AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH

Assignment Title: CNN_CIFAR10	
Assignment No: 01	Date of Submission: 31-07-23
Course Title: Computer Vision and Patter Recognition	Semester: Summer 2022-23
Course Code:	Section: C
Submitted by: Ridita Zaman Adikta Id: 20-43679-2	Submitted To: Dr. Debajyoti Karmaker

Explanations of the CNN model, CIFAR10 dataset loading and plotting

Importing necessary libraries:

We import the required libraries for building and training the CNN model, loading the CIFAR-10 dataset, and plotting the results.

Loading the CIFAR-10 dataset:

We use the cifar10.load_data() function from Keras to load the CIFAR-10 dataset. The dataset consists of 50,000 training images and 10,000 test images, each with a corresponding label from 10 different classes (e.g., airplane, automobile, bird, etc.). We then normalize the pixel values of the images to be in the range [0, 1] by dividing them by 255.

Task 1: Applying different optimizers:

We define a basic CNN model architecture using the build_basic_cnn_model() function. This function creates a simple CNN with three Conv2D layers, followed by MaxPooling2D layers, a Flatten layer, and two Dense layers.

The train_model_with_optimizer() function is used to train the model with different optimizers. It takes an optimizer and its parameters (learning_rate and momentum) as inputs, compiles the model with the specified optimizer and loss function, and then trains the model on the CIFAR-10 training dataset for 10 epochs. The training process is performed using the fit() function. The training history (accuracy and loss over epochs) and the trained model are returned.

We then train the model using three different optimizers: Stochastic Gradient Descent (SGD), Adam, and RMSprop. We visualize the accuracy and loss trends for each optimizer using Matplotlib.

Task 2: Effect of using regularizes (L2):

We define a new CNN model architecture with L2 regularization on the Conv2D layers using the build_cnn_with_l2_regularization() function. L2 regularization is applied to the kernel weights of the Conv2D layers to prevent over fitting.

The train_model_with_l2_regularization() function is used to train the L2-regularized model. It takes an optimizer as input (default is 'Adam'), compiles the model with the specified optimizer and loss function, and then trains the model on the CIFAR-10 training dataset for 10 epochs.

We then compare the performance of the L2-regularized model with the basic model (without regularization) by plotting the accuracy and loss trends over epochs.

Task 3: Comparison of using data preprocessing vs no preprocessing:

We define another CNN model without data preprocessing (normalization) using the build_cnn_model_without_preprocessing() function.

The train_model_without_preprocessing() function is used to train the model without data preprocessing. It takes an optimizer as input (default is 'Adam'), compiles the model with the specified optimizer and loss function, and then trains the model on the CIFAR-10 training dataset for 10 epochs.

We then compare the performance of the model with data preprocessing (normalized pixel values) to the model without preprocessing by plotting the accuracy and loss trends over epochs.

Plotting the results:

Finally, we use Matplotlib to plot the accuracy and loss trends for each task. The plots are shown side by side for easy comparison. The x-axis represents the number of training epochs, and the y-axis represents accuracy or loss values. The different lines in the plots represent different optimizers, regularization techniques, or preprocessing methods.

By executing this code, we can see the effects of different optimizers, the impact of L2 regularization, and the benefits of data preprocessing on the CNN model's performance in classifying the CIFAR-10 dataset.