ECS659U/ ECS659P - NEURAL NETWORKS AND DEEP LEARNING

Coursework Report

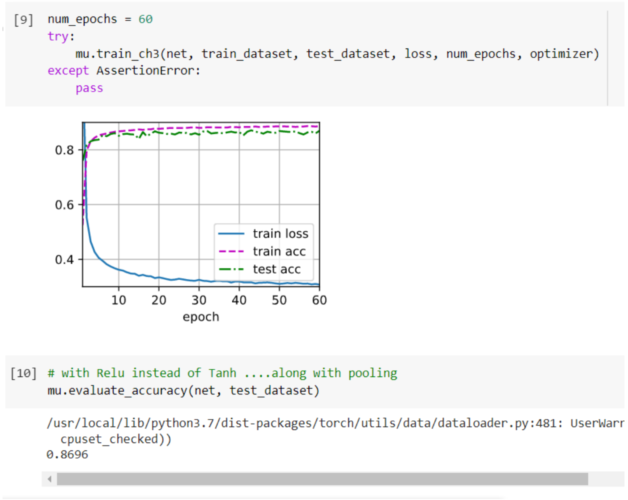
Student Number: 180510010

Details of the Final and Most Optimal Solution Found:

Parameter/ technique used	Value / Type used
Patch size	8
Pooling	used
Loss algorithm	CrossEntropy Loss
Optimizer algorithm	Adam
Learning Rate	0.01
Weight Decay	0.0005
Linear layer in stem	`(8, 126) # (8, num_hidden)`
Linear layer 1	`(98, 56) # (num_inputs, 56)`
Linear layer 2	`(56, 8)`
Linear layer 3	`(124, 89)`
Linear layer 4	`(89, 10) # (89, num_outputs)`
Batch Size	256
Number of epochs	30
Activation function	ReLU

Curves for loss, train accuracy and test accuracy:

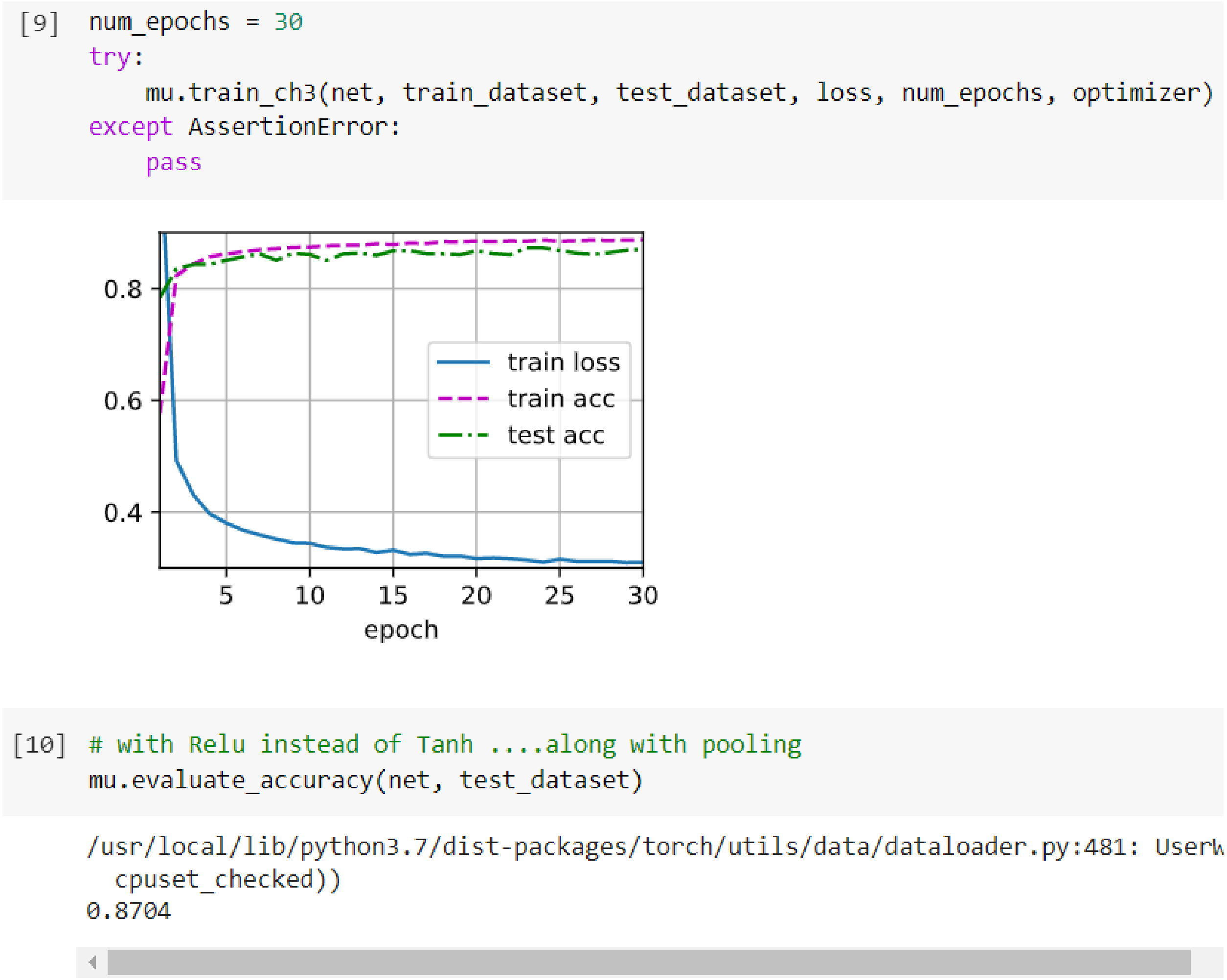
For 60 epochs:



```
[9]  num_epochs = 60
     try:
         mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimizer)
     except AssertionError:
         pass
```

```
[10] # with Relu instead of Tanh ....along with pooling
     mu.evaluate_accuracy(net, test_dataset)

     /usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarr
       cpuset_checked))
     0.8696
```

For 30 epochs:

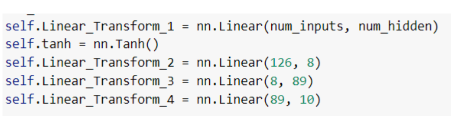


```
[9]  num_epochs = 30
     try:
         mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimizer)
     except AssertionError:
         pass
```

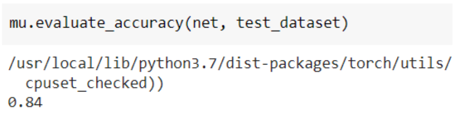
```
[10] # with Relu instead of Tanh ....along with pooling
     mu.evaluate_accuracy(net, test_dataset)

     /usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserW
       cpuset_checked))
     0.8704
```

Different ways/ methods tried:

1. No linear layer for stem and no usage of Max Pooling

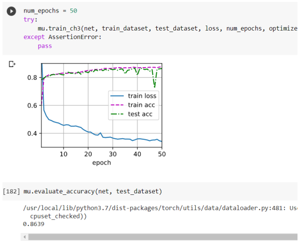


```
self.Linear_Transform_1 = nn.Linear(num_inputs, num_hidden)
self.tanh = nn.Tanh()
self.Linear_Transform_2 = nn.Linear(126, 8)
self.Linear_Transform_3 = nn.Linear(8, 89)
self.Linear_Transform_4 = nn.Linear(89, 10)
```



```
mu.evaluate_accuracy(net, test_dataset)

/usr/local/lib/python3.7/dist-packages/torch/utils/
  cpuset_checked))
0.84
```

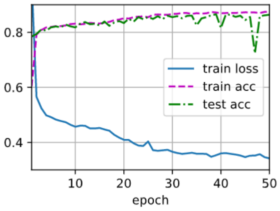
2. Add a linear layer in stem and change dimensions of other layers, also add max pooling



```
num_epochs = 50
try:
    mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimize
except AssertionError:
    pass
```

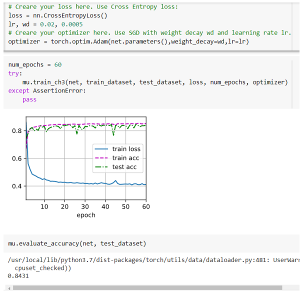
```
[182] mu.evaluate_accuracy(net, test_dataset)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: Us
  cpuset_checked))
0.8639
```



more than 50 epochs…..ran too slow

3. Change the learning rate to 0.02 with no pooling….



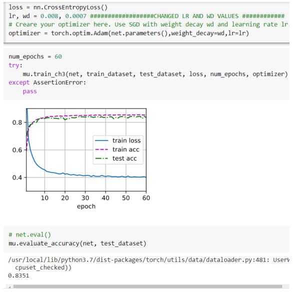
```
# Creare your loss here. Use Cross Entropy loss:
loss = nn.CrossEntropyLoss()
lr, wd = 0.02, 0.0005
# Creare your optimizer here. Use SGD with weight decay wd and learning rate lr.
optimizer = torch.optim.Adam(net.parameters(),weight_decay=wd,lr=lr)
```

```
num_epochs = 60
try:
    mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimizer)
except AssertionError:
    pass
```

```
mu.evaluate_accuracy(net, test_dataset)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWar
  cpuset_checked))
0.8431
```

4. Change learning rate and weight decay to 0.008 and 0.0007 respectively with pooling



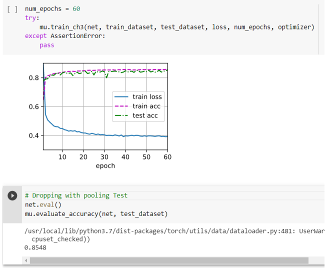
```
loss = nn.CrossEntropyLoss()
lr, wd = 0.008, 0.0007 ##################CHANGED LR AND WD VALUES ############
# Creare your optimizer here. Use SGD with weight decay wd and learning rate lr
optimizer = torch.optim.Adam(net.parameters(),weight_decay=wd,lr=lr)
```

```
num_epochs = 60
try:
    mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimizer)
except AssertionError:
    pass
```

```
# net.eval()
mu.evaluate_accuracy(net, test_dataset)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserW
  cpuset_checked))
0.8351
```

5. Dropping with pooling and the rest values as the same from experiment number 2



```
[ ] num_epochs = 60
    try:
        mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimizer)
    except AssertionError:
        pass
```

```
# Dropping with pooling Test
net.eval()
mu.evaluate_accuracy(net, test_dataset)
```

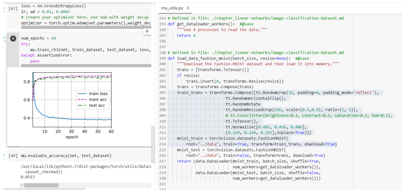
```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWar
  cpuset_checked))
0.8548
```

6. Tests with Data Augmentation:

In my_utils file, line 199, in the function “`def load_data_fashion_mnist(batch_size, resize=None):`” replace line 204 to 206 with these lines below..

```
  trans = transforms.Compose(trans)
   train_trans = transforms.Compose([transforms.RandomCrop(28, padding=4, padding_mode='reflect'),
                  transforms.RandomHorizontalFlip(),
                  #transforms.RandomRotate(),
                  transforms.RandomResizedCrop(256, scale=(0.5,0.9), ratio=(1, 1)),
                  # tt.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1),
                  transforms.ToTensor(),
                  # transforms.Normalize([0.485, 0.456, 0.406],
                  # [0.229, 0.224, 0.225],inplace=True)
                  ])
   mnist_train = torchvision.datasets.FashionMNIST(
       root="../data", train=True, transform=train_trans, download=True)
```

6.1 The test result when I tried running data augmentation without a linear layer after stem…..



```
loss = nn.CrossEntropyLoss()
lr, wd = 0.01, 0.0005
# Creare your optimizer here. Use SGD with weight decay 
optimizer = torch.optim.Adam(net.parameters(),weight_dec
```

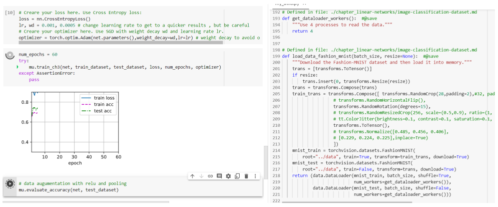
```
num_epochs = 60
try:
    mu.train_ch3(net, train_dataset, test_dataset, loss,
except AssertionError:
    pass
```

```
mu.evaluate_accuracy(net, test_dataset)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/
  cpuset_checked))
0.8513
```

```
# Defined in file: ./chapter_linear-networks/image-classification-dataset.md
def get_dataloader_workers():  #@save
    """Use 4 processes to read the data."""
    return 4


# Defined in file: ./chapter_linear-networks/image-classification-dataset.md
def load_data_fashion_mnist(batch_size, resize=None):  #@save
    """Download the Fashion-MNIST dataset and then load it into memory."""
    trans = [transforms.ToTensor()]
    if resize:
        trans.insert(0, transforms.Resize(resize))
    trans = transforms.Compose(trans)
    train_trans = transforms.Compose([tt.RandomCrop(32, padding=4, padding_mode='reflect'),
                            tt.RandomHorizontalFlip(),
                            tt.RandomRotate
                            tt.RandomResizedCrop(256, scale=(0.5,0.9), ratio=(1, 1)),
                            # tt.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1),
                            tt.ToTensor(),
                            tt.Normalize([0.485, 0.456, 0.406],
                            [0.229, 0.224, 0.225],inplace=True)])
    mnist_train = torchvision.datasets.FashionMNIST(
        root="../data", train=True, transform=train_trans, download=True)
    mnist_test = torchvision.datasets.FashionMNIST(
        root="../data", train=False, transform=trans, download=True)
    return (data.DataLoader(mnist_train, batch_size, shuffle=True,
                            num_workers=get_dataloader_workers()),
            data.DataLoader(mnist_test, batch_size, shuffle=False,
                            num_workers=get_dataloader_workers()))
```

6.2 Data augmentation did not result in good accuracy, it couldn't beat the accuracy from experiment 2 and took way too long to run, it always ran for more than 5 hours.



```
# Creare your loss here. Use Cross Entropy loss:
loss = nn.CrossEntropyLoss()
lr, wd = 0.001, 0.0005 # change learning rate to get to a quicker results , but be careful
# Creare your optimizer here. Use SGD with weight decay wd and learning rate lr.
optimizer = torch.optim.Adam(net.parameters(),weight_decay=wd,lr=lr) # weight decay to avoid o
```

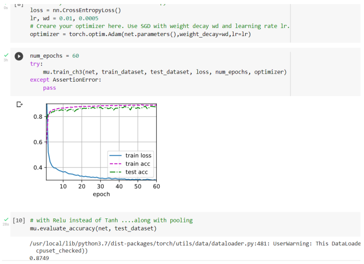
```
num_epochs = 60
try:
    mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimizer)
except AssertionError:
    pass
```

```
# data augmentation with relu and pooling
mu.evaluate_accuracy(net, test_dataset)
```

```
# Defined in file: ./chapter_linear-networks/image-classification-dataset.md
def get_dataloader_workers():  #@save
    """Use 4 processes to read the data."""
    return 4


# Defined in file: ./chapter_linear-networks/image-classification-dataset.md
def load_data_fashion_mnist(batch_size, resize=None):  #@save
    """Download the Fashion-MNIST dataset and then load it into memory."""
    trans = [transforms.ToTensor()]
    if resize:
        trans.insert(0, transforms.Resize(resize))
    trans = transforms.Compose(trans)
    train_trans = transforms.Compose([ transforms.RandomCrop(28,padding=2),#32, pad
                            # transforms.RandomHorizontalFlip(),
                            transforms.RandomRotation(degrees=15),
                            # transforms.RandomResizedCrop(256, scale=(0.5,0.9), ratio=(1,
                            # tt.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1,
                            transforms.ToTensor(),
                            # transforms.Normalize([0.485, 0.456, 0.406],
                            # [0.229, 0.224, 0.225],inplace=True)
                            ])
    mnist_train = torchvision.datasets.FashionMNIST(
        root="../data", train=True, transform=train_trans, download=True)
    mnist_test = torchvision.datasets.FashionMNIST(
        root="../data", train=False, transform=trans, download=True)
    return (data.DataLoader(mnist_train, batch_size, shuffle=True,
                            num_workers=get_dataloader_workers()),
            data.DataLoader(mnist_test, batch_size, shuffle=False,
                            num_workers=get_dataloader_workers()))
```

Tried it a couple times with different lr and wd and different values in linear layers and different activation functions but it still was less accurate then the one without data augmentation.

7. Tried activation function as ReLU instead of Tanh; this showed an increase in the accuracy when compared to the best till now from experiment 2.



```
loss = nn.CrossEntropyLoss()
lr, wd = 0.01, 0.0005
# Creare your optimizer here. Use SGD with weight decay wd and learning rate lr.
optimizer = torch.optim.Adam(net.parameters(),weight_decay=wd,lr=lr)
```

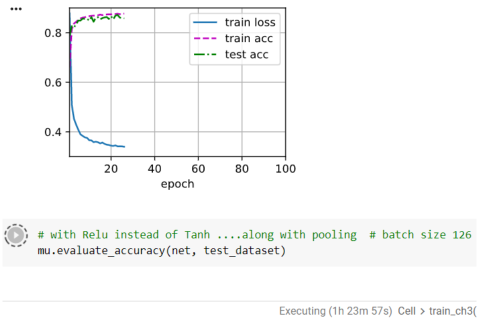
```
num_epochs = 60
try:
    mu.train_ch3(net, train_dataset, test_dataset, loss, num_epochs, optimizer)
except AssertionError:
    pass
```

```
# with Relu instead of Tanh ....along with pooling
mu.evaluate_accuracy(net, test_dataset)
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoade
  cpuset_checked))
0.8749
```

8. Also tired various optimisation algorithms like SGD, Adadelta and others from https://pytorch.org/docs/stable/optim.html#algorithms but Adam showed best results.

9. testing the best accuracy with a different batch size, took too long to run and not as efficient as the one above.



```
# with Relu instead of Tanh ....along with pooling  # batch size 126
mu.evaluate_accuracy(net, test_dataset)
```

References Used

https://towardsdatascience.com/build-a-fashion-mnist-cnn-pytorch-style-efb297e22582
https://www.marktechpost.com/2019/07/30/introduction-to-image-classification-using-pytorch-to-classify-fashionmnist-dataset/
https://jovian.ai/shrivastavatanuj5/fashion-mnist/v/2#C11
Increase dataset size using Data Augmentation - PyTorch Forums