# written4

November 11, 2024

## 1 Building CNN for Fashion-MNIST data

### 1.1 STP598 Machine Learning & Deep Learning

### 1.2 Written Assignment 4

### 1.3 Due 11:59pm Friday 11/22/2024

In the class, we have learned CNN for image classification. In this short exerciese, you are asked to build a CNN for the similar classification task using PyTorch.

### 1.4 Set up PyTorch

Import PyTorch into your program to get started:

[ ]:
```
%matplotlib inline
import torch
print("PyTorch version:", torch.__version__)

import numpy as np
import matplotlib.pyplot as plt
```

### 1.5 Load a dataset

Load and prepare the Fashion MNIST dataset.

[ ]:
```
import torchvision
import torchvision.transforms as transforms

transform = transforms.Compose(
    [transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))])

# Create datasets for training & validation, download if necessary
training_set = torchvision.datasets.FashionMNIST('./data', train=True,␣
 ↪transform=transform, download=True)
validation_set = torchvision.datasets.FashionMNIST('./data', train=False,␣
 ↪transform=transform, download=True)

# Create data loaders for our datasets; shuffle for training, not for validation
```

```
batch_size = 16
training_loader = torch.utils.data.DataLoader(training_set,␣
 ↪batch_size=batch_size, shuffle=True)
validation_loader = torch.utils.data.DataLoader(validation_set,␣
 ↪batch_size=batch_size, shuffle=False)

# Class labels
classes = ('T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
        'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle Boot')

# Report split sizes
print('Training set has {} instances'.format(len(training_set)))
print('Validation set has {} instances'.format(len(validation_set)))
```

Let's take a peek at these images.

[ ]:
```
dataiter = iter(training_loader)
images, labels = next(dataiter)

plt.figure(figsize=(10,10))
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(images[i].squeeze(), cmap="Greys")
    plt.xlabel(classes[labels[i]])
plt.show()
```

## 2 Build a CNN model

1. First, build a 2-layer CNN `model` by stacking `torch.nn.Conv2D` and `torch.nn.MaxPool2D` layers, followed by a `torch.nn.Flatten` and `torch.nn.Linear` with output dimension being 10 (classes). Define the model as a class named `CNNClassifier` inherited from `torch.nn.Module` (note you need to write your own `forward` method).

[ ]:
```

```

2. We use `torch.nn.CrossEntropyLoss` for the loss function, and `adam` for the optimizer.

[ ]:
```

```

### 2.1 Train and evaluate your model

3. In PyTorch, you need to write your own training/validiation loop. Write **training and validation steps** as sepearate functions (each go over trainig/testing dataset once). Then write the **outer loop** with `n_epochs=100` epochs and call the training/validation steps in

each epoch. **Remember**: in the training step you need to set `model.train()` and in the validation step you need to set `model.eval()`.

[ ]:

4. Record training/testing losses and accuracy. Plot two figures: one for **training/testing losses**, the other for **training/testing accuracy**.

[ ]: