In this lab we will review some basics of pytorch. Save your answers for this lab as they will be used for part of Lab 2.

(1) Create a dataloader for the MNIST training data using torchvision package. Have your dataloader iterate over the training set outputing mini-batches of size 256 image samples. Note you do not need to use the image labels in this lab. You may follow the example in the official pytorch examples:

https://github.com/pytorch/examples/blob/master/mnist/main.py#L112-L120 (https://github.com/pytorch/examples/blob/master/mnist/main.py#L112-L120)

In [195]:

```
import numpy as np
import math
import matplotlib.pyplot as plt
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
```

In [196]:

```
def imshow(img):
    img = img / 2 + 0.5     # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
```

In [197]:

```
batch_size_train = 256
batch_size_test = 256
```

In [201]:

```
train_loader = torch.utils.data.DataLoader(
  torchvision.datasets.MNIST('/files/', train=True, download=True,
                             transform=torchvision.transforms.Compose([
                               torchvision.transforms.ToTensor(),
                               torchvision.transforms.Normalize(
                                 (0.1307,), (0.3081,))
                             ])),
  batch_size=batch_size_train, shuffle=True)

test_loader = torch.utils.data.DataLoader(
  torchvision.datasets.MNIST('/files/', train=False, download=True,
                             transform=torchvision.transforms.Compose([
                               torchvision.transforms.ToTensor(),
                               torchvision.transforms.Normalize(
                                 (0.1307,), (0.3081,))
                             ])),
  batch_size=batch_size_test, shuffle=True)
```

In [204]:

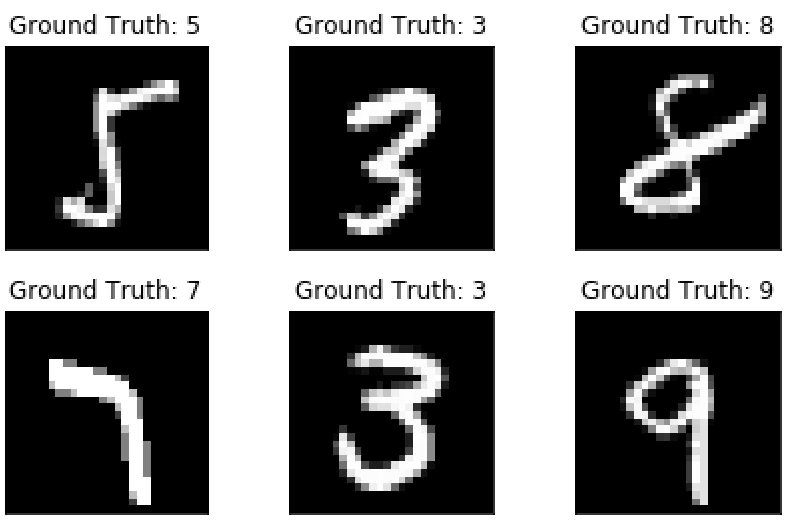
```
examples = enumerate(test_loader)
batch_idx, (example_data, example_targets) = next(examples)
print(example_data.shape)
```

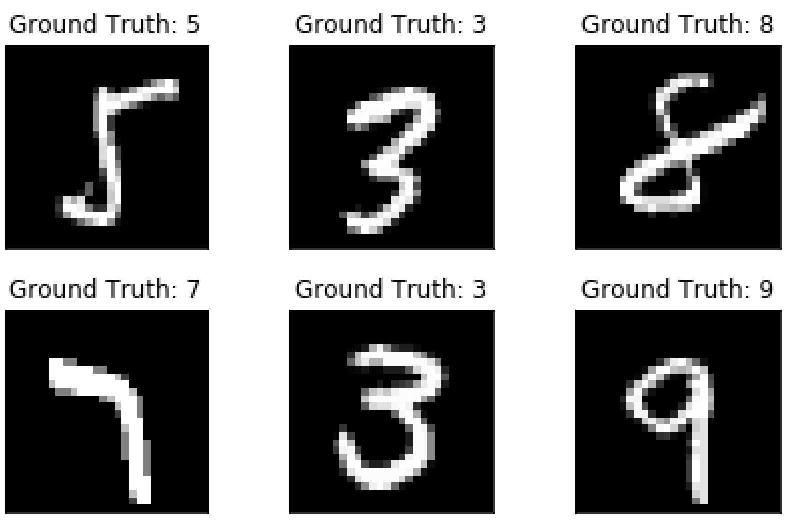
```
torch.Size([256, 1, 28, 28])
```

In [203]:

```
import matplotlib.pyplot as plt
fig = plt.figure()
for i in range(6):
  plt.subplot(2,3,i+1)
  plt.tight_layout()
#   print(len(torch.flatten(example_data[i+10][0])))
  plt.imshow(example_data[i+10][0], cmap='gray', interpolation='none')
  plt.title("Ground Truth: {}".format(example_targets[i+10]))
  plt.xticks([])
  plt.yticks([])
fig
```

Out[203]:





(2) Using only torch primitives (e.g. torch.matmul, torch._relu, etc) implement a simple feedforward neural network with 2 hidden layers that takes as input MNIST digits and outputs a single scalar value. You may select the hidden layer width (greater than 20) and activations (tanh, relu, sigmoid, others) as desired. Initialize the weights and biases with uniform random values in the range -1 to 1. Avoid using any functions from torch.nn class. Using the loop from (1) Forward pass through the dataset in mini-batches of 256 and record the time this takes.

In [205]:

```
# generating some random features
#     print(features)
#     print(features.shape)
# define the weights
# W1 = torch.randn((784, 12), requires_grad=True)
import timeit
params={"W1":torch.FloatTensor(784, 12).uniform_(-1, 1),
    # print(W1)
    "W2": torch.FloatTensor(12, 10).uniform_(-1, 1), #torch.randn((12, 10), requires_gr
ad=True)
    "W3": torch.FloatTensor(10, 1).uniform_(-1, 1), #torch.randn((10, 1), requires_grad
=True)
    # define the bias terms
    "B1": torch.FloatTensor(12).uniform_(-1, 1), #torch.randn((12), requires_grad=True)
    "B2": torch.FloatTensor(10).uniform_(-1, 1), #torch.randn((10), requires_grad=True)
    "B3": torch.FloatTensor(1).uniform_(-1, 1), #torch.randn((1), requires_grad=True)]
       }
```

In [206]:

```
def my_simple_ff_nn(img,nn_params):
    W1=nn_params["W1"]
    W2=nn_params["W2"]
    W3=nn_params["W3"]
    B1=nn_params["B1"]
    B2=nn_params["B2"]
    B3=nn_params["B3"]
    features = torch.flatten(img).reshape(-1)
        # calculate hidden and output layers
    h1 = F.relu((features @ W1) + B1)
    h2 = F.relu((h1 @ W2) + B2)
    output = torch.sigmoid((h2 @ W3) + B3)
    print(output.item())

start = timeit.default_timer()
for idx in range(len(example_data)):
     my_simple_ff_nn(example_data[idx][0],params)
stop = timeit.default_timer()
print('FF 256 images Time= :',stop - start)
```

```
1.4402577797056892e-08
4.507679587600444e-16
0.9999852180480957
1.6640108180254742e-09
7.80124437271823e-11
0.9999995231628418
0.5415602326393127
1.790246528798889e-08
2.4867345516099704e-08
4.730475733349948e-11
0.9545738697052002
3.9292109477173653e-07
6.954677519388497e-05
2.8942963581357617e-06
6.483115004110346e-16
1.8606501701792094e-15
4.46362148631696e-11
5.756077281482419e-10
1.614570237506996e-07
1.1819055411153944e-18
1.0
0.9999982118606567
1.861804026884523e-26
5.697913696423029e-09
7.362937841891293e-13
3.2223538255493622e-06
0.0029047552961856127
0.12742029130458832
0.0029612232465296984
9.802370503839142e-13
4.693649440466741e-13
6.734572899347313e-09
9.653705319578876e-07
0.0008177683921530843
0.0007500430219806731
2.176302336545499e-20
3.252176247769967e-05
6.777034286642447e-06
2.586648975011485e-07
2.2797326092866e-12
1.1692201651314349e-09
0.00046810111962258816
1.3473541002895217e-06
1.3107412163488013e-28
0.12671048939228058
0.6211549639701843
2.900422957363702e-13
7.484991328965407e-07
3.107848669969826e-06
4.431589839494085e-22
4.909878043690696e-06
0.003292708657681942
3.118396207923979e-09
7.0183404658398884e-18
3.1464360197158703e-12
6.9447955866053235e-06
8.238909572355624e-07
0.000362334365490824
1.6401768831997288e-08
1.4709608819885034e-07
5.852489445758863e-14
```

```
0.9909515380859375
0.05372113734483719
0.0013875305885449052
4.620893014577815e-19
2.9937056306161836e-11
1.7022852571990654e-10
8.498809295209853e-13
0.0009132169652730227
0.967231810092926
8.539151052922023e-10
1.0207362732828074e-09
2.1468832755999756e-07
5.281339430764473e-17
3.6520662050065766e-09
0.5735708475112915
4.3310680562791415e-12
0.863641083240509
0.06711430102586746
0.0287513118237257
2.829558987116343e-08
0.06425981968641281
1.1789620835145342e-08
0.1827048361301422
0.060055334120988846
7.660440815016045e-08
6.3346415117848665e-06
2.482576861098651e-13
4.426501476473277e-08
2.253787911854488e-08
0.0004025489615742117
0.9823784232139587
0.3906150758266449
0.7923251390457153
5.527327727461451e-13
1.5371188208986725e-13
4.973527666152222e-06
3.594823647290468e-05
0.9513925313949585
1.828611784224026e-16
0.9999476671218872
3.923189808574534e-07
0.6548846364021301
0.002520273206755519
3.482542048230941e-19
2.25317755564447e-08
0.011190677061676979
9.55264400914757e-09
4.700009138535271e-17
1.52650505083618e-12
1.1725599657896879e-11
0.0015990736428648233
5.720400153563787e-09
1.0146504791919142e-05
0.318092942237854
0.0
1.696946055895765e-10
1.1009478884105306e-09
3.356590179415253e-12
9.89749332802603e-06
0.8055285215377808
2.0014249348702363e-22
```

```
    1.1445393965914263e-07
    3.157634743955473e-13
    1.1494387663901762e-09
    3.8228969496227924e-20
    2.799103299058244e-18
    1.1181242598468089e-07
    9.512720566817734e-08
    9.678724381956272e-07
    2.756066930231782e-09
    1.2579655539468604e-08
    9.691276545709115e-07
    0.013858329504728317
    3.9241735353609175e-25
    0.00031054270220920444
    1.0268488495057682e-06
    2.4293061073876743e-07
    0.0003938642330467701
    0.00026626151520758867
    1.165203229902545e-06
    4.624874418368563e-05
    3.69119419474373e-07
    0.002168803010135889
    4.345211343514954e-11
    5.967056495137513e-05
    1.6563757526455447e-05
    0.0009709612349979579
    1.6611828357326885e-08
    7.4044281817009505e-09
    6.990982365095988e-05
    0.006242815405130386
    1.717399555900556e-08
    0.00012678901839535683
    5.5310454133428166e-11
    2.758705038712073e-18
    1.7356811976453658e-21
    9.236453713382828e-26
    1.765051914169721e-15
    3.646767936515971e-07
    0.00014833608292974532
    2.1778266727778828e-06
    2.2476839944829408e-07
    0.00041066447738558054
    0.0010323785245418549
    0.9912692308425903
    0.9864056706428528
    3.4171491734014126e-06
    4.747738785226829e-06
    6.975338034130243e-19
    0.0804533138871193
    9.947524439368402e-14
    5.247704848443391e-07
    4.44286087031287e-07
    0.04061568155884743
    1.685496317804791e-05
    0.9998099207878113
    0.8102703094482422
    1.472712714303045e-26
    1.0184166967519559e-05
    5.712819661773949e-10
    7.208080660126015e-11
    0.0005006641149520874
```

```
3.1895972085749236e-08
0.9740603566169739
8.382863825318054e-07
0.8695939183235168
0.0018845685990527272
2.27153923918907e-22
1.70133507282344e-09
1.4328931285945146e-07
1.254595076716214e-06
0.003785116132348776
0.0003267493157181889
2.480082343936374e-07
1.9873618058433654e-18
0.0017044750275090337
0.521851122379303
0.15545541048049927
0.9984502792358398
0.9202014803886414
1.881198727460287e-06
2.6015859688754972e-08
6.465308783372284e-16
0.20641110837459564
0.017619024962186813
0.7027040719985962
0.5815284252166748
2.1022249541147175e-07
1.3269408905003388e-09
0.994576096534729
2.9055728933968794e-09
1.510478686572725e-10
2.9721152839182197e-14
0.2732855975627899
0.05491092801094055
6.596179330877576e-21
6.830154219983342e-10
7.073225657279444e-25
2.1823480522709104e-11
5.089693111415272e-09
3.378403926035389e-05
0.043236907571554184
1.4748181842438868e-11
5.474609679367859e-06
0.00412067212164402
1.599712007305243e-08
6.069195706004393e-07
1.1085811024713621e-07
0.0026400277856737375
6.666722285331161e-09
4.2242410408009147e-19
0.09600368142127991
0.6510941386222839
6.39379001132756e-12
1.2808877121125734e-14
9.773781357580447e-07
5.715772229434798e-14
0.3497486412525177
0.88978111743927
0.0014468361623585224
0.003839551964640617
5.3782073337060865e-06
2.880649674352753e-07
```

```
0.0001019063129206188
3.830219156952808e-06
0.0011528186732903123
0.006829683668911457
6.086305347707821e-06
2.0656136257457547e-05
0.3427250385284424
0.018638744950294495
1.310100036166714e-15
0.000381559191737324
6.611006408974163e-09
3.0366340070031583e-05
FF 256 images Time= : 0.4360132710025937
```

(3) Implement a new torch.nn.module that performs the equivalent of the network in (2). Initialize it with the same weights and validate the outputs of this network is the same as the one in (2) on MNIST training set.

In [207]:

```python
# define the network class
class my_torch_nn(nn.Module):
    def __init__(self):
        # call constructor from superclass
        super().__init__()
        # define network layers
        self.fc1 = nn.Linear(784, 12)
        self.fc2 = nn.Linear(12, 10)
        self.fc3 = nn.Linear(10, 1)
        self.b1=params["B1"]
        self.b2=params["B2"]
        self.b3=params["B3"]

    def forward(self, x):
        x= torch.flatten(x).reshape(-1)
        x.requires_grad_(True)
        # define forward pass
        x = F.relu(self.fc1(x)+self.b1)
        x = F.relu(self.fc2(x)+self.b2)
        x = torch.sigmoid(self.fc3(x)+self.b3)
#          print(x.item())
        return x
def weights_init_uniform(m):
        classname = m.__class__.__name__
        # for every Linear layer in a model..
        if classname.find('Linear') != -1:
            # apply a uniform distribution to the weights and a bias=0
            m.weight.data.uniform_(-1, 1.0)
            m.bias.data.uniform_(-1, 1.0)
```

In [208]:

```
network = my_torch_nn()
network.apply(weights_init_uniform)
print(network)
```

```
my_torch_nn(
  (fc1): Linear(in_features=784, out_features=12, bias=True)
  (fc2): Linear(in_features=12, out_features=10, bias=True)
  (fc3): Linear(in_features=10, out_features=1, bias=True)
)
```

In [212]:

```
start = timeit.default_timer()
for idx in range(len(example_data)):
    out=network(example_data[idx][0])
    print(out.item())
stop = timeit.default_timer()
print('FF 256 images Time= :',stop - start)
```

```
0.9917540550231934
1.0
3.567948078853078e-05
0.9346117973327637
0.6078295111656189
1.0
0.0036443835124373436
0.02089640311896801
6.486820893769618e-07
0.9999998807907104
1.1377377262533628e-07
7.598759452775994e-07
0.0009539852617308497
0.9999992847442627
1.0
0.006148576270788908
1.0
3.0443759897025302e-05
1.0
0.998969316482544
1.0
4.9460263063585276e-20
1.0
8.881455814569875e-19
1.0
4.6736272756595434e-14
2.4547870403068828e-09
1.0
1.0
0.003484695451334119
0.2297322303056717
0.9760248064994812
1.044636701408308e-05
3.120041031090537e-11
0.12806713581085205
1.0
1.0
0.9999998807907104
0.9410972595214844
6.196972451451266e-18
1.0
1.0
1.2419676398423007e-09
1.0
0.0096505181863904
8.0370855803924e-11
0.9547421336174011
1.0
1.0
1.0
4.318089497079569e-10
1.0
0.43043747544288635
1.0
1.0
1.0
1.0
0.002916357247158885
2.1210651084402343e-06
8.052815246628597e-06
0.0035085310228168964
```

```
0.9999955892562866
5.719215587771977e-15
0.9998886585235596
1.2634658332899562e-06
2.3455434643160356e-13
0.994412362575531
0.9748363494873047
1.202216104445597e-10
0.00025193486362695694
2.272718666063156e-05
5.456720828078687e-06
0.02949371561408043
0.9999439716339111
1.0
0.9999825954437256
0.13604603707790375
4.193636777927168e-05
4.712225290859351e-06
0.9984211921691895
0.10260171443223953
9.448166207737252e-12
0.9859226942062378
0.562605619430542
2.643078681266453e-12
0.001111001125536859
1.0
0.9843794703483582
4.488009880790646e-34
1.0
0.999923825263977
1.3462961214599778e-12
0.007517380174249411
4.947393529578984e-39
0.9993465542793274
1.0
1.0
0.5802053809165955
0.0349331833422184
1.0
1.0
1.0
9.688443525202301e-25
0.9617414474487305
0.9868940114974976
0.010467179119586945
1.0340322348367792e-12
1.0
0.9653767943382263
0.9999877214431763
0.9802634119987488
1.0
1.0070044705870126e-13
1.0
1.650797720742503e-08
1.0
1.0
1.0
0.5113903284072876
6.004683555937973e-11
1.0705129227517318e-08
0.7982217073440552
```

```
1.0
0.9978679418563843
0.9996434450149536
4.5050590813389135e-08
1.0
1.0
1.5469454606176214e-09
1.0
0.9843103885650635
1.0
2.7070980195276206e-06
0.8798226118087769
1.0
0.9998650550842285
0.9998214840888977
0.1861869990825653
0.999998927116394
1.0
0.915547788143158
0.6972649693489075
1.358540771434491e-06
4.98056077381519e-20
0.835419774055481
0.9102473855018616
0.001386109390296042
1.0
1.0
0.9999809265136719
0.26559239625930786
1.452054947635728e-12
0.011598258279263973
0.00010589445446385071
0.9987773299217224
1.0
0.04776821285486221
0.9999827146530151
1.0
0.9999984502792358
0.9999997615814209
0.9999990463256836
0.26709774136543274
0.9999998807907104
0.28492385149002075
2.4478124416305036e-08
0.17424257099628448
1.0
1.0
0.9999908208847046
0.7847537398338318
0.9999912977218628
0.9998518228530884
6.17144962821482e-17
5.870496534043923e-05
0.19193504750728607
0.9170371890068054
0.5297399759292603
1.0
0.11800438910722733
0.9999194145202637
1.6347447213179572e-13
1.0
```

```
0.9987058639526367
0.9947890043258667
8.353630060281166e-22
2.626532602523213e-10
1.054411171708125e-07
1.0
0.9980043768882751
0.7744426727294922
1.8627124495206915e-27
0.9998531341552734
1.0
0.9999998807907104
1.0
0.8907246589660645
0.9992200136184692
5.5845696985379386e-18
2.430991571600316e-06
2.3826264623494353e-06
1.080171411833413e-19
1.0
0.21373014152050018
0.9539979696273804
0.018299464136362076
0.9951139688491821
1.0
1.0
9.342791713606857e-07
1.2211480679980014e-06
1.0
5.13734494254766e-21
1.0
1.0
2.322866663462264e-07
1.0
0.9999213218688965
1.0
0.9995430707931519
0.12970297038555145
0.23948687314987183
1.3640244560519932e-06
0.9990639090538025
5.611632900581753e-07
0.9867427349090576
0.0401960052549839
6.001197760951982e-09
0.9999998807907104
8.80788819564984e-15
0.9999914169311523
0.44502192735671997
0.9013323187828064
1.1657306458801031e-05
1.0
1.0
0.001309512066654861
1.0
3.0767248517804546e-06
4.80148872050723e-11
0.9999924898147583
3.927719660623552e-07
0.00010346751514589414
0.9996016621589661
```

```
1.0
0.07346361875534058
0.19638405740261078
0.9998432397842407
1.6645551256799784e-11
1.2247907577034312e-08
0.853763997554779
5.796571826977015e-07
1.0
0.9776801466941833
1.0
1.0
FF 256 images Time= : 0.6209280330003821
```

(4) For a batch of 256 random samples, compute the gradient of the average of the neural network outputs (over the batch) w.r.t to the weights using torch autograd. Compute the gradients for the torch.nn based model in (3) and validate the gradients match those from those computed with (2).

Note: The network here is $f : \mathcal{R}^{HW} \rightarrow \mathcal{R}$, with 256 samples you should obtain $o = \frac{1}{256}\sum_{i=0}^{255} f(x_i)$. You are asked to find $\nabla_w o$ for all the parameters w.

In [213]:

```python
loss_fn = torch.nn.MSELoss(reduction='sum')
for idx in range(len(example_data)):
#     print(example_targets[idx])
    output = network(example_data[idx][0])
    loss = output-example_targets[idx]
    loss.backward()
#     print(output.grad)
```

In [215]:

```
print('fc1 avg grad=\n',sum(network.fc1.weight.grad)/len(network.fc1.weight.grad))
print('fc1 bais=',network.fc1.bias.grad)
print('fc2 avg grad=\n',sum(network.fc2.weight.grad)/len(network.fc2.weight.grad))
print('fc2 bais=',network.fc2.bias.grad)
```

```
fc1 avg grad=
 tensor([-2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9693e-02, -3.0182e-02, -3.2058e-02, -3.7198e-02, -3.7180e-02,
        -4.8430e-02, -4.1192e-02, -3.0306e-02, -2.9829e-02, -3.1752e-02,
        -3.4242e-02, -3.3649e-02, -3.1320e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -3.0180e-02, -8.7466e-02,
        -1.4056e-01, -1.2409e-01, -7.4385e-02, -8.1645e-02, -1.6084e-01,
        -1.3519e-01, -2.6013e-02, -1.8329e-02, -3.1053e-02, -3.4286e-02,
        -3.2556e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9698e-02, -2.9863e-02,
        -2.9944e-02, -1.3120e-02,  1.8946e-02, -7.1393e-02, -1.5213e-01,
        -1.5307e-02, -2.6898e-02, -4.9969e-02, -5.2298e-02, -2.1142e-03,
         6.8151e-02,  2.1999e-01,  1.2613e-01, -5.2896e-02, -1.6329e-02,
        -2.4213e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.1374e-02,
        -3.0607e-03,  2.8111e-02,  2.7372e-02,  7.4073e-02,  1.0836e-01,
         1.6849e-01, -1.1874e-01, -2.4881e-01, -2.3116e-01, -6.9664e-02,
         4.6965e-03,  1.9177e-01,  3.0338e-01,  4.4411e-01,  4.4474e-01,
         3.3124e-01, -1.8784e-02, -9.2263e-03, -1.1194e-02, -2.9824e-02,
        -2.9856e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02,  1.9777e-02,  2.8366e-02,  1.6101e-02,
        -1.8432e-01, -7.9752e-02,  1.4809e-01,  8.6823e-02, -2.2938e-01,
        -3.2024e-01, -4.1201e-01, -2.7603e-01, -1.5991e-01,  1.3566e-01,
         4.5755e-01,  6.5332e-01,  6.2069e-01,  5.3880e-01,  2.2323e-01,
         1.7593e-01,  8.1333e-02, -2.9328e-02, -2.9733e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9696e-02,
        -1.0721e-03, -6.3412e-03, -5.7287e-02, -5.8071e-02,  7.7558e-02,
        -5.8088e-02, -8.5118e-02, -1.3913e-01, -8.9766e-02, -1.1916e-01,
        -2.7606e-03, -1.1692e-01, -1.5612e-01,  1.8214e-01,  3.9096e-01,
         4.7259e-01,  5.1702e-01,  2.4751e-01,  1.5847e-01,  2.8637e-02,
        -1.9196e-02, -2.1823e-02, -2.7361e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9696e-02, -3.6631e-02, -7.6157e-02,
        -1.2163e-01, -5.4136e-02,  3.9324e-01,  2.0391e-01,  1.8346e-01,
         2.1471e-01,  2.0427e-01,  1.9313e-01,  2.4416e-01, -2.0081e-02,
        -1.9003e-01, -1.5582e-01,  2.0478e-01,  1.8105e-01,  3.7686e-01,
         2.3771e-01,  1.7640e-01,  1.8730e-01, -1.7590e-02, -1.9588e-02,
        -2.7849e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9696e-02, -3.6687e-02, -9.9918e-02, -1.2190e-01,  3.9142e-02,
         5.1088e-01,  4.2403e-01,  4.5001e-01,  5.7520e-01,  6.2558e-01,
         6.8527e-01,  2.8815e-01,  2.6680e-01,  5.7026e-02,  2.9876e-02,
         1.9854e-01,  4.5845e-02,  1.0361e-01, -9.4161e-03,  4.5132e-02,
        -1.9454e-02, -2.1724e-02, -2.6966e-02, -2.9012e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -3.3034e-02,
        -1.1653e-01, -7.4812e-02,  1.1774e-01,  3.6570e-01,  5.8231e-01,
         6.5867e-01,  8.1738e-01,  5.1575e-01,  2.5565e-01,  7.5698e-03,
         8.4670e-02, -6.6139e-02,  1.7401e-01,  1.0274e-01, -1.1159e-01,
        -1.7708e-01, -4.4147e-02, -3.1728e-02, -3.4330e-02, -2.9375e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -3.6162e-02, -1.1892e-01, -9.2876e-02,
         4.2709e-02,  3.0249e-01,  4.6520e-01,  6.5489e-01,  7.0426e-01,
         3.8355e-01,  2.4300e-01,  7.9840e-02,  7.4218e-02,  6.2812e-02,
        -1.5892e-02,  5.5336e-02, -1.5541e-01, -1.8873e-01, -5.9850e-02,
        -5.2364e-02, -6.0661e-02, -2.9739e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -5.0384e-02, -1.0154e-01, -9.7516e-02, -1.0679e-01,  1.7755e-01,
         3.4896e-01,  4.1916e-01,  5.9225e-01,  2.9324e-01,  8.6177e-02,
         1.0605e-01,  2.5693e-01,  2.6943e-01,  5.5918e-02, -1.2957e-01,
        -1.7105e-01, -4.3858e-02, -6.6306e-02, -8.3883e-02, -7.9386e-02,
        -2.9063e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -6.8151e-02, -1.0034e-01,
        -1.3516e-01, -1.4242e-01, -5.9363e-02,  1.7707e-01,  1.3538e-01,
         1.7181e-01,  8.7764e-02, -2.4822e-01, -1.2173e-02,  5.5200e-01,
         3.8379e-01,  2.8829e-01, -2.1656e-01,  2.0914e-02, -6.5640e-02,
        -1.6988e-01, -1.8993e-01, -1.6889e-01, -5.1011e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -6.8341e-02, -1.2216e-01, -1.8814e-01, -1.1088e-01,
        -9.5257e-02, -6.1596e-02, -1.1226e-01,  6.9797e-02,  1.9418e-01,
         2.1562e-01,  2.6568e-01,  3.3794e-01,  3.5120e-01,  2.1503e-01,
        -9.0000e-02,  1.0407e-01, -4.4530e-02, -1.6227e-01, -1.8426e-01,
        -2.2147e-01, -1.0041e-01, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -5.2892e-02,
        -1.1551e-01, -2.1301e-01, -9.5149e-02, -2.8288e-02,  8.4305e-02,
        -8.1234e-02, -1.2081e-01,  4.4354e-02, -4.3769e-02,  1.0717e-01,
         1.5498e-01,  5.7330e-03,  8.0549e-02,  4.8496e-02,  1.1365e-01,
         1.8560e-01,  6.0579e-02, -1.4659e-01, -2.0885e-01, -1.1234e-01,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -3.8804e-02, -6.3362e-02, -1.6528e-01,
        -1.1950e-02, -1.6663e-02, -1.0703e-01, -8.4725e-02, -1.4227e-01,
        -2.8974e-02, -4.1061e-02, -3.1436e-02,  1.1045e-01,  9.6134e-02,
        -8.8881e-02,  3.2895e-02,  1.5821e-01,  1.6889e-01,  8.4747e-02,
        -1.4881e-01, -1.2836e-01, -8.0845e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.5074e-02,  1.2421e-01,  4.7462e-02,  6.9495e-02,  1.0232e-01,
         1.2878e-01, -1.7063e-02,  2.6080e-01,  4.5406e-01,  3.5947e-01,
         2.9577e-01,  2.6167e-01,  3.5824e-01, -8.3887e-02, -2.3961e-01,
         7.6368e-02,  1.6817e-01,  6.8351e-02, -2.3789e-02, -4.1433e-02,
        -7.2991e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02,  1.3683e-01,  1.5222e-01,
         1.2472e-01,  1.2238e-01,  9.6351e-02, -7.9148e-02,  5.4021e-03,
         4.6636e-01,  5.6494e-01,  2.9687e-01,  6.8696e-02,  2.7078e-01,
         3.0104e-01,  1.7831e-01, -1.8903e-01,  3.3056e-02,  2.3431e-01,
         1.4743e-01, -1.7350e-02, -6.2319e-02, -5.2058e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9728e-02,  3.3336e-02,  1.7600e-01,  1.4686e-01,  1.0743e-01,
        -6.0429e-02, -1.7105e-01,  2.0375e-01,  5.7613e-01,  4.6143e-01,
         5.3275e-02, -5.0176e-02,  2.1809e-01,  3.4796e-01,  2.2709e-01,
        -4.9820e-04,  1.4409e-01,  4.1504e-01,  3.0695e-01,  1.5127e-02,
        -7.0894e-02, -4.6476e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -3.3347e-02, -3.9384e-02, -3.4215e-02,
        -2.6305e-02,  1.9585e-02,  9.9019e-02, -2.1089e-01, -1.9889e-01,
         1.7856e-01,  5.2982e-01,  4.0201e-01,  2.0082e-02,  1.5700e-02,
         1.6989e-01,  1.7311e-01,  1.2120e-02,  5.8746e-02,  4.3340e-01,
         4.7851e-01,  4.1024e-01, -1.5911e-02, -3.2046e-02, -3.1757e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -3.9768e-02, -3.9882e-02, -3.9456e-02, -3.2071e-02, -1.9040e-02,
        -1.4318e-01, -1.5846e-01, -2.4214e-01,  2.6305e-02,  3.6285e-01,
         4.3300e-01,  4.4394e-01,  5.5328e-01,  3.3431e-01,  1.0101e-01,
        -7.8036e-03,  3.2430e-01,  6.0649e-01,  5.0382e-01,  2.6226e-01,
         2.9229e-02, -2.9998e-02, -2.9696e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9799e-02, -2.9936e-02,
        -3.0113e-02, -3.1075e-02, -1.7850e-02, -4.2673e-02, -1.5173e-01,
        -2.3869e-01, -1.6514e-01,  1.6376e-01,  4.4153e-01,  4.5632e-01,
         4.6896e-01,  2.7268e-01,  1.2125e-01,  2.0693e-01,  3.6229e-01,
         4.7943e-01,  3.1362e-01,  2.1294e-01,  3.9497e-02, -2.9696e-02,
        -2.4557e-02, -2.4802e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9745e-02, -2.9945e-02,
         6.1056e-03,  8.9809e-02, -7.1136e-02, -1.6706e-01, -9.3492e-02,
        -1.8178e-02,  3.6023e-02,  1.4118e-01,  1.6549e-01,  2.6254e-01,
         3.2445e-01,  1.0466e-01,  9.9152e-03,  3.1511e-01,  2.4205e-01,
         5.2892e-02,  2.4451e-02, -1.6057e-02, -2.4558e-02, -2.4802e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.6047e-02,  7.2500e-02,
         5.4720e-02, -6.2791e-02, -1.7437e-01, -2.0742e-01, -1.9031e-01,
         7.5187e-04,  1.7896e-01,  3.7437e-01,  3.0403e-01, -1.6911e-02,
        -1.1727e-01, -1.7393e-01, -1.6839e-01,  3.0449e-02, -9.1215e-03,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02,  1.0015e-02,  7.9161e-03,
        -1.3918e-02, -2.7363e-02, -7.9079e-02,  4.6091e-02,  1.4662e-01,
         9.6927e-02, -5.5653e-03, -4.6131e-02, -2.4161e-01, -1.5125e-01,
        -3.0730e-02, -2.9285e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9914e-02, -2.9312e-02, -1.0207e-02, -9.5697e-03,
        -3.3224e-02, -4.1456e-02, -4.0739e-02, -3.6262e-02, -3.0969e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.0591e-02, -1.1563e-02, -2.6195e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02,
        -2.9697e-02, -2.9697e-02, -2.9697e-02, -2.9697e-02])
fc1 bais= tensor([ 8.9123, -4.2559,  3.0208,  2.0765, -0.6074,  0.7950, -
2.6577, -0.8769,
         1.1657, -4.4276,  0.9656, -3.2704])
fc2 avg grad=
 tensor([ 0.7341,  2.6997,  0.4672,  1.0483,  0.4293, -0.0643, -0.3271,
0.2700,
         2.6865,  1.1172,  0.9700, -1.4761])
fc2 bais= tensor([ 4.9468,  4.6024,  6.6368,  2.7679, -0.3700, -3.8922, -
0.1486, -0.6637,
        -6.5121, -5.9982])
```

(5) Perform the forward and backward passes from (3), 10 times on cpu and 10 times on gpu, report the average time for both. Repeat this for just the forward pass. In the end you should obtain 4 average run times (forward and backward, forward only) x (cpu, gpu)

In [216]:

```
import timeit
loss_fn = torch.nn.MSELoss(reduction='sum')

total_t=0
for iter in range(10):
    start = timeit.default_timer()
    for idx in range(len(example_data)):
        output = network(example_data[idx][0])
        loss = output-example_targets[idx]
        loss.backward()
    stop = timeit.default_timer()
    total_t=total_t+(stop - start)
print('avg_forward and backward_cpu_time=',total_t/10,'\n')

total_t=0
for iter in range(10):
    start = timeit.default_timer()
    for idx in range(len(example_data)):
        output = network(example_data[idx][0])
    stop = timeit.default_timer()
    total_t=total_t+(stop - start)
print('avg_forward_cpu_time=',total_t/10,'\n')
```

```
avg_forward and backward_cpu_time= 0.4352928799002257

avg_forward_cpu_time= 0.2147864459999255
```

In [ ]: