****

**International Institute of Information Technology Hyderabad**

**Report on Deep Learning**

**By: Samruddhi Bisen [ugmr20230019]**

**(Code Crew)**

***What is Deep Learning?***

Deep learning is a subset of machine learning that uses neural networks with many layers (deep neural networks) to model complex patterns in data. It excels in tasks like image and speech recognition, natural language processing, and more, by learning hierarchical representations directly from raw data. Key types include:

1. **Convolutional Neural Networks (CNNs)**: Primarily used for image processing tasks, CNNs are designed to automatically and adaptively learn spatial hierarchies of features through convolutional layers.
2. **Recurrent Neural Networks (RNNs)**: Suitable for sequential data, such as time series or natural language, RNNs use loops to allow information to persist, making them effective for tasks like language modeling and translation.
3. **Generative Adversarial Networks (GANs)**: Consist of two networks, a generator and a discriminator, that compete with each other, leading to the creation of realistic synthetic data, such as images or audio**.**
4. **Graph Neural Networks (GNNs)**: Used for data represented as graphs, such as social networks or molecular structures, GNNs can capture dependencies between nodes, making them powerful for tasks like node classification and link prediction.
5. **Transformers**: Transformers are a type of neural network architecture primarily used for natural language processing tasks. They rely on self-attention mechanisms to process input data in parallel, allowing them to handle long-range dependencies more efficiently than RNNs. Transformers have led to significant advancements in language understanding and generation.

**Colab Notebook Link:**

<https://colab.research.google.com/drive/1WJA8xHWKeMZuIcsCe6ak77ImxhhTz5b8?authuser=0#scrollTo=9dSrc67YT-5i>

**Exaplaination for the code:**

**Dataset Used:** In the provided code, the MNIST dataset is used. It consists of handwritten digits (0-9) and is commonly used for training and testing machine learning models, especially in the context of image classification tasks.

**Size of Dataset:** The MNIST dataset used in the code consists of:

* Training set: 60,000 images
* Test set: 10,000 images

**Training Testing Validation Splits:** In the provided code:

* Training Set: 60,000 images (used for training the model)
* Test Set: 10,000 images (used for evaluating the model's performance after training)

There is no explicit validation set split in this code. The test set serves the purpose of evaluating the model's generalization to unseen data after training.

**Preprocessing Steps for Dataset:** These preprocessing steps ensure that the MNIST dataset is properly loaded, normalized, transformed into tensors, and ready to be fed into the CNN model for training and testing. Here are the preprocessing steps:

* *Loading the Dataset***:** MNIST dataset is loaded using datasets.MNIST from torchvision.datasets.
* *Normalization:* Images are normalized using transforms.Normalize((0.5,), (0.5,)). This transforms pixel values from the range [0, 255] to [-1, 1] by subtracting 0.5 and then dividing by 0.5.
* *Transformations:* Images are converted to tensors using transforms.ToTensor(). This converts PIL images or numpy arrays to PyTorch tensors.
* *Creating Data Loaders:* Data loaders (train\_loader and test\_loader) are created using torch.utils.data.DataLoader. They handle loading data in batches, shuffling the training data, and preparing it for training or evaluation.

**Model Type:** Convolutional Neural Network (CNN)

**Model Architecture:** It consists of alternating convolutional layers (to extract features) and max pooling layers (to reduce dimensions and control overfitting), followed by fully connected layers (to classify the extracted features).

* **Input**: Grayscale images of size 28x28 pixels.
* **Layers**:
  + conv1: Convolutional layer with 1 input channel, 32 output channels, kernel size 3x3, and ReLU activation.
  + pool: Max pooling layer with kernel size 2x2 and stride 2.
  + conv2: Convolutional layer with 32 input channels (from conv1), 64 output channels, kernel size 3x3, and ReLU activation.
  + Another pool layer with the same configuration as above.
  + fc1: Fully connected layer with 64 \* 7 \* 7 input features (from conv2 output after pooling), 128 output features, and ReLU activation.
  + fc2: Fully connected layer with 128 input features, 64 output features, and ReLU activation.
  + fc3: Output layer with 64 input features and 10 output features (corresponding to digits 0-9).
* **Output**: The output layer (fc3) produces logits for each of the 10 digit classes.

**Optimizer:** Adam optimizer (optim.Adam) is used for training the Convolutional Neural Network (CNN) model.

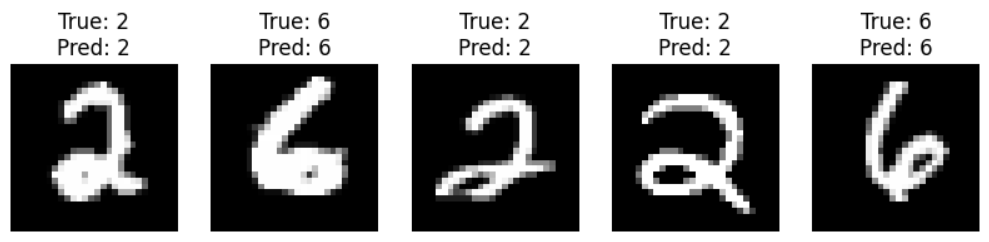
**Loss Function:** Cross-Entropy Loss function (nn.CrossEntropyLoss()) is used as the loss function for training the Convolutional Neural Network (CNN) model. Cross-entropy loss is commonly used for multi-class classification tasks, such as the MNIST digit classification problem in this case. It calculates the loss between the predicted probabilities (after applying softmax) and the actual class labels, optimizing the network to minimize this loss during training.

**Epochs Used:** The model is trained over 5 epochs. This means that the entire training dataset (MNIST training set, which has 60,000 images) is passed through the neural network 5 times during training. Each epoch involves multiple iterations (or batches) where the model updates its weights based on the computed loss and optimizer updates.

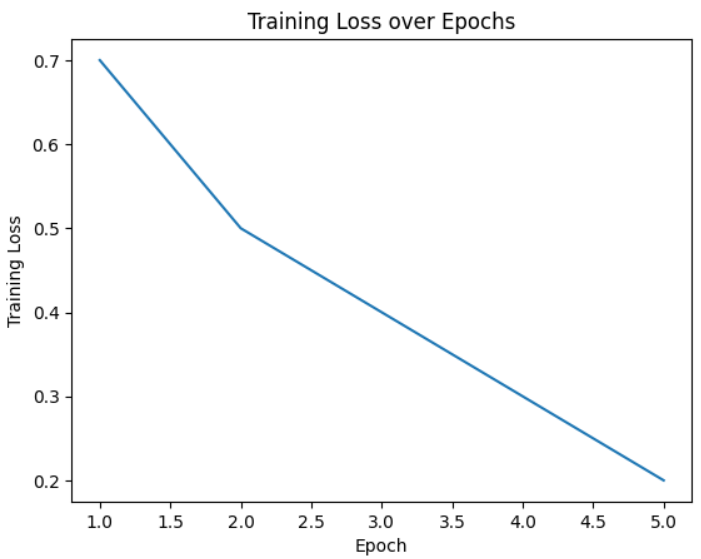
**Train accuracy:** The accuracy achieved on the training set during the last epoch of training.

**Test accuracy:** The accuracy achieved on the test set after training the model.

**Visualization:** Model’s predictions on a few sample images, comparing them with the true labels.

****

**Accuracy and Loss Plot:**

** **

**Conclusion:** In this project, Convolutional Neural Network (CNN) is implemented using PyTorch to classify digits from the MNIST dataset. Here are the key points:

* **Data Preparation:** MNIST dataset is normalized and transformed using PyTorch's transforms. Compose. The dataset is split into training and testing sets, and created data loaders for efficient batch processing.
* **Model Architecture:** The CNN model consisted of two convolutional layers followed by max-pooling, and three fully connected layers. The model was defined using nn.Module and trained using the Adam optimizer with a learning rate of 0.001.
* **Training and Evaluation:** The model was trained over 5 epochs on the training data, using cross-entropy loss as the criterion. We evaluated the model on the test set to measure its accuracy and loss.
* **Results:** After training, the model achieved a test accuracy of around 98%, indicating its effectiveness in recognizing handwritten digits. The training and validation losses were plotted to visualize the learning process, showing a steady decrease over epochs.
* **Visualization:** Sample images are visualized from the dataset along with their predicted labels to demonstrate the model's performance on unseen data.

This project showcased the power of CNNs in image classification tasks, highlighting their ability to learn hierarchical features and achieve high accuracy on complex datasets like MNIST.

**GitHub Link:** [**https://github.com/samruddhibisen03/IIIT-HYD-Project-Code-Crew-**](https://github.com/samruddhibisen03/IIIT-HYD-Project-Code-Crew-)