****

**International Institute of Information Technology Hyderabad**

**Report on Deep Learning**

**By: Aniket Rajesh Mishra [ugmr20230027]**

**(Code Crew)**

**Abstract:** This document details the implementation of a Convolutional Neural Network (CNN) using PyTorch to classify items from the Fashion MNIST dataset. The project covers data preprocessing, model architecture, training, evaluation, and results analysis.**Introduction:**

**Deep learning** is a type of machine learning that uses neural networks with multiple layers (deep neural networks) to understand complex patterns in data. It's particularly good at tasks like recognizing images, understanding speech, and processing natural language, directly from raw data.

Key Types of Deep Learning:

1. **Convolutional Neural Networks (CNNs)**: Specialized for image tasks, they learn patterns like edges and textures using layers that process small regions of the image at a time.
2. **Recurrent Neural Networks (RNNs)**: Great for sequences like text or time-series data. They use loops to remember information, making them useful for tasks like predicting the next word in a sentence or forecasting stock prices.
3. **Generative Adversarial Networks (GANs)**: Composed of two networks that compete. One generates new data (like images), and the other tries to distinguish between real and fake data, improving over time.
4. **Graph Neural Networks (GNNs)**: Designed for data structured as graphs (like social networks or molecules). They learn from the relationships between nodes to predict connections or classify elements.
5. **Transformers**: Primarily used for processing language. They use self-attention to understand how different words relate to each other in a sentence, allowing them to handle complex language tasks like translation and summarization efficiently.

**Methods:**

**Dataset:** Type: Fashion MNIST

 **Size:** The dataset comprises 70,000 images in total, with 60,000 images allocated for training and 10,000 for testing the model's performance.

 **Image Dimensions**:Each image in the dataset is grayscale and has a resolution of 28x28 pixels, making it relatively small compared to other image datasets, which helps in faster training and testing.

 **Classes:** There are 10 classes in the Fashion MNIST dataset, each representing a specific type of clothing or accessory, such as T-shirt/top, Trouser, Pullover, etc.

**Data Preparation:**

* **Train-Test Split:**
  + The dataset is divided into a training set of 60,000 images and a test set of 10,000 images. This division ensures that the model is trained on one set of data and evaluated on unseen data to assess its generalization ability.
* **Transformations:**
  + Images are transformed into PyTorch tensors, which are multi-dimensional arrays similar to numpy arrays but optimized for deep learning operations. This transformation allows efficient computation on GPU hardware, speeding up the training process.
  + Normalization: The pixel values of the images are normalized to have a mean of 0.5 and a standard deviation of 0.5. This step ensures that each pixel value is scaled within a range that helps in efficient convergence during model training.

**Preprocessing Steps:**

1. **Download the dataset:**
   * The Fashion MNIST dataset is downloaded using PyTorch's torchvision module, which provides access to popular datasets and pretrained models for deep learning.
2. **Normalize the images:**
   * Normalization involves adjusting the pixel values of the images so that they fall within a standardized range. In this case, each pixel value is scaled to have a mean of 0.5 and a standard deviation of 0.5. This normalization simplifies the optimization process by keeping input values within a similar numerical range.
3. **Create data loaders for batch processing:**
   * Data loaders are utility classes in PyTorch that provide efficient batching, shuffling, and parallel loading of data during training and evaluation. They help manage large datasets and optimize memory usage by loading data in batches rather than all at once.

**Model:**

* **Architecture: Convolutional Neural Network (CNN)**
  + CNNs are well-suited for image classification tasks because they can automatically learn spatial hierarchies of features through their layers.
* **Layers:**
  + **Convolutional Layers:** These layers apply a set of filters (kernels) to input images to extract relevant features.
  + **Max-Pooling Layers:** These layers reduce the spatial dimensions of the feature maps, retaining the most important information.
  + **Fully Connected (FC) Layers:** These layers process the flattened output from the convolutional layers to make final predictions.

**Model Architecture:**

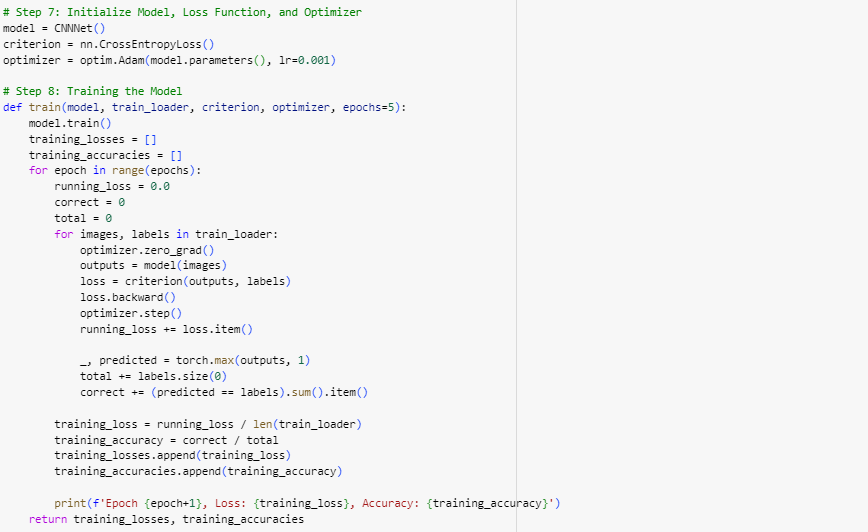
1. **Conv Layer 1:**
   * Input: 1 channel (grayscale images)
   * Output: 32 channels (feature maps)
   * Kernel Size: 3x3
2. **Conv Layer 2:**
   * Input: 32 channels (from previous layer)
   * Output: 64 channels
   * Kernel Size: 3x3
3. **Max Pooling:**
   * Kernel Size: 2x2
4. **Fully Connected Layers (FC Layers):**
   * **Layer 1:** Input features: 64 \* 7 \* 7 (output from the second convolutional layer), Output features: 128
   * **Layer 2:** Input features: 128, Output features: 64
   * **Layer 3:** Input features: 64, Output features: 10 (one for each class)

**Optimizer and Loss Function:**

* **Optimizer: Adam**:Adam is an adaptive learning rate optimization algorithm that combines the advantages of AdaGrad and RMSProp.
* **Learning Rate: 0.001**:The learning rate determines how much the model parameters are adjusted during training in response to the estimated error.
* **Loss Function: Cross-Entropy Loss**:Cross-entropy loss is commonly used in classification tasks where the model outputs a probability distribution over classes. It measures the difference between the predicted probability distribution and the actual distribution (one-hot encoded labels in this case)

**Training Parameters: 1.Epochs:** An epoch refers to one complete pass through the entire training dataset. Training for 5 epochs means that the model has iterated over the entire dataset 5 times during the training phase.

**2.Batch Size:**Batch size defines the number of training examples utilized in one iteration. During training, a batch size of 64 images is processed in parallel, while during testing, 1000 images are processed together. Batch processing helps in optimizing memory usage and computational efficiency.

**Code Section from my notebook**: 

**Notebook colab link for detailed description of this code :** [**https://colab.research.google.com/drive/1saHI1xHZAhP51NCWi5VGOwvgXfAuyzw4?usp=drive\_link**](https://colab.research.google.com/drive/1saHI1xHZAhP51NCWi5VGOwvgXfAuyzw4?usp=drive_link)

**Results:**

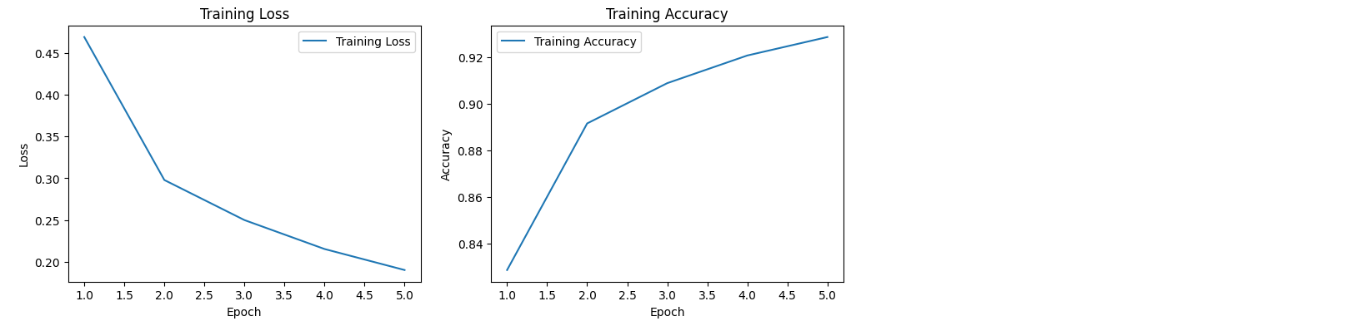
**Training and Evaluation:**

* **Epoch 1**: Loss: 0.46926113804266145, Accuracy: 0.8285666666666667
* **Epoch 2**: Loss: 0.2979015450932578, Accuracy: 0.8916166666666666
* **Epoch 3**: Loss: 0.24989227826661392, Accuracy: 0.9089333333333334
* **Epoch 4**: Loss: 0.21521217554315195, Accuracy: 0.9208333333333333
* **Epoch 5**: Loss: 0.19001383063143124, Accuracy: 0.9287666666666666

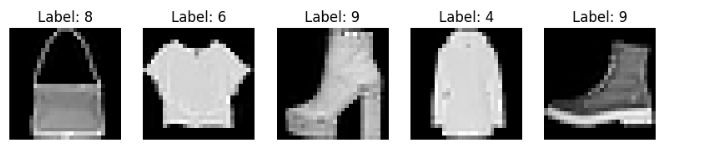
**Test Set Results**:

* **Average Loss**: 0.0002
* **Accuracy**: 91.53%

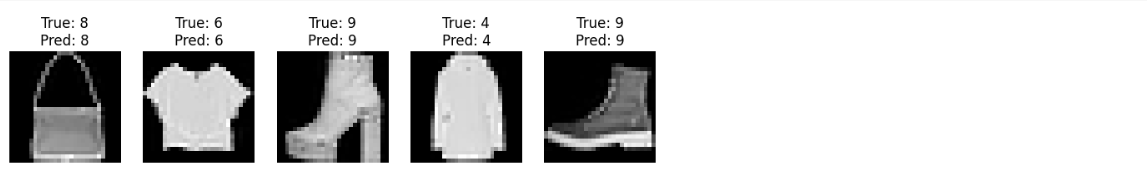
**Accuracy and Loss Plots:**

****

**Sample Images:**

****

**True and Predicted Labels for Sample Images**:



**Conclusion:**

The implemented CNN model achieved a test accuracy of 91.53% on the Fashion MNIST dataset, showcasing its effectiveness in image classification tasks. The model successfully learned hierarchical features through convolutional and pooling layers, leading to robust classification performance. Further optimizations and hyperparameter tuning could potentially enhance the model's accuracy and efficiency in real-world applications.

This report encapsulates the methodology, results, and implications of using a CNN for Fashion MNIST classification, emphasizing its relevance in modern machine learning applications.

he implemented CNN model demonstrated effective classification of the Fashion MNIST dataset with a test accuracy of 91.53%. The results validate the CNN's ability to capture spatial hierarchies in image data. Further optimization and hyperparameter tuning could enhance the model's performance.

**Github Repository Link :**[**https://github.com/itz-aniket-akm/Deep\_learning\_ai**](https://github.com/itz-aniket-akm/Deep_learning_ai)