Implement Grad-CAM or similar techniques to visualize regions influencing predictions, making the model more interpretable.

Expected Results and Findings

We anticipate achieving a satisfactory test accuracy, demonstrating the CNN's capacity to classify images in the CIFAR-10 dataset. Our evaluation will include detailed performance metrics, providing insights into the model's effectiveness across the 10 classes. Additionally, we aim to validate the efficiency and scalability of our distributed Spark setup for large-scale image classification.

Conclusion

Through this project, we aim to demonstrate the viability of distributed computing and CNNs for image classification, leveraging Spark to optimize the training pipeline. This approach will establish a scalable framework for future work, including advanced architectures and interpretability techniques for broader applications.

Appendix: Code and Results Documentation

- **Code Snippets**: We will provide implementation details for the CNN architecture, Spark configuration, and hyperparameter settings.
- **Evaluation Report**: Our final report will document test results, accuracy scores, and classification metrics in a dedicated file for reference.

Saketha Kusu

Title: Distributed Computing for CIFAR-10 Image Classification Using CNN and Spark

Team Members: Varshitha Reddy Davarapalli, Saketha Kusu, Karunakar Uppalapati
Individual Contribution: I am responsible for configuring the Spark environment, handling data distribution across VMs, and optimizing data processing for efficient model training.

Project Goals and Problem Statement

This project aims to classify CIFAR-10 images into 10 categories using a CNN model. A significant aspect of this project is to handle the training on a distributed computing environment, optimizing resource use for faster processing.

Specific Tasks

1. **Spark Configuration: Set up and configure Spark across multiple VMs, specifying memory and core allocations for efficient training.**
2. **Data Distribution: Divide and manage data distribution across executors, enabling parallel processing.**
3. **Data Preprocessing: Normalize image data and convert labels to a format compatible with the model, enabling seamless integration with Spark.**

Implementation Strategy

I will implement Spark session initialization with appropriate memory and core configurations to balance resource use. Batch processing will be managed to keep data flow efficient across distributed executors.

Performance Evaluation

The model's performance will be assessed based on processing speed improvements and the accuracy achieved through distributed learning.

Dataset : https://www.cs.toronto.edu/~kriz/cifar.html