PyTorch Models: Fashion MNIST Report

Prashant Kumar Singh

5 January 2025

Introduction

This report documents the implementation and experiments of neural network models in PyTorch using the Fashion MNIST dataset. The objective is to build and train a simple fully connected network, understand data preprocessing, and evaluate model performance on a complex image classification task.

Fashion MNIST consists of 70,000 grayscale images of size 28x28 representing 10 fashion categories. Unlike MNIST digits, this dataset is more challenging, with simple linear models achieving around 83% accuracy.

Setup and Requirements

For optimal performance, run this notebook on Google Colab with GPU enabled:

Steps to enable GPU:

1. Runtime → Change runtime type
2. Hardware accelerator → GPU

Required packages:

```
!pip install numpy torch matplotlib scikit-learn torchsummary
```

Data Loading and Preprocessing

Fashion MNIST data is loaded from GitHub repository. The dataset is split into training, validation, and test sets.

Normalization and Splitting

- Normalize grayscale pixel values to [0,1].
- Perform stratified train-validation split using `sklearn.model_selection.train_test_split`.

PyTorch Dataset API

To handle large datasets efficiently, we subclass `torch.utils.data.Dataset`. This allows dynamic loading of images from disk to minimize memory usage.

Listing 1: Custom FashionMNIST Dataset Class

```
from torch.utils.data import Dataset
class FashionMNIST(Dataset):
    def __init__(self, indices, labels):
        self.indices = indices
        self.labels = labels

    def __len__(self):
        return len(self.indices)

    def __getitem__(self, idx):
        sample_path = f'data/unzipped/img_{self.indices[idx]}.npy'
        sample = np.load(sample_path).astype(np.float32) / 255.0
        label = self.labels[idx]
        return sample, label
```

Model Architecture

A simple fully connected network is implemented as a subclass of `torch.nn.Module`:

Listing 2: Custom PyTorch Network

```
import torch.nn as nn
import torch.nn.functional as F

class CustomNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = nn.Linear(784, 100)
        self.layer2 = nn.Linear(100, 10)

    def forward(self, x):
        x = F.relu(self.layer1(x))
        x = self.layer2(x)
        return x
```

Model Summary

Using `torchsummary`:

- Layer1: Linear($784 \rightarrow 100$)
- Layer2: Linear($100 \rightarrow 10$)
- Total parameters: 79,510

Training and Evaluation

- Batch size: 32
- Loss function: Cross-entropy
- Optimizer: Adam or SGD
- Evaluate accuracy on validation and test sets.

Data is loaded in batches using `torch.utils.data.DataLoader`:

```
from torch.utils.data import DataLoader
loader_train = DataLoader(data_train, batch_size=32, shuffle=True)
loader_valid = DataLoader(data_valid, batch_size=32, shuffle=False)
loader_test  = DataLoader(data_test, batch_size=32, shuffle=False)
```

Conclusion

This lab demonstrates:

1. Loading and preprocessing Fashion MNIST using NumPy and PyTorch.
2. Efficient dataset handling with `Dataset` and `DataLoader`.
3. Creating and training a fully connected neural network.
4. Evaluating model performance on a challenging image classification task.

The structure can be extended to more complex models like CNNs or RNNs using the same workflow.

References

- Fashion MNIST dataset: https://github.com/zalandoresearch/fashion-mnist
- PyTorch Documentation: https://pytorch.org/docs/stable/index.html