In [71]:
```
import torch
import torchvision
import torch.nn as nn
import torch.optim as optim
```

In [72]:
```
num_inputs, num_outputs, num_hiddens = 784, 10, 256
batch_size_train = 256
n_epochs = 10
```

In [73]:
```
train_loader = torch.utils.data.DataLoader(
    torchvision.datasets.MNIST('./data/', train=True, download=True,
                               transform=torchvision.transforms.ToTensor
    batch_size=batch_size_train, shuffle=True)
```

In [74]:
```
test_loader = torch.utils.data.DataLoader(
    torchvision.datasets.MNIST('./data/', train=False, download=True,
                               transform=torchvision.transforms.ToTensor
test_size = 10000  # may be derived from test_loader
```

In [75]:
```
model2 = nn.Sequential(
    nn.Linear(num_inputs, num_hiddens),
    nn.ReLU(),
    nn.Linear(num_hiddens, num_outputs))
```

In [76]:
```
optimizer = optim.SGD(model2.parameters(), 1e-2)
loss_fn = nn.CrossEntropyLoss()
```

In [79]:
```
for epoch in range(n_epochs):
    for batch_idx, (data, target) in enumerate(train_loader):
        data = data.reshape(-1, num_inputs)
        p = model2(data)
        train_loss = loss_fn(p, target)
        if batch_idx % 100 == 0:
            print('train', epoch, batch_idx, float(train_loss))
        optimizer.zero_grad()
        train_loss.backward()
        optimizer.step()
    m = 0
    for batch_idx, (data, target) in enumerate(test_loader):
        data = data.reshape(-1, num_inputs)
        if int(torch.argmax(model2(data))) == int(target[0]):
            m = m + 1
    print("test", epoch, m, "among", test_size, "correctly classified")
```

```
train 0 0 0.4582749605178833
train 0 100 0.5171159505844116
```

```
train 0 200 0.3756759464740753
test 0 8946 among 10000 correctly classified
train 1 0 0.3633015751838684
train 1 100 0.36128607392311096
train 1 200 0.40276893973350525
test 1 8975 among 10000 correctly classified
train 2 0 0.4527241885662079
train 2 100 0.30566829442977905
train 2 200 0.3201013505458832
test 2 9000 among 10000 correctly classified
train 3 0 0.42652836441993713
train 3 100 0.35005390644073486
train 3 200 0.37130802869796753
test 3 9019 among 10000 correctly classified
train 4 0 0.3349844515323639
train 4 100 0.3257303833961487

train 4 200 0.3827919363975525
test 4 9024 among 10000 correctly classified
train 5 0 0.4580317735671997
train 5 100 0.3367235064506531
train 5 200 0.3692334294319153
test 5 9047 among 10000 correctly classified
train 6 0 0.3415865898132324
train 6 100 0.3283933103084564
train 6 200 0.3658069372177124
test 6 9068 among 10000 correctly classified
train 7 0 0.25363895297050476
train 7 100 0.4067297577857971
train 7 200 0.38327425718307495
test 7 9080 among 10000 correctly classified
train 8 0 0.37642592191696167
train 8 100 0.229636549949646
train 8 200 0.31582850217819214
test 8 9088 among 10000 correctly classified
train 9 0 0.3990965783596039
train 9 100 0.2945866882801056
train 9 200 0.33675438165664673
test 9 9113 among 10000 correctly classified
```

Repeating the process for FashionMNIST

In [80]:
```
F_train_loader = torch.utils.data.DataLoader(
    torchvision.datasets.FashionMNIST('./Fashion/data/', train=True, dow
                                      transform=torchvision.transforms.ToTensor
    batch_size=batch_size_train, shuffle=True)
```

In [81]:
```
F_test_loader = torch.utils.data.DataLoader(
    torchvision.datasets.FashionMNIST('./Fashion/data/', train=False, do
                                      transform=torchvision.transforms.ToTensor
test_size = 10000
```

In [82]:
```
model_F = nn.Sequential(
    nn.Linear(num_inputs, num_hiddens),
    nn.ReLU(),
    nn.Linear(num_hiddens, num_outputs))
```

In [83]:
```
optimizer = optim.SGD(model2.parameters(), 1e-2)
loss_fn = nn.CrossEntropyLoss()
```

In [84]:
```
for epoch in range(n_epochs):
    for batch_idx, (data, target) in enumerate(F_train_loader):
        data = data.reshape(-1, num_inputs)
        p = model2(data)
        train_loss = loss_fn(p, target)
        if batch_idx % 100 == 0:
            print('train', epoch, batch_idx, float(train_loss))
        optimizer.zero_grad()
        train_loss.backward()
        optimizer.step()
    m = 0
    for batch_idx, (data, target) in enumerate(F_test_loader):
        data = data.reshape(-1, num_inputs)
        if int(torch.argmax(model2(data))) == int(target[0]):
            m = m + 1
    print("test", epoch, m, "among", test_size, "correctly classified")
```

```
train 0 0 6.532036304473877
train 0 100 1.4547721147537231
train 0 200 1.1391723155975342
test 0 6570 among 10000 correctly classified
train 1 0 0.9756852984428406
train 1 100 0.8390918374061584
train 1 200 0.7326231598854065
test 1 7185 among 10000 correctly classified
train 2 0 0.7262467741966248
train 2 100 0.8471097350120544
train 2 200 0.6616992354393005
test 2 7536 among 10000 correctly classified
train 3 0 0.6829558610916138
train 3 100 0.7366448044776917
train 3 200 0.6167636513710022
test 3 7711 among 10000 correctly classified
train 4 0 0.6856856942176819
```

```
train 4 100 0.6515212655067444
train 4 200 0.6388264298439026
test 4 7830 among 10000 correctly classified
train 5 0 0.6866046786308289
train 5 100 0.7168999314308167
train 5 200 0.5854805707931519
test 5 7874 among 10000 correctly classified
train 6 0 0.6105462312698364
train 6 100 0.6228758692741394
train 6 200 0.5404723882675171
test 6 7945 among 10000 correctly classified
train 7 0 0.5818050503730774
train 7 100 0.5861023664474487
train 7 200 0.6396629810333252
test 7 7972 among 10000 correctly classified
train 8 0 0.6484094262123108
train 8 100 0.6154968738555908
train 8 200 0.46430712938308716
test 8 8027 among 10000 correctly classified
train 9 0 0.5149075984954834
train 9 100 0.6584874391555786
train 9 200 0.5263075828552246
test 9 8060 among 10000 correctly classified
```

There are 10 classes in both MNIST and FashionMNIST data(Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total.values ranging from 0-255)
There are 70,000 examples almost in which 10,000 are used for testing and remaining for training.
#of epochs are 10.
For every epoch, we see that number of correctly classified examples get better.
Our ReLU model performs better on MNIST(90% accuracy) with 9113 correctly classified among 10000 while 8060 for FashionMNIST(80% accuracy for 10 epochs).These results can get better if number of epochs are increased.