**Title: Convolutional Neural Networks (CNNs) for Fashion MNIST Classification**

**Roll Number: 23**

**Name: Aniket Rajesh MishraAbstract:** 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:**Convolutional Neural Networks (CNNs) are a class of deep learning models commonly used for image recognition tasks. The Fashion MNIST dataset, comprising grayscale images of fashion items, serves as an excellent benchmark for evaluating CNN performance. This project aims to build, train, and evaluate a CNN model on the Fashion MNIST dataset, with detailed insights into the methodology and results.**Methods:**

**Dataset:**

* **Type**: Fashion MNIST
* **Size**: 70,000 images (60,000 training and 10,000 test images)
* **Image Dimensions**: 28x28 pixels
* **Classes**: 10 (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot)

**Data Preparation:**

* **Train-Test Split**: 60,000 training images and 10,000 test images.
* **Transformations**: Convert images to PyTorch tensors and normalize them to have a mean of 0.5 and standard deviation of 0.5.

**Preprocessing Steps:**

1. Download the dataset.
2. Normalize the images: Pixel values normalized to ensure uniformity in data distribution.
3. Create data loaders for batch processing

**Model:**

* **Architecture**: Convolutional Neural Network (CNN)
* **Layers**:
  + Two convolutional layers
  + Max-pooling layers
  + Three fully connected layers

**Model Architecture:**

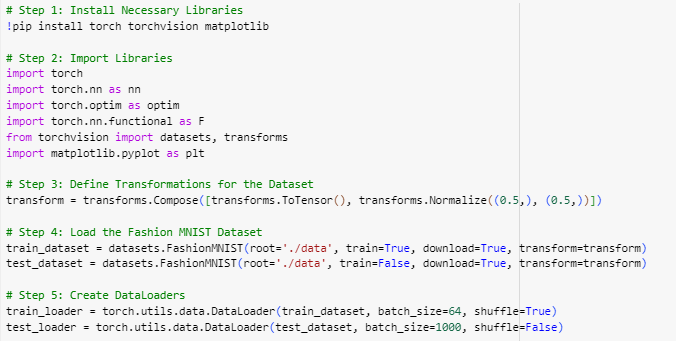
1. **Conv Layer 1**: 1 input channel, 32 output channels, 3x3 kernel
2. **Conv Layer 2**: 32 input channels, 64 output channels, 3x3 kernel
3. **Max Pooling**: 2x2 kernel
4. **Fully Connected Layers**:
   * Layer 1: 64 \* 7 \* 7 input features, 128 output features
   * Layer 2: 128 input features, 64 output features
   * Layer 3: 64 input features, 10 output features (one for each class)

**Optimizer and Loss Function:**

**Optimizer**: Adam,**Learning Rate**: 0.001,**Loss Function**: Cross-Entropy Loss

**Training Parameters:**

* **Epochs**: 5
* **Batch Size**: 64 for training, 1000 for testing

**Code Section from my notebook**: 

**Results:**

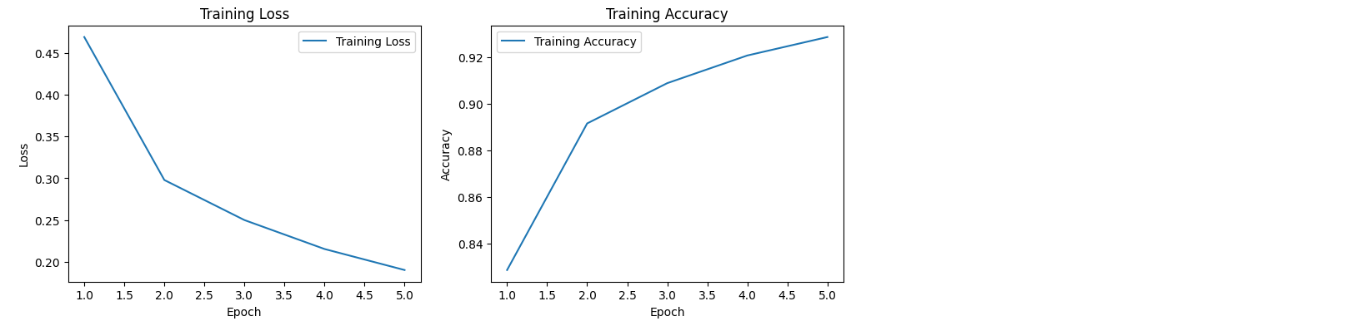
**Training and Evaluation:**

* **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

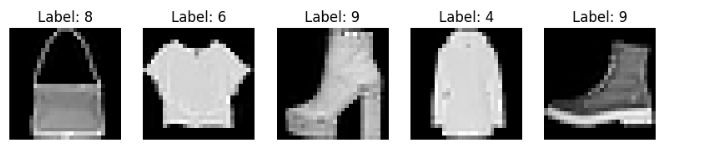
**Test Set Results**:

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

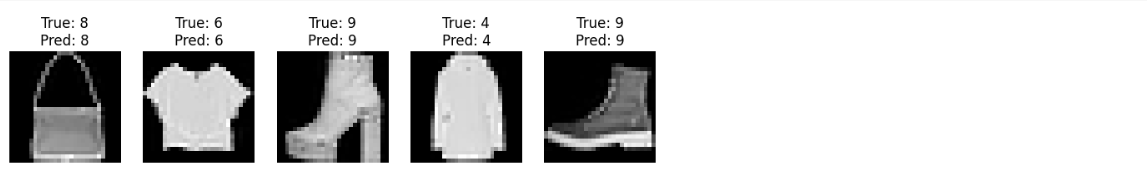
**Accuracy and Loss Plots:**

****

**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.