Project Summary: CIFAR-10 Image Classification Using Keras Tuner

(by ATUMONYE JAMES CHUKWUEMEKA)

Introduction

In the age of artificial intelligence and deep learning, image classification stands out as one of the fundamental tasks, widely applicable in various fields including computer vision, medical imaging, and autonomous driving. This project focuses on building and optimizing a Convolutional Neural Network (CNN) to classify images from the CIFAR-10 dataset. The CIFAR-10 dataset is a well-known benchmark in the machine learning community, consisting of 60,000 32x32 color images in 10 different classes with 6,000 images per class.

Data Source

The CIFAR-10 dataset, created by Alex Krizhevsky and Geoffrey Hinton, is publicly available and can be easily accessed via the TensorFlow and Keras libraries. This dataset is widely used for training machine learning and computer vision algorithms due to its manageable size and variety of image classes.

Aims and Objectives

- **Aim**: To build a robust Convolutional Neural Network (CNN) model for accurately classifying images in the CIFAR-10 dataset.
- **Objectives**:
    1. Preprocess the CIFAR-10 dataset to make it suitable for model training.
    2. Develop a CNN model architecture capable of learning complex image features.
    3. Optimize the hyperparameters of the CNN using Keras Tuner for improved performance.
    4. Evaluate the model's performance on training and validation data.
    5. Visualize the training process and model performance metrics.

Methodology

1. **Data Preprocessing**:
    - Loaded the CIFAR-10 dataset using tensorflow.keras.datasets.
    - Normalized the pixel values of the images to fall between 0 and 1 to facilitate better convergence during training.
    - Flattened the labels to match the model output format.

2. **Model Architecture**:
    - Constructed a Convolutional Neural Network (CNN) using the tensorflow.keras library.
    - The architecture includes three convolutional layers with ReLU activation functions and MaxPooling layers to down-sample the image dimensions.
    - Added a fully connected Dense layer followed by a Dropout layer to prevent overfitting.
    - The final output layer is a Dense layer with 10 units, representing the 10 classes of the CIFAR-10 dataset.
3. **Hyperparameter Tuning**:
    - Employed Keras Tuner's RandomSearch strategy to identify the optimal hyperparameters.
    - Defined a search space for hyperparameters including the number of units in each convolutional layer, the learning rate of the optimizer, the number of units in the dense layer, and the dropout rate.
    - Configured the tuner to run multiple trials and select the best model based on validation accuracy.
4. **Training the Model**:
    - Initialized the RandomSearch tuner and conducted hyperparameter tuning with a maximum of 10 trials.
    - Trained the model using the optimal hyperparameters over 10 epochs.
    - Evaluated the model on the validation set to monitor performance improvements and potential overfitting.
5. **Performance Evaluation**:
    - Monitored training and validation accuracy and loss over epochs to assess the learning curve.
    - Visualized the training history using Matplotlib to provide insights into model performance and optimization.

Results

The optimized CNN model achieved a training accuracy of 75.64% and a validation accuracy of 71.93% after 10 epochs. The training loss decreased significantly, indicating effective learning, while the validation loss also showed a downward trend, albeit with slight overfitting.

Conclusion

This project demonstrates the successful application of deep learning techniques to image classification tasks using the CIFAR-10 dataset. Through systematic data preprocessing, model development, and hyperparameter tuning, we achieved a model

that performs well on unseen data. Future work could involve implementing additional regularization techniques, exploring different model architectures, and further increasing the dataset's diversity through data augmentation.

Visualizations

To illustrate the model's performance, the following plots were generated:

1. **Training and Validation Accuracy**: A plot showing the improvement in accuracy over epochs, highlighting the model's learning ability.
2. **Training and Validation Loss**: A plot demonstrating the reduction in loss, indicating better optimization of the model.

Call to Action

Feel free to reach out if you have any questions or are interested in collaborating on similar projects. Let's connect and share our passion for machine learning and artificial intelligence!