CS610

Applied Machine Learning

Assignment 2

**[Dataset] CIFAR-10**

**Description:**

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is divided into 50,000 training images and 10,000 testing images. The classes are mutually exclusive and there is no overlap between them.

**Features:**

**Number of Instances:** 60,000 (50,000 for training, 10,000 for testing)

**Image Size:** 32x32 pixels, color images (3 color channels)

**Number of Classes:** 10

**Classes:**

The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

**Accessibility:**

The CIFAR-10 dataset is widely used for machine learning and computer vision research and can be easily accessed through libraries like TensorFlow and PyTorch.

**Sample Code to Load the CIFAR-10 Dataset with TensorFlow:**

import tensorflow as tf

# Load the CIFAR-10 dataset

cifar10 = tf.keras.datasets.cifar10

(x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()

# Exploring the dataset

print("Training set shape:", x\_train.shape)

print("Test set shape:", x\_test.shape)

**[Question 1] Data Normalizaton (4 marks)**

To effectively prepare the CIFAR-10 dataset for use with a neural network model, it is essential to normalize the data. Normalization plays a pivotal role in preprocessing, enhancing the efficiency of gradient descent, mitigating numerical instability issues such as exploding gradients, and speeding up the training process. This task involves normalizing the data in two distinct manners:

1. **Normalization to [0,1]:** This method involves scaling the pixel values of the images to fall within the [0,1] range, using Min-Max Scaling:
2. **Normalization to Following a Normal Distribution:** This approach requires adjusting the data to have a mean of 0 and a standard deviation of 1. Prior to applying this normalization, you must calculate the mean and standard deviation of the CIFAR-10 training dataset's pixel values.

**Implement the Two Normalization Methods:** Write the necessary code to apply both normalization methods to the CIFAR-10 dataset.

**Application for Subsequent Questions:** For the questions that follow, you will only need to utilize one of these normalization methods to process the data. Typically, method (1) is preferred when working with TensorFlow, while method (2) is commonly used in PyTorch environments.

**[Question 2] Implementing a Multi-Layer Perceptron for CIFAR-10 Classification (10 marks)**

***Objective:*** *Build and train a Multi-Layer Perceptron (MLP) to classify images from the CIFAR-10 dataset, and achieve* ***>50% accuracy****.*

**Dataset Preparation:** Load and normalizae the CIFAR-10 dataset. Flatten the images to use them with an MLP, as MLPs do not take the spatial structure of the images into account.

**Model Architecture:** Design an MLP with at least one hidden layer. Choose an appropriate activation function for the hidden layer (e.g., ReLU) and the output layer (e.g., softmax for multi-class classification).

**Training:** Compile the model with an appropriate loss function (e.g., categorical crossentropy) and optimizer. Train the model on the CIFAR-10 training set and validate its performance on the test set.

**Evaluation:** Report the accuracy of the model on the CIFAR-10 test set. Discuss the model's performance and any potential improvements.

**Hyperparameter Tuning**: Besides learning rate, batch size, and number of epochs, number and size of layers of MLP are also hyperparameters that can be tuned. Adding more layers and/or increasing the number of neurons per layer can enable the modeling of more complex functions, but this also increases the risk of overfitting and computational costs. It's advisable to start with simpler models and gradually increase complexity. Try the following architectures using ReLU as the activation function for hidden layers and softmax as the activation function for the output layer:

1. MLP with one hidden layer consisting of 128 neurons.
2. MLP with one hidden layer consisting of 256 neurons.
3. MLP with two hidden layers consisting of 256 and 128 neurons, respectively.

If none of the above architectures are able to achieve an accuracy greater than 50%, consider increasing the complexity of the MLP by adding more hidden layers or neurons. You may also experiment with other activation functions to achieve accuracy greater than 50%.

**[Question 3] Building a Convolutional Neural Network for CIFAR-10 (10 marks)**

***Objective:*** *Develop a Convolutional Neural Network (CNN) to improve the classification performance on the CIFAR-10 dataset, and achieve* ***>70% accuracy****.*

**Dataset Preparation:** Load and normalize the CIFAR-10 dataset. Ensure the images are normalized to a range suitable for CNN processing.

**CNN Architecture:** Design a CNN that includes convolutional layers, activation functions, pooling layers, and fully connected layers. Detail your choice of kernel sizes, pooling sizes, and the architecture's depth.

**Training:** Compile your CNN with a suitable loss function and optimizer. Utilize techniques such as dropout and batch normalization to prevent overfitting and ensure more stable training.

**Evaluation and Comparison:** Report the accuracy of the model on the CIFAR-10 test set. Compare the performance of your CNN to the MLP model from Question 2, discussing why the CNN performs differently.

**[Question 4] Leveraging a Pretrained Model for CIFAR-10 (10 marks)**

***Objective:*** *Use a pretrained model from TensorFlow's Keras applications or PyTorch's torchvision models as a feature extractor or for fine-tuning on the CIFAR-10 dataset, and achieve* ***>90% accuracy*** *on the test dataset of CIFAR-10.*

**Dataset Preparation:** Load the CIFAR-10 dataset, applying normalization and any required preprocessing to match the input format of the pretrained model.

**Model Selection and Modification:** Choose a pretrained model (e.g., ResNet, VGG16) and modify it for CIFAR-10 classification. Describe how you adapt the model for the 10 classes of CIFAR-10 (e.g., modifying the top layer, adjusting input size).

**Training Strategy:** Decide whether to freeze the weights of the pretrained layers and only train the top layer(s), or to fine-tune the entire network. Justify your choice based on the CIFAR-10 dataset's characteristics.

**Evaluation:** Test the model on the CIFAR-10 test set and report the accuracy. Compare the results with the MLP and CNN models from the previous questions, discussing the advantages and limitations of using pretrained models on a dataset like CIFAR-10.

**Deliverables (6 marks)**

* A Jupyter Notebook containing the code for the model implementation, training, and evaluation.
* A brief report (200-300 words) discussing the model design choices, training process, challenges faced, and a comparison of the model's performance with the models from the other questions.
* Suggestions for further improving the model's performance on the CIFAR-10 dataset.