ASSIGNMENT

GitHub Link: https://github.com/BXP11850/Neural-Networks-Deep-Learning-Assignments-BXP11850/tree/main/Assignment%20-5

Bhavana Parasa  
700761185

**VGG16:**

**Code:**

**A screenshot of a computer

Description automatically generated**

**Explanation:**

1. Random Seed Fixation: Ensures reproducibility by setting a consistent initial random state for NumPy and TensorFlow.

2. CIFAR-10 Dataset Loading and Preprocessing: Normalizes images to [0, 1] and converts labels to one-hot encoding.

3. Model Configuration: Sets up input shape, batch size, number of classes, and training epochs.

4. CNN Model Building: Uses a VGG-inspired architecture with multiple convolutional, max pooling layers, and dense layers with dropout for regularization.

5. Model Compilation: Utilizes Adam optimizer, categorical cross entropy loss, and tracks accuracy metric.

6. Model Training: Trains the model on the preprocessed training data with validation split for monitoring.

7. Evaluation: Assesses model performance on the test set to obtain loss and accuracy.

8. Visualization: Displays an image from the test dataset.

9. Prediction: Predicts the class of a test image and compares it to the true label.

10. Model Saving: Saves the trained model to disk for future use.

**Output:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**VGG19:**

**Code:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Explanation:**

1. Library Imports: Code to import libraries and set up the environment for deep learning with TensorFlow and other helper libraries.

2.Dataset Loading: Loads the CIFAR-100 dataset into training and testing sets (note: there seems to be a mismatch in the summary referring to CIFAR-100 and the code snippet loading CIFAR-10, which might be a typo).

3. Class Visualization: Defines a list of class names and includes a visualization block to display images from the dataset.

4. Image Display: A loop to plot a subset of images from the training set, showing the variety within the dataset.

5.Data Splitting: Uses train\_test\_split from sklearn to further divide the training data into training and validation sets.

**Output:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**LeNet & AlexNet:**

**Code:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Explanation:**

1. LeNet Architecture: This part of the notebook likely introduces the LeNet architecture, one of the earliest convolutional neural networks (CNNs) that significantly impacted the field of deep learning. LeNet is designed for handwritten and machine-printed character recognition. It's a relatively simple model consisting of convolutional layers, subsampling (pooling) layers, and fully connected layers.

2. AlexNet Architecture: This section probably covers AlexNet, a more advanced CNN that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. AlexNet is known for its deeper architecture compared to LeNet, featuring several convolutional layers with large kernel sizes in the first layer, followed by max pooling, normalization layers, and fully connected layers at the end. It uses ReLU as its non-linear activation function.

3. Implementation Details: The notebook likely includes code to implement these architectures using a deep learning framework like TensorFlow or PyTorch. This includes defining the model layers, the activation functions, the optimizer, and the loss function.

4. Training the Models: There would be sections dedicated to training LeNet and AlexNet models on a dataset (possibly MNIST for LeNet and CIFAR-10 or ImageNet for AlexNet). This includes loading the dataset, preprocessing the data (normalizing, resizing, converting to tensors), defining training parameters (batch size, number of epochs), and running the training loop.

5. Evaluation and Results: After training, the notebook probably evaluates the models on a test dataset to assess their performance. Metrics such as accuracy, precision, recall, and F1 score might be used for evaluation. The results section might compare the effectiveness of LeNet and AlexNet in handling different types of images.

6. Visualization: To help understand the model's performance and decision-making process, there could be visualizations such as loss and accuracy curves over epochs, confusion matrices, or even feature maps from convolutional layers to see what the network learns at various stages.

7. Conclusion and Discussion: The notebook might conclude with a discussion of the results, insights gained from training and evaluating the models, challenges encountered, and possible improvements or alternatives for future exploration.

**Output:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**