Classification with CNNs: Practice Lab

Below you are provided with the code to load the fashion MNIST dataset (https://github.com/zalandoresearch/fashion-mnist).

```
import torchvision
import torchvision.transforms as transforms
import torch
import torch.nn as nn
import matplotlib.pyplot as plt
import numpy as np
import random

train_set = torchvision.datasets.FashionMNIST(root = ".", train=True,
download=True, transform=transforms.ToTensor())
test_set = torchvision.datasets.FashionMNIST(root = ".", train=False,
download=True, transform=transforms.ToTensor())

train_loader = torch.utils.data.DataLoader(train_set, batch_size=32,
shuffle=False)
test_loader = torch.utils.data.DataLoader(test_set, batch_size=32,
shuffle=False)

# fix the seed to be able to get the same randomness across runs and
hence reproducible outcomes
torch.manual_seed(0)
random.seed(0)
np.random.seed(0)
# if you are using CuDNN , otherwise you can just ignore
torch.cuda.manual_seed(0)
torch.backends.cudnn.deterministic=True
torch.backends.cudnn.benchmark=False
```

Task 1: Create and train a Convolutional Neural Network corresponding to the following architecture:

1. **Input image size:** 28 x 28 x 1 (height x width x number of channels).
2. **First convolutional layer:** Kernel size (3 x 3), Stride size (1 x 1) and 12 output channels. Activation function. **Output dimension:** (28-3+1)/1 x (28-3+1)/1 x 12 = 26 x 26 x 12
3. **Max pooling layer:** Kernel size (2 x 2) and Stride size (2 x 2). **Output dimension:** (26-2+2)/2 x (26-2+2)/2 x 12 = 13 x 13 x 12
4. **Second convolutional layer:** Kernel size (3 x 3), Stride size (1 X 1) and 26 output channels. Activation function. **Output dimension:** (13-3+1)/1 x (13-3+1)/1 x 26 = 11 x 11 x 26
5. **Max pooling layer:** Kernel size (2 X 2) and Stride size (2 x 2). **Output dimension:** (11-2+2)/2 x (11-2+2)/2 x 26 = 5 x 5 x 26

6. **First fully-connected layer** with input size being the output size of max pooling layer in 5. (flattened, i.e. 650) and output size 650. Activation function.

7. **Second fully-connected layer** with input size being the output size of fully connected layer in 6. (i.e. 650) and output size 256. Activation function.

8. **Output layer** with input size being the output size of fully-connected layer in 7. (i.e. 256) and output size 10.

Task 2: For training, initialise your weights using the **Xavier Normal** initialisation, use **ELU** as the activation function, a learning rate of 0.1 with the SGD optimiser. You will train your neural network over 30 epochs.

Task 3: Use 3 different learning rates: 0.001, 0.1, 1.

Task 4: Add a dropout of 0.5 rate on the first and second fully connected layer using the learning rate 0.1.