HW3_F21

November 18, 2021

1 Homework 3

DUE Nov 15th at 11:59 PM

1.1 Problem 1

In this problem, you will implement a simple feed-forward neural network using PyTorch, a straightforward and simple-to-pickup framework for quickly prototyping deep learning model.

PyTorch provides 2 powerful things. First, a nice data structure called Tensor (basically a matrix, similar to Numpy ndarray). Tensor is optimized for matrix calculation and can be loaded to a GPU. Tensor is also implemented so that it's easy to calculate and pass back chains of gradients, which is extremely useful for backpropagation on neural network. Second, a nice inner mechanism called Autograd that nicely maps variables involved a chain of calculations and efficiently calculates their gradients via the chain rule when needed. Read more here: https://towardsdatascience.com/pytorch-autograd-understanding-the-heart-of-pytorchs-magic-2686cd94ec95

You will define a neural network class in PyTorch and use the network to learn a classification task on the famous KDD CUP 99 dataset. You can refer to Problem 2 to see how a network class can be defined, how to use a PyTorch's DataLoader, and how a training loop may looks like.

There are many greate tutorial on PyTorch out there. For example, this video on Youtube explains how to build a simple network in PyTorch quite clearly: https://www.youtube.com/watch?v=oPhxf2fXHkQ

1.1.1 Part a

Firstly, load and inspect the "**KDD CUP 99**" dataset.

[1]:
```
from sklearn.datasets import fetch_kddcup99
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
import torch

X, y = fetch_kddcup99(return_X_y=True, percent10=True)

enc = OrdinalEncoder()
enc.fit(X[:,1:4])
X[:,1:4] = enc.transform(X[:,1:4])

lab = LabelEncoder()
```

```
lab.fit(y)
y = lab.transform(y)
```

Split them into a train set (70%), a validation set (10%), and a test set (20%). Then, create a PyTorch's DataLoader for the train set, a DataLoader for the validation set, and a DataLoader for the test set.

You can read about PyTorch's DataLoader from:

- https://stanford.edu/~shervine/blog/pytorch-how-to-generate-data-parallel
- https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader

[82]:
```
class Dataset(torch.utils.data.Dataset):
    # Characterizes a dataset for PyTorch'
    def __init__(self, X, y):
        #'Initialization'
        self.labels = torch.tensor(y)
        self.features = torch.tensor(X)

    def __len__(self):
        #'Denotes the total number of samples'
        return len(self.labels)

    def __getitem__(self, index):
        #'Generates one sample of data'
        # Select sample
        return self.features[index], self.labels[index]
```

[83]:
```
from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader
import numpy as np

# Define test and train
X_train, X_test, y_train, y_test = train_test_split(X,y,
                                                    test_size=0.2,␣
 ↪random_state=123,
                                                    shuffle=True)

# Split part of train into validation set
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train,
                                                  test_size = 0.125,␣
 ↪random_state=123,
                                                  shuffle=True)

X_train = np.array(X_train, dtype=float)
X_val = np.array(X_val, dtype=float)
X_test = np.array(X_test, dtype=float)
```

[84]:
```
# Create Datasets to input to loaders
train_dataset = Dataset(X_train, y_train)
val_dataset = Dataset(X_val, y_val)
test_dataset = Dataset(X_test, y_test)

# Create data loaders
train_loader = DataLoader(train_dataset, batch_size=100, shuffle=True,␣
↪num_workers=0)
val_loader = DataLoader(val_dataset, batch_size=100, shuffle=True,␣
↪num_workers=0)
test_loader = DataLoader(test_dataset, batch_size=100, shuffle=True,␣
↪num_workers=0)
```

1.1.2 Part b

Create a Python class for our neural network model. The network should have 1 input layer, 1 hidden layer, and 1 output layer. You are free to choose the size of the hidden layer (it may affect the performance). Use ReLU as the activation function (torch.relu).

[85]:
```
input_size = 41
hidden_size = 64
num_classes = 23
```

[86]:
```
import torch.nn as nn

# Any Pytorch's network class is an extension of the torch.nn.Module parent␣
↪class.
# To define a network class, you need to define at least 2 methods:
# an __init__() method (constructor) and a forward() method
class SimpleNetwork(torch.nn.Module):
    # Create the network class by filling in this block of code

    # Create the constructor. Add any additional arguments as you wish
    def __init__(self, input_size, hidden_size, num_classes):
        super().__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.l2 = nn.Linear(hidden_size, num_classes)

    # Define the feed forward function.
    # x is the input example/examples.
    # Add any additional arguments as you wish.
    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        return out
```

1.1.3 Part c

Train the network using the training dataset. Use the SGD optimizer and CrossEntropyLoss. After each epoch, record the current loss and the current training accuracy. The current training accuracy is obtained by evaluating the model on the train dataset. Use the DataLoaders defined in part a to efficiently pass training and testing data.

You can learn about the available optimizers at: https://pytorch.org/docs/stable/optim.html

You can learn about the available loss functions at: https://pytorch.org/docs/stable/nn.html#loss-functions

[87]:
```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = SimpleNetwork(input_size, hidden_size, num_classes).to(device)

LEARNING_RATE = 0.01
EPOCHS = 100

# number of parameters in model is equal to number of hidden channels
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=LEARNING_RATE)

training_acc = []
validation_acc = []
losses = []
n_total_steps = len(train_loader)

for epoch in range(EPOCHS):
    for i, (features, labels) in enumerate(train_loader):

        features = features.to(device)
        labels = labels.to(device)

        # Forward pass
        outputs = model(features.float())
        loss = criterion(outputs, labels)

        # Reset the gradients, backpropogate, optimize
        optimizer.zero_grad()
        loss.backward()  # backwards pass
        optimizer.step()  # gradient descent

        if i == 0:
            losses.append(loss.item())
            print (f'Epoch [{epoch+1}/{EPOCHS}], Loss: {loss.item():.4f}')
            # Compute train accuracy
            with torch.no_grad():
                n_correct = 0
                n_samples = 0
```

```
            for features, labels in train_loader:
                features = features.to(device)
                labels = labels.to(device)
                outputs = model(features.float())
                _, predicted = torch.max(outputs.data, 1)
                n_samples += labels.size(0)
                n_correct += (predicted == labels).sum().item()

        train_acc = n_correct / n_samples

        training_acc.append(train_acc)
        print(f'Accuracy of the network on training set: {100*train_acc:.
↪4f} %')

        with torch.no_grad():
            n_correct = 0
            n_samples = 0
            for features, labels in val_loader:
                features = features.to(device)
                labels = labels.to(device)
                outputs = model(features.float())
                _, predicted = torch.max(outputs.data, 1)
                n_samples += labels.size(0)
                n_correct += (predicted == labels).sum().item()

        val_acc = n_correct / n_samples
        validation_acc.append(val_acc)
        print(f'Accuracy of the network on validation set: {100*val_acc:.4}␣
↪%')
```

```
Epoch [1/100], Loss: 112.0067
Accuracy of the network on training set: 64.9254 %
Accuracy of the network on validation set: 65.06 %
Epoch [2/100], Loss: 0.5963
Accuracy of the network on training set: 80.2319 %
Accuracy of the network on validation set: 79.94 %
Epoch [3/100], Loss: 0.4339
Accuracy of the network on training set: 81.0783 %
Accuracy of the network on validation set: 80.8 %
Epoch [4/100], Loss: 0.6055
Accuracy of the network on training set: 80.1246 %
Accuracy of the network on validation set: 79.87 %
Epoch [5/100], Loss: 0.4929
Accuracy of the network on training set: 80.6246 %
Accuracy of the network on validation set: 80.33 %
Epoch [6/100], Loss: 0.4164
Accuracy of the network on training set: 80.9198 %
Accuracy of the network on validation set: 80.66 %
```

```
Epoch [7/100], Loss: 0.4200
Accuracy of the network on training set: 80.9103 %
Accuracy of the network on validation set: 80.62 %
Epoch [8/100], Loss: 0.3959
Accuracy of the network on training set: 80.8518 %
Accuracy of the network on validation set: 80.56 %
Epoch [9/100], Loss: 0.6321
Accuracy of the network on training set: 80.7986 %
Accuracy of the network on validation set: 80.55 %
Epoch [10/100], Loss: 0.4591
Accuracy of the network on training set: 81.0919 %
Accuracy of the network on validation set: 80.78 %
Epoch [11/100], Loss: 0.4037
Accuracy of the network on training set: 81.1202 %
Accuracy of the network on validation set: 80.82 %
Epoch [12/100], Loss: 0.5305
Accuracy of the network on training set: 80.0899 %
Accuracy of the network on validation set: 79.8 %
Epoch [13/100], Loss: 0.5205
Accuracy of the network on training set: 80.1379 %
Accuracy of the network on validation set: 79.87 %
Epoch [14/100], Loss: 0.4428
Accuracy of the network on training set: 80.1503 %
Accuracy of the network on validation set: 79.88 %
Epoch [15/100], Loss: 0.5225
Accuracy of the network on training set: 80.8741 %
Accuracy of the network on validation set: 80.59 %
Epoch [16/100], Loss: 0.4725
Accuracy of the network on training set: 80.9643 %
Accuracy of the network on validation set: 80.69 %
Epoch [17/100], Loss: 0.3620
Accuracy of the network on training set: 81.0199 %
Accuracy of the network on validation set: 80.75 %
Epoch [18/100], Loss: 0.4256
Accuracy of the network on training set: 80.8741 %
Accuracy of the network on validation set: 80.6 %
Epoch [19/100], Loss: 0.5597
Accuracy of the network on training set: 80.9105 %
Accuracy of the network on validation set: 80.65 %
Epoch [20/100], Loss: 0.4471
Accuracy of the network on training set: 81.0251 %
Accuracy of the network on validation set: 80.76 %
Epoch [21/100], Loss: 0.5164
Accuracy of the network on training set: 81.0285 %
Accuracy of the network on validation set: 80.75 %
Epoch [22/100], Loss: 0.5334
Accuracy of the network on training set: 81.0421 %
Accuracy of the network on validation set: 80.77 %
```

```
Epoch [23/100], Loss: 0.3617
Accuracy of the network on training set: 81.0919 %
Accuracy of the network on validation set: 80.81 %
Epoch [24/100], Loss: 0.5227
Accuracy of the network on training set: 80.9507 %
Accuracy of the network on validation set: 80.68 %
Epoch [25/100], Loss: 0.4347
Accuracy of the network on training set: 81.0465 %
Accuracy of the network on validation set: 80.78 %
Epoch [26/100], Loss: 0.5830
Accuracy of the network on training set: 81.0459 %
Accuracy of the network on validation set: 80.77 %
Epoch [27/100], Loss: 0.5943
Accuracy of the network on training set: 81.0548 %
Accuracy of the network on validation set: 80.78 %
Epoch [28/100], Loss: 0.4437
Accuracy of the network on training set: 81.1824 %
Accuracy of the network on validation set: 80.89 %
Epoch [29/100], Loss: 0.5831
Accuracy of the network on training set: 80.6754 %
Accuracy of the network on validation set: 80.39 %
Epoch [30/100], Loss: 0.4350
Accuracy of the network on training set: 81.0233 %
Accuracy of the network on validation set: 80.75 %
Epoch [31/100], Loss: 0.4116
Accuracy of the network on training set: 81.2318 %
Accuracy of the network on validation set: 80.94 %
Epoch [32/100], Loss: 0.4170
Accuracy of the network on training set: 81.2425 %
Accuracy of the network on validation set: 80.94 %
Epoch [33/100], Loss: 0.4724
Accuracy of the network on training set: 81.2711 %
Accuracy of the network on validation set: 80.98 %
Epoch [34/100], Loss: 0.4598
Accuracy of the network on training set: 81.1832 %
Accuracy of the network on validation set: 80.89 %
Epoch [35/100], Loss: 0.1042
Accuracy of the network on training set: 98.9477 %
Accuracy of the network on validation set: 98.94 %
Epoch [36/100], Loss: 0.6946
Accuracy of the network on training set: 42.0853 %
Accuracy of the network on validation set: 42.06 %
Epoch [37/100], Loss: 0.5031
Accuracy of the network on training set: 81.0933 %
Accuracy of the network on validation set: 80.83 %
Epoch [38/100], Loss: 0.4371
Accuracy of the network on training set: 81.1847 %
Accuracy of the network on validation set: 80.88 %
```

```
Epoch [39/100], Loss: 0.3963
Accuracy of the network on training set: 81.1827 %
Accuracy of the network on validation set: 80.88 %
Epoch [40/100], Loss: 0.4179
Accuracy of the network on training set: 81.1913 %
Accuracy of the network on validation set: 80.9 %
Epoch [41/100], Loss: 0.5508
Accuracy of the network on training set: 80.3487 %
Accuracy of the network on validation set: 80.07 %
Epoch [42/100], Loss: 0.5251
Accuracy of the network on training set: 81.0638 %
Accuracy of the network on validation set: 80.77 %
Epoch [43/100], Loss: 0.4144
Accuracy of the network on training set: 81.0728 %
Accuracy of the network on validation set: 80.79 %
Epoch [44/100], Loss: 0.5570
Accuracy of the network on training set: 81.0930 %
Accuracy of the network on validation set: 80.82 %
Epoch [45/100], Loss: 0.5640
Accuracy of the network on training set: 81.0762 %
Accuracy of the network on validation set: 80.79 %
Epoch [46/100], Loss: 0.4187
Accuracy of the network on training set: 81.0600 %
Accuracy of the network on validation set: 80.78 %
Epoch [47/100], Loss: 0.3815
Accuracy of the network on training set: 81.1983 %
Accuracy of the network on validation set: 80.92 %
Epoch [48/100], Loss: 0.3395
Accuracy of the network on training set: 81.0974 %
Accuracy of the network on validation set: 80.82 %
Epoch [49/100], Loss: 0.4505
Accuracy of the network on training set: 81.0950 %
Accuracy of the network on validation set: 80.79 %
Epoch [50/100], Loss: 0.5068
Accuracy of the network on training set: 81.2055 %
Accuracy of the network on validation set: 80.9 %
Epoch [51/100], Loss: 0.4581
Accuracy of the network on training set: 81.1054 %
Accuracy of the network on validation set: 80.82 %
Epoch [52/100], Loss: 0.4862
Accuracy of the network on training set: 81.1052 %
Accuracy of the network on validation set: 80.82 %
Epoch [53/100], Loss: 0.5072
Accuracy of the network on training set: 81.1286 %
Accuracy of the network on validation set: 80.85 %
Epoch [54/100], Loss: 0.4483
Accuracy of the network on training set: 81.2440 %
Accuracy of the network on validation set: 80.94 %
```

```
Epoch [55/100], Loss: 0.3213
Accuracy of the network on training set: 83.2861 %
Accuracy of the network on validation set: 83.02 %
Epoch [56/100], Loss: 0.3901
Accuracy of the network on training set: 83.6129 %
Accuracy of the network on validation set: 83.34 %
Epoch [57/100], Loss: 0.5030
Accuracy of the network on training set: 83.4113 %
Accuracy of the network on validation set: 83.14 %
Epoch [58/100], Loss: 0.4304
Accuracy of the network on training set: 80.5549 %
Accuracy of the network on validation set: 80.26 %
Epoch [59/100], Loss: 0.4358
Accuracy of the network on training set: 83.2766 %
Accuracy of the network on validation set: 83.06 %
Epoch [60/100], Loss: 0.3693
Accuracy of the network on training set: 83.3454 %
Accuracy of the network on validation set: 83.11 %
Epoch [61/100], Loss: 0.4241
Accuracy of the network on training set: 83.4159 %
Accuracy of the network on validation set: 83.18 %
Epoch [62/100], Loss: 0.4106
Accuracy of the network on training set: 83.7462 %
Accuracy of the network on validation set: 83.49 %
Epoch [63/100], Loss: 0.4020
Accuracy of the network on training set: 80.3270 %
Accuracy of the network on validation set: 80.1 %
Epoch [64/100], Loss: 0.4535
Accuracy of the network on training set: 80.4875 %
Accuracy of the network on validation set: 80.25 %
Epoch [65/100], Loss: 0.5326
Accuracy of the network on training set: 82.0201 %
Accuracy of the network on validation set: 81.73 %
Epoch [66/100], Loss: 0.4580
Accuracy of the network on training set: 80.4936 %
Accuracy of the network on validation set: 80.27 %
Epoch [67/100], Loss: 0.6374
Accuracy of the network on training set: 80.3845 %
Accuracy of the network on validation set: 80.11 %
Epoch [68/100], Loss: 0.3933
Accuracy of the network on training set: 81.2176 %
Accuracy of the network on validation set: 80.94 %
Epoch [69/100], Loss: 0.5192
Accuracy of the network on training set: 81.6228 %
Accuracy of the network on validation set: 81.34 %
Epoch [70/100], Loss: 0.5353
Accuracy of the network on training set: 81.6265 %
Accuracy of the network on validation set: 81.31 %
```

```
Epoch [71/100], Loss: 0.4523
Accuracy of the network on training set: 81.5832 %
Accuracy of the network on validation set: 81.26 %
Epoch [72/100], Loss: 0.4149
Accuracy of the network on training set: 82.5114 %
Accuracy of the network on validation set: 82.27 %
Epoch [73/100], Loss: 0.4766
Accuracy of the network on training set: 83.1965 %
Accuracy of the network on validation set: 82.98 %
Epoch [74/100], Loss: 0.5230
Accuracy of the network on training set: 83.3168 %
Accuracy of the network on validation set: 83.11 %
Epoch [75/100], Loss: 0.4658
Accuracy of the network on training set: 83.2005 %
Accuracy of the network on validation set: 82.98 %
Epoch [76/100], Loss: 0.4168
Accuracy of the network on training set: 81.5381 %
Accuracy of the network on validation set: 81.23 %
Epoch [77/100], Loss: 0.4869
Accuracy of the network on training set: 83.5076 %
Accuracy of the network on validation set: 83.27 %
Epoch [78/100], Loss: 0.5952
Accuracy of the network on training set: 83.6580 %
Accuracy of the network on validation set: 83.44 %
Epoch [79/100], Loss: 0.4281
Accuracy of the network on training set: 83.6837 %
Accuracy of the network on validation set: 83.46 %
Epoch [80/100], Loss: 0.4975
Accuracy of the network on training set: 83.5652 %
Accuracy of the network on validation set: 83.34 %
Epoch [81/100], Loss: 0.4093
Accuracy of the network on training set: 83.5594 %
Accuracy of the network on validation set: 83.31 %
Epoch [82/100], Loss: 0.4961
Accuracy of the network on training set: 83.7222 %
Accuracy of the network on validation set: 83.49 %
Epoch [83/100], Loss: 0.3804
Accuracy of the network on training set: 83.5897 %
Accuracy of the network on validation set: 83.37 %
Epoch [84/100], Loss: 0.5407
Accuracy of the network on training set: 83.4547 %
Accuracy of the network on validation set: 83.21 %
Epoch [85/100], Loss: 0.5036
Accuracy of the network on training set: 83.7022 %
Accuracy of the network on validation set: 83.46 %
Epoch [86/100], Loss: 0.4343
Accuracy of the network on training set: 83.4504 %
Accuracy of the network on validation set: 83.2 %
```

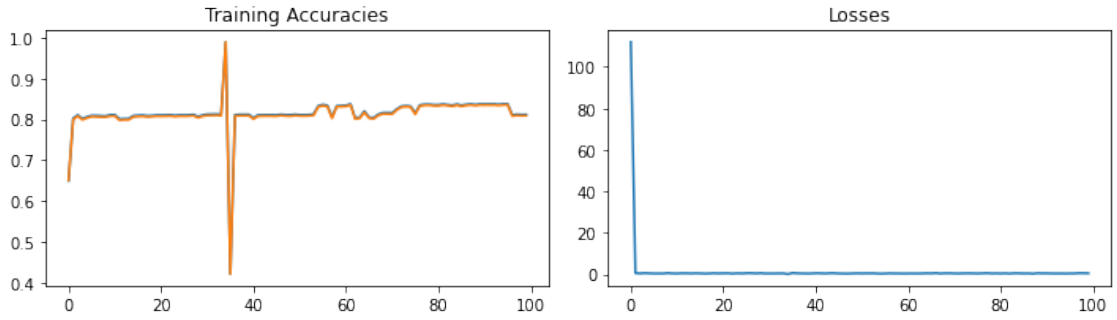
```
Epoch [87/100], Loss: 0.4536
Accuracy of the network on training set: 83.6372 %
Accuracy of the network on validation set: 83.4 %
Epoch [88/100], Loss: 0.3002
Accuracy of the network on training set: 83.7453 %
Accuracy of the network on validation set: 83.51 %
Epoch [89/100], Loss: 0.5756
Accuracy of the network on training set: 83.6068 %
Accuracy of the network on validation set: 83.38 %
Epoch [90/100], Loss: 0.4668
Accuracy of the network on training set: 83.7502 %
Accuracy of the network on validation set: 83.52 %
Epoch [91/100], Loss: 0.5210
Accuracy of the network on training set: 83.7210 %
Accuracy of the network on validation set: 83.48 %
Epoch [92/100], Loss: 0.4359
Accuracy of the network on training set: 83.7442 %
Accuracy of the network on validation set: 83.5 %
Epoch [93/100], Loss: 0.4162
Accuracy of the network on training set: 83.7476 %
Accuracy of the network on validation set: 83.5 %
Epoch [94/100], Loss: 0.4263
Accuracy of the network on training set: 83.6583 %
Accuracy of the network on validation set: 83.42 %
Epoch [95/100], Loss: 0.3675
Accuracy of the network on training set: 83.7615 %
Accuracy of the network on validation set: 83.51 %
Epoch [96/100], Loss: 0.4359
Accuracy of the network on training set: 83.7887 %
Accuracy of the network on validation set: 83.56 %
Epoch [97/100], Loss: 0.4512
Accuracy of the network on training set: 81.0644 %
Accuracy of the network on validation set: 80.78 %
Epoch [98/100], Loss: 0.5873
Accuracy of the network on training set: 81.2150 %
Accuracy of the network on validation set: 80.93 %
Epoch [99/100], Loss: 0.5753
Accuracy of the network on training set: 81.1517 %
Accuracy of the network on validation set: 80.87 %
Epoch [100/100], Loss: 0.5022
Accuracy of the network on training set: 81.1749 %
Accuracy of the network on validation set: 80.9 %
```

Plot how the loss and the training accuracy and the validation accuracy change over the epochs. Is there a point where overfitting occurs? If you cannot spot one, answer no.

[88]:
```
import matplotlib.pyplot as plt
```

```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(10,3))
ax[0].plot(range(EPOCHS), training_acc, label = "Train Accuracy")
ax[0].plot(range(EPOCHS), validation_acc, label = "Validation Accuracy")
ax[0].set_title('Training Accuracies')
ax[1].plot(range(EPOCHS), losses)
ax[1].set_title('Losses')
fig.tight_layout()

plt.show()
```



It looks like there is point where overfitting occurs at roughly the 35th point, and as a result, the model's accuracy suffers immediately after. However, it is able to still converge back to a fairly consistent result

1.1.4 Part d

Evaluate the model on the test dataset. Print out the accuracy. Does this accuracy agrees with the training accuracy showed on the plot?

[89]:
```
def test(model, criterion, test_loader):
    # Turn the model to testing mode (gradients will not be calculated)
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for features, labels in test_loader:
            features = features.to(device)
            labels = labels.to(device)
            outputs = model(features.float())
            test_loss += criterion(outputs, labels).item()  # sum up batch loss
            pred = outputs.argmax(dim=1, keepdim=True)  # get the index of the␣
↪max log-probability
            correct += pred.eq(labels.view_as(pred)).sum().item()
```

```
    print(f'Accuracy of the network on test set: {100 * correct /␣
↪len(test_loader.dataset)}')

criterion = nn.CrossEntropyLoss()
test(model, criterion, test_loader)
```

```
Accuracy of the network on test set: 81.0414452709883
```

Based on our results from the training data, the accuarcy here lines up.

1.2 Problem 2

In this problem, we will investigate the effects of various common hyperparameters on the performance of a neural network. In the following cell, you can find a network class already defined for you. You can initiate network instances with different hyperparameters by changing the constructor's arguments.

You are graded based on how you implement and execute the experiments. Since there is some randomness in initiating and training a neural network, there is no guarantee that you will get an expected result for an experiment or that your results should be similar to those of your peers. The expected outcome is that you execute the experiments correctly and the conclusion you get are consistent with your results. For each experiments, try to run the code multiple times and record the average results like what we did in Homework 2 (it will take some time to run, as expected when training any neural network).

[21]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F

class mnist_network(torch.nn.Module):

    def __init__(self, num_hidden_layers=1, layer_size = 100, activation=None):
        super(mnist_network, self).__init__()
        # layers of the network
        self.num_hidden_layers = num_hidden_layers
        self.layer_size = layer_size
        self.activation = activation

        if(self.activation == 'relu'):
            self.activation = F.relu
        elif(self.activation == 'tanh'):
            self.activation = torch.tanh

        self.layers = nn.ModuleList([nn.Linear(784,self.layer_size)])
        for i in range(1, self.num_hidden_layers):
            self.layers.append(nn.Linear(self.layer_size,self.layer_size))
        self.layers.append(nn.Linear(self.layer_size,10))

    def forward(self, x):
```

```
        # converting each image into a vector
        batch_size = x.shape[0]
        x = x.reshape(batch_size,-1)
        # rest of the forward pass
        for i in range(self.num_hidden_layers+1):
            x = self.layers[i](x)
            if(self.activation is not None):
                x = self.activation(x)
        return x
```

Run the following code to load the MNIST dataset. For the sake of simplicity, we do not have a validation set in this problem.

[22]:
```
import torch
from torchvision import datasets, transforms

transform=transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.1307,), (0.3081,))
        ])

dataset1 = datasets.MNIST('../data', train=True, download=True,
                    transform=transform)
dataset2 = datasets.MNIST('../data', train=False,
                    transform=transform)

# DataLoader is a nice tool provided by PyTorch for passing training or testing␣
 ↪examples
train_loader = torch.utils.data.DataLoader(dataset1, batch_size=64)
test_loader = torch.utils.data.DataLoader(dataset2, batch_size=1000)
```

Here is an example of training and testing a model:

[40]:
```
import torch.nn as nn
import torch.optim as optim
import itertools

def train(model, criterion, train_loader, optimizer, epoch):
    # Turn the model to training mode (gradients will be calculated)
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # We have to call zero_grad() on the optimizer to remove gradients from␣
 ↪the previous data pass.
        # Otherwise, the gradients will be accumulated throughout many passes.
        optimizer.zero_grad()
        # Pass in the data and obtain the output.
        # When you pass the data directly by calling model(data), the model␣
 ↪will internally pass the data through the forward() function.
```

```
        output = model(data)
        # Compare the output and the ground truth and calculate the loss.
        loss = criterion(output, target)
        # From the calculated loss, call backward() to calculate the gradients␣
↪for all the paramters in the network.
        loss.backward()
        # Update the parameters according to the gradients.
        optimizer.step()
        if batch_idx % 100 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))


def test(model, criterion, test_loader):
    # Turn the model to testing mode (gradients will not be calculated)
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            output = model(data)
            test_loss += criterion(output, target).item()  # sum up batch loss
            pred = output.argmax(dim=1, keepdim=True)  # get the index of the␣
↪max log-probability
            correct += pred.eq(target.view_as(pred)).sum().item()
    accuracy = 100. * correct / len(test_loader.dataset)
    test_loss /= len(test_loader.dataset)

    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.
↪format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))

    return accuracy

# Number of training epochs
epochs = 5
# Learning rate
lr = 0.01

# Create the model
model = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')

# Define the training and testing loss
train_criterion = nn.CrossEntropyLoss()
test_criterion = nn.CrossEntropyLoss(reduction='sum')
```

```
# Define the optimizer
# We have to specify the learning rate and the parameters that the optimizer␣
 ↪should update during training.
# In this case, we specify that the optimizer should update all the parameters␣
 ↪from our model.
optimizer = optim.SGD(model.parameters(),lr=lr)

for epoch in range(1, epochs + 1):
    # Training
    train(model, train_criterion, train_loader, optimizer, epoch)
    # Testing
    test(model, test_criterion, test_loader)
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.312148
Train Epoch: 1 [6400/60000 (11%)]       Loss: 1.661013
Train Epoch: 1 [12800/60000 (21%)]      Loss: 1.153798
Train Epoch: 1 [19200/60000 (32%)]      Loss: 0.890773
Train Epoch: 1 [25600/60000 (43%)]      Loss: 0.601178
Train Epoch: 1 [32000/60000 (53%)]      Loss: 0.625142
Train Epoch: 1 [38400/60000 (64%)]      Loss: 0.615317
Train Epoch: 1 [44800/60000 (75%)]      Loss: 0.800792
Train Epoch: 1 [51200/60000 (85%)]      Loss: 0.657874
Train Epoch: 1 [57600/60000 (96%)]      Loss: 0.708585

Test set: Average loss: 0.5738, Accuracy: 8494/10000 (85%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 0.560406
Train Epoch: 2 [6400/60000 (11%)]       Loss: 0.346600
Train Epoch: 2 [12800/60000 (21%)]      Loss: 0.263422
Train Epoch: 2 [19200/60000 (32%)]      Loss: 0.404671
Train Epoch: 2 [25600/60000 (43%)]      Loss: 0.278740
Train Epoch: 2 [32000/60000 (53%)]      Loss: 0.366671
Train Epoch: 2 [38400/60000 (64%)]      Loss: 0.221094
Train Epoch: 2 [44800/60000 (75%)]      Loss: 0.428373
Train Epoch: 2 [51200/60000 (85%)]      Loss: 0.348852
Train Epoch: 2 [57600/60000 (96%)]      Loss: 0.437842

Test set: Average loss: 0.3023, Accuracy: 9119/10000 (91%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 0.201378
Train Epoch: 3 [6400/60000 (11%)]       Loss: 0.259739
Train Epoch: 3 [12800/60000 (21%)]      Loss: 0.195460
Train Epoch: 3 [19200/60000 (32%)]      Loss: 0.329895
Train Epoch: 3 [25600/60000 (43%)]      Loss: 0.216567
Train Epoch: 3 [32000/60000 (53%)]      Loss: 0.321375
Train Epoch: 3 [38400/60000 (64%)]      Loss: 0.194861
```

```
Train Epoch: 3 [44800/60000 (75%)]      Loss: 0.403567
Train Epoch: 3 [51200/60000 (85%)]      Loss: 0.313561
Train Epoch: 3 [57600/60000 (96%)]      Loss: 0.432743

Test set: Average loss: 0.2783, Accuracy: 9190/10000 (92%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 0.164790
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.240532
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.171940
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.292491
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.187159
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.299042
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.178126
Train Epoch: 4 [44800/60000 (75%)]      Loss: 0.389411
Train Epoch: 4 [51200/60000 (85%)]      Loss: 0.284042
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.414966

Test set: Average loss: 0.2626, Accuracy: 9238/10000 (92%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.143980
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.230140
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.159278
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.260374
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.167460
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.282735
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.163291
Train Epoch: 5 [44800/60000 (75%)]      Loss: 0.373959
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.258135
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.393625

Test set: Average loss: 0.2491, Accuracy: 9261/10000 (93%)
```

1.3 Part a

First, we will investigate the effect of varying the size of the hidden layer. Create 3 one-hidden-layer networks with the sizes of the hidden layers being 5, 20, 50, respectively. We will call these the 5-network, the 20-network, and the 50-network. All networks should use ReLU activation.

Train the 5-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[41]:
```
# Number of training epochs
epochs = 10
# Learning rate
lr = 0.001

# Create the model
```

```
model_5 = mnist_network(num_hidden_layers=1,layer_size=5,activation='relu')

# Define the training and testing loss
train_criterion = nn.CrossEntropyLoss()
test_criterion = nn.CrossEntropyLoss(reduction='sum')

# Define the optimizer
# We have to specify the learning rate and the parameters that the optimizer␣
 ↪should update during training.
# In this case, we specify that the optimizer should update all the parameters␣
 ↪from our model.
optimizer = optim.SGD(model_5.parameters(),lr=lr)

train_5 = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_5, train_criterion, train_loader, optimizer, epoch)
        # Testing
        train_5.append(test(model_5, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.321440
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.208445
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.209888
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.199331
Train Epoch: 1 [25600/60000 (43%)]      Loss: 1.964409
Train Epoch: 1 [32000/60000 (53%)]      Loss: 1.998427
Train Epoch: 1 [38400/60000 (64%)]      Loss: 2.061806
Train Epoch: 1 [44800/60000 (75%)]      Loss: 2.065061
Train Epoch: 1 [51200/60000 (85%)]      Loss: 2.017438
Train Epoch: 1 [57600/60000 (96%)]      Loss: 2.168442

Test set: Average loss: 1.9569, Accuracy: 22724/60000 (38%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 1.875798
Train Epoch: 2 [6400/60000 (11%)]       Loss: 1.841540
Train Epoch: 2 [12800/60000 (21%)]      Loss: 1.966337
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.961886
Train Epoch: 2 [25600/60000 (43%)]      Loss: 1.657763
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.735698
Train Epoch: 2 [38400/60000 (64%)]      Loss: 1.806366
Train Epoch: 2 [44800/60000 (75%)]      Loss: 1.913833
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.841156
Train Epoch: 2 [57600/60000 (96%)]      Loss: 1.993012

Test set: Average loss: 1.7405, Accuracy: 25020/60000 (42%)
```

```
Train Epoch: 3 [0/60000 (0%)]   Loss: 1.595093
Train Epoch: 3 [6400/60000 (11%)]       Loss: 1.622040
Train Epoch: 3 [12800/60000 (21%)]      Loss: 1.774720
Train Epoch: 3 [19200/60000 (32%)]      Loss: 1.788640
Train Epoch: 3 [25600/60000 (43%)]      Loss: 1.467511
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.523957
Train Epoch: 3 [38400/60000 (64%)]      Loss: 1.610221
Train Epoch: 3 [44800/60000 (75%)]      Loss: 1.786236
Train Epoch: 3 [51200/60000 (85%)]      Loss: 1.711848
Train Epoch: 3 [57600/60000 (96%)]      Loss: 1.908397

Test set: Average loss: 1.6107, Accuracy: 26327/60000 (44%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 1.445289
Train Epoch: 4 [6400/60000 (11%)]       Loss: 1.504944
Train Epoch: 4 [12800/60000 (21%)]      Loss: 1.647934
Train Epoch: 4 [19200/60000 (32%)]      Loss: 1.706139
Train Epoch: 4 [25600/60000 (43%)]      Loss: 1.368004
Train Epoch: 4 [32000/60000 (53%)]      Loss: 1.414211
Train Epoch: 4 [38400/60000 (64%)]      Loss: 1.508293
Train Epoch: 4 [44800/60000 (75%)]      Loss: 1.696847
Train Epoch: 4 [51200/60000 (85%)]      Loss: 1.639622
Train Epoch: 4 [57600/60000 (96%)]      Loss: 1.865132

Test set: Average loss: 1.5359, Accuracy: 27569/60000 (46%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 1.366419
Train Epoch: 5 [6400/60000 (11%)]       Loss: 1.438758
Train Epoch: 5 [12800/60000 (21%)]      Loss: 1.561051
Train Epoch: 5 [19200/60000 (32%)]      Loss: 1.658328
Train Epoch: 5 [25600/60000 (43%)]      Loss: 1.306920
Train Epoch: 5 [32000/60000 (53%)]      Loss: 1.341393
Train Epoch: 5 [38400/60000 (64%)]      Loss: 1.447988
Train Epoch: 5 [44800/60000 (75%)]      Loss: 1.632639
Train Epoch: 5 [51200/60000 (85%)]      Loss: 1.592777
Train Epoch: 5 [57600/60000 (96%)]      Loss: 1.835964

Test set: Average loss: 1.4850, Accuracy: 28724/60000 (48%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 1.314261
Train Epoch: 6 [6400/60000 (11%)]       Loss: 1.392637
Train Epoch: 6 [12800/60000 (21%)]      Loss: 1.494554
Train Epoch: 6 [19200/60000 (32%)]      Loss: 1.626842
Train Epoch: 6 [25600/60000 (43%)]      Loss: 1.261460
Train Epoch: 6 [32000/60000 (53%)]      Loss: 1.292435
Train Epoch: 6 [38400/60000 (64%)]      Loss: 1.408106
Train Epoch: 6 [44800/60000 (75%)]      Loss: 1.585056
Train Epoch: 6 [51200/60000 (85%)]      Loss: 1.561042
```

```
Train Epoch: 6 [57600/60000 (96%)]      Loss: 1.813286

Test set: Average loss: 1.4467, Accuracy: 29744/60000 (50%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 1.273937
Train Epoch: 7 [6400/60000 (11%)]       Loss: 1.357681
Train Epoch: 7 [12800/60000 (21%)]      Loss: 1.441472
Train Epoch: 7 [19200/60000 (32%)]      Loss: 1.602588
Train Epoch: 7 [25600/60000 (43%)]      Loss: 1.224352
Train Epoch: 7 [32000/60000 (53%)]      Loss: 1.257230
Train Epoch: 7 [38400/60000 (64%)]      Loss: 1.381082
Train Epoch: 7 [44800/60000 (75%)]      Loss: 1.548830
Train Epoch: 7 [51200/60000 (85%)]      Loss: 1.536857
Train Epoch: 7 [57600/60000 (96%)]      Loss: 1.794150

Test set: Average loss: 1.4165, Accuracy: 30452/60000 (51%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 1.240835
Train Epoch: 8 [6400/60000 (11%)]       Loss: 1.331462
Train Epoch: 8 [12800/60000 (21%)]      Loss: 1.397474
Train Epoch: 8 [19200/60000 (32%)]      Loss: 1.583604
Train Epoch: 8 [25600/60000 (43%)]      Loss: 1.194715
Train Epoch: 8 [32000/60000 (53%)]      Loss: 1.230296
Train Epoch: 8 [38400/60000 (64%)]      Loss: 1.361830
Train Epoch: 8 [44800/60000 (75%)]      Loss: 1.519452
Train Epoch: 8 [51200/60000 (85%)]      Loss: 1.518716
Train Epoch: 8 [57600/60000 (96%)]      Loss: 1.778399

Test set: Average loss: 1.3922, Accuracy: 30887/60000 (51%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 1.213635
Train Epoch: 9 [6400/60000 (11%)]       Loss: 1.311694
Train Epoch: 9 [12800/60000 (21%)]      Loss: 1.360272
Train Epoch: 9 [19200/60000 (32%)]      Loss: 1.568353
Train Epoch: 9 [25600/60000 (43%)]      Loss: 1.171842
Train Epoch: 9 [32000/60000 (53%)]      Loss: 1.209202
Train Epoch: 9 [38400/60000 (64%)]      Loss: 1.348386
Train Epoch: 9 [44800/60000 (75%)]      Loss: 1.495468
Train Epoch: 9 [51200/60000 (85%)]      Loss: 1.504374
Train Epoch: 9 [57600/60000 (96%)]      Loss: 1.764977

Test set: Average loss: 1.3724, Accuracy: 31155/60000 (52%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 1.191072
Train Epoch: 10 [6400/60000 (11%)]      Loss: 1.298062
Train Epoch: 10 [12800/60000 (21%)]     Loss: 1.329031
Train Epoch: 10 [19200/60000 (32%)]     Loss: 1.555923
Train Epoch: 10 [25600/60000 (43%)]     Loss: 1.155095
```

```
Train Epoch: 10 [32000/60000 (53%)]     Loss: 1.191846
Train Epoch: 10 [38400/60000 (64%)]     Loss: 1.338663
Train Epoch: 10 [44800/60000 (75%)]     Loss: 1.475884
Train Epoch: 10 [51200/60000 (85%)]     Loss: 1.492763
Train Epoch: 10 [57600/60000 (96%)]     Loss: 1.754420

Test set: Average loss: 1.3561, Accuracy: 31408/60000 (52%)
```

Test the trained 5-network on the test data. Print out the accuracy.

[42]:
```
test(model_5, test_criterion, test_loader)
```

```
Test set: Average loss: 1.3408, Accuracy: 5252/10000 (53%)
```

[42]:
```
52.52
```

Train the 20-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[43]:
```
model_20 = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')
optimizer = optim.SGD(model_20.parameters(),lr=lr)

train_20 = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_20, train_criterion, train_loader, optimizer, epoch)
        # Testing
        train_20.append(test(model_20, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.317551
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.228011
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.240905
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.129658
Train Epoch: 1 [25600/60000 (43%)]      Loss: 2.076414
Train Epoch: 1 [32000/60000 (53%)]      Loss: 2.021779
Train Epoch: 1 [38400/60000 (64%)]      Loss: 1.940824
Train Epoch: 1 [44800/60000 (75%)]      Loss: 2.206467
Train Epoch: 1 [51200/60000 (85%)]      Loss: 1.881568
Train Epoch: 1 [57600/60000 (96%)]      Loss: 1.686215

Test set: Average loss: 1.7618, Accuracy: 29224/60000 (49%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 1.756814
Train Epoch: 2 [6400/60000 (11%)]       Loss: 1.633259
Train Epoch: 2 [12800/60000 (21%)]      Loss: 1.617725
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.500919
```

```
Train Epoch: 2 [25600/60000 (43%)]      Loss: 1.376581
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.399405
Train Epoch: 2 [38400/60000 (64%)]      Loss: 1.417740
Train Epoch: 2 [44800/60000 (75%)]      Loss: 1.695238
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.391897
Train Epoch: 2 [57600/60000 (96%)]      Loss: 1.231258

Test set: Average loss: 1.3073, Accuracy: 37845/60000 (63%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 1.347686
Train Epoch: 3 [6400/60000 (11%)]       Loss: 1.257826
Train Epoch: 3 [12800/60000 (21%)]      Loss: 1.144017
Train Epoch: 3 [19200/60000 (32%)]      Loss: 1.200168
Train Epoch: 3 [25600/60000 (43%)]      Loss: 1.031729
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.119624
Train Epoch: 3 [38400/60000 (64%)]      Loss: 1.194929
Train Epoch: 3 [44800/60000 (75%)]      Loss: 1.426228
Train Epoch: 3 [51200/60000 (85%)]      Loss: 1.201401
Train Epoch: 3 [57600/60000 (96%)]      Loss: 1.064206

Test set: Average loss: 1.1284, Accuracy: 39874/60000 (66%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 1.163255
Train Epoch: 4 [6400/60000 (11%)]       Loss: 1.096178
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.941166
Train Epoch: 4 [19200/60000 (32%)]      Loss: 1.068254
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.882248
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.991922
Train Epoch: 4 [38400/60000 (64%)]      Loss: 1.078296
Train Epoch: 4 [44800/60000 (75%)]      Loss: 1.291503
Train Epoch: 4 [51200/60000 (85%)]      Loss: 1.092905
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.972041

Test set: Average loss: 1.0271, Accuracy: 40855/60000 (68%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 1.044743
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.998121
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.823461
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.984143
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.790498
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.913644
Train Epoch: 5 [38400/60000 (64%)]      Loss: 1.003321
Train Epoch: 5 [44800/60000 (75%)]      Loss: 1.204571
Train Epoch: 5 [51200/60000 (85%)]      Loss: 1.013789
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.908981

Test set: Average loss: 0.9577, Accuracy: 41634/60000 (69%)
```

```
Train Epoch: 6 [0/60000 (0%)]   Loss: 0.957021
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.930360
Train Epoch: 6 [12800/60000 (21%)]      Loss: 0.745000
Train Epoch: 6 [19200/60000 (32%)]      Loss: 0.922439
Train Epoch: 6 [25600/60000 (43%)]      Loss: 0.728423
Train Epoch: 6 [32000/60000 (53%)]      Loss: 0.862371
Train Epoch: 6 [38400/60000 (64%)]      Loss: 0.946467
Train Epoch: 6 [44800/60000 (75%)]      Loss: 1.143029
Train Epoch: 6 [51200/60000 (85%)]      Loss: 0.956005
Train Epoch: 6 [57600/60000 (96%)]      Loss: 0.864212

Test set: Average loss: 0.9089, Accuracy: 42818/60000 (71%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.893874
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.886713
Train Epoch: 7 [12800/60000 (21%)]      Loss: 0.689993
Train Epoch: 7 [19200/60000 (32%)]      Loss: 0.881252
Train Epoch: 7 [25600/60000 (43%)]      Loss: 0.686079
Train Epoch: 7 [32000/60000 (53%)]      Loss: 0.828426
Train Epoch: 7 [38400/60000 (64%)]      Loss: 0.907147
Train Epoch: 7 [44800/60000 (75%)]      Loss: 1.097515
Train Epoch: 7 [51200/60000 (85%)]      Loss: 0.913405
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.831053

Test set: Average loss: 0.8737, Accuracy: 43894/60000 (73%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.846168
Train Epoch: 8 [6400/60000 (11%)]       Loss: 0.857395
Train Epoch: 8 [12800/60000 (21%)]      Loss: 0.650536
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.851669
Train Epoch: 8 [25600/60000 (43%)]      Loss: 0.656288
Train Epoch: 8 [32000/60000 (53%)]      Loss: 0.805344
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.879373
Train Epoch: 8 [44800/60000 (75%)]      Loss: 1.062770
Train Epoch: 8 [51200/60000 (85%)]      Loss: 0.881151
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.806592

Test set: Average loss: 0.8475, Accuracy: 44695/60000 (74%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.809421
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.835507
Train Epoch: 9 [12800/60000 (21%)]      Loss: 0.620108
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.829331
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.634724
Train Epoch: 9 [32000/60000 (53%)]      Loss: 0.788642
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.857706
Train Epoch: 9 [44800/60000 (75%)]      Loss: 1.035580
Train Epoch: 9 [51200/60000 (85%)]      Loss: 0.856389
```

```
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.788387

Test set: Average loss: 0.8272, Accuracy: 45331/60000 (76%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.780313
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.819248
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.596329
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.811330
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.618445
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.776778
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.839723
Train Epoch: 10 [44800/60000 (75%)]     Loss: 1.013773
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.837873
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.774376

Test set: Average loss: 0.8110, Accuracy: 45830/60000 (76%)
```

Test the trained 20-network on the test data. Print out the accuracy.

[44]:
```
test(model_20, test_criterion, test_loader)
```

```
Test set: Average loss: 0.7815, Accuracy: 7730/10000 (77%)
```

[44]:
```
77.3
```

Train the 50-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[45]:
```
model_50 = mnist_network(num_hidden_layers=1,layer_size=50,activation='relu')
optimizer = optim.SGD(model_50.parameters(),lr=lr)

train_50 = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_50, train_criterion, train_loader, optimizer, epoch)
        # Testing
        train_50.append(test(model_50, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.298186
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.266451
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.189259
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.049706
Train Epoch: 1 [25600/60000 (43%)]      Loss: 2.027826
Train Epoch: 1 [32000/60000 (53%)]      Loss: 1.910260
Train Epoch: 1 [38400/60000 (64%)]      Loss: 1.848434
Train Epoch: 1 [44800/60000 (75%)]      Loss: 1.915681
```

```
Train Epoch: 1 [51200/60000 (85%)]      Loss: 1.741885
Train Epoch: 1 [57600/60000 (96%)]      Loss: 1.581292

Test set: Average loss: 1.5920, Accuracy: 34350/60000 (57%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 1.628133
Train Epoch: 2 [6400/60000 (11%)]       Loss: 1.541206
Train Epoch: 2 [12800/60000 (21%)]      Loss: 1.418858
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.353191
Train Epoch: 2 [25600/60000 (43%)]      Loss: 1.233823
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.278089
Train Epoch: 2 [38400/60000 (64%)]      Loss: 1.314813
Train Epoch: 2 [44800/60000 (75%)]      Loss: 1.413110
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.310634
Train Epoch: 2 [57600/60000 (96%)]      Loss: 1.116786

Test set: Average loss: 1.1994, Accuracy: 39641/60000 (66%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 1.258184
Train Epoch: 3 [6400/60000 (11%)]       Loss: 1.198323
Train Epoch: 3 [12800/60000 (21%)]      Loss: 1.021117
Train Epoch: 3 [19200/60000 (32%)]      Loss: 1.089432
Train Epoch: 3 [25600/60000 (43%)]      Loss: 0.936937
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.037165
Train Epoch: 3 [38400/60000 (64%)]      Loss: 1.089594
Train Epoch: 3 [44800/60000 (75%)]      Loss: 1.202116
Train Epoch: 3 [51200/60000 (85%)]      Loss: 1.127207
Train Epoch: 3 [57600/60000 (96%)]      Loss: 0.949338

Test set: Average loss: 1.0377, Accuracy: 41147/60000 (69%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 1.078277
Train Epoch: 4 [6400/60000 (11%)]       Loss: 1.048785
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.849161
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.976227
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.805352
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.923850
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.976337
Train Epoch: 4 [44800/60000 (75%)]      Loss: 1.099885
Train Epoch: 4 [51200/60000 (85%)]      Loss: 1.025159
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.864643

Test set: Average loss: 0.9517, Accuracy: 42549/60000 (71%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.973912
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.966628
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.753371
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.913916
```

```
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.730780
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.861041
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.912786
Train Epoch: 5 [44800/60000 (75%)]      Loss: 1.039184
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.958342
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.812257

Test set: Average loss: 0.8994, Accuracy: 43665/60000 (73%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 0.906911
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.915169
Train Epoch: 6 [12800/60000 (21%)]      Loss: 0.693767
Train Epoch: 6 [19200/60000 (32%)]      Loss: 0.873584
Train Epoch: 6 [25600/60000 (43%)]      Loss: 0.682414
Train Epoch: 6 [32000/60000 (53%)]      Loss: 0.824000
Train Epoch: 6 [38400/60000 (64%)]      Loss: 0.871938
Train Epoch: 6 [44800/60000 (75%)]      Loss: 1.000326
Train Epoch: 6 [51200/60000 (85%)]      Loss: 0.912071
Train Epoch: 6 [57600/60000 (96%)]      Loss: 0.778144

Test set: Average loss: 0.8642, Accuracy: 44488/60000 (74%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.860060
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.879856
Train Epoch: 7 [12800/60000 (21%)]      Loss: 0.652376
Train Epoch: 7 [19200/60000 (32%)]      Loss: 0.845697
Train Epoch: 7 [25600/60000 (43%)]      Loss: 0.649507
Train Epoch: 7 [32000/60000 (53%)]      Loss: 0.799385
Train Epoch: 7 [38400/60000 (64%)]      Loss: 0.842351
Train Epoch: 7 [44800/60000 (75%)]      Loss: 0.973335
Train Epoch: 7 [51200/60000 (85%)]      Loss: 0.877140
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.755018

Test set: Average loss: 0.8387, Accuracy: 45109/60000 (75%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.824947
Train Epoch: 8 [6400/60000 (11%)]       Loss: 0.855702
Train Epoch: 8 [12800/60000 (21%)]      Loss: 0.621496
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.825064
Train Epoch: 8 [25600/60000 (43%)]      Loss: 0.626086
Train Epoch: 8 [32000/60000 (53%)]      Loss: 0.782028
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.820530
Train Epoch: 8 [44800/60000 (75%)]      Loss: 0.953454
Train Epoch: 8 [51200/60000 (85%)]      Loss: 0.848978
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.738381

Test set: Average loss: 0.8190, Accuracy: 45576/60000 (76%)
```

```
Train Epoch: 9 [0/60000 (0%)]   Loss: 0.797276
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.837142
Train Epoch: 9 [12800/60000 (21%)]      Loss: 0.596835
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.808428
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.608451
Train Epoch: 9 [32000/60000 (53%)]      Loss: 0.768819
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.803843
Train Epoch: 9 [44800/60000 (75%)]      Loss: 0.937671
Train Epoch: 9 [51200/60000 (85%)]      Loss: 0.825218
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.726467

Test set: Average loss: 0.8032, Accuracy: 45973/60000 (77%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.775361
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.822661
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.576997
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.794504
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.594659
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.758383
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.790371
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.924863
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.805281
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.717186

Test set: Average loss: 0.7902, Accuracy: 46302/60000 (77%)
```

Test the trained 50-network on the test data. Print out the accuracy.

[46]:
```
test(model_50, test_criterion, test_loader)
```

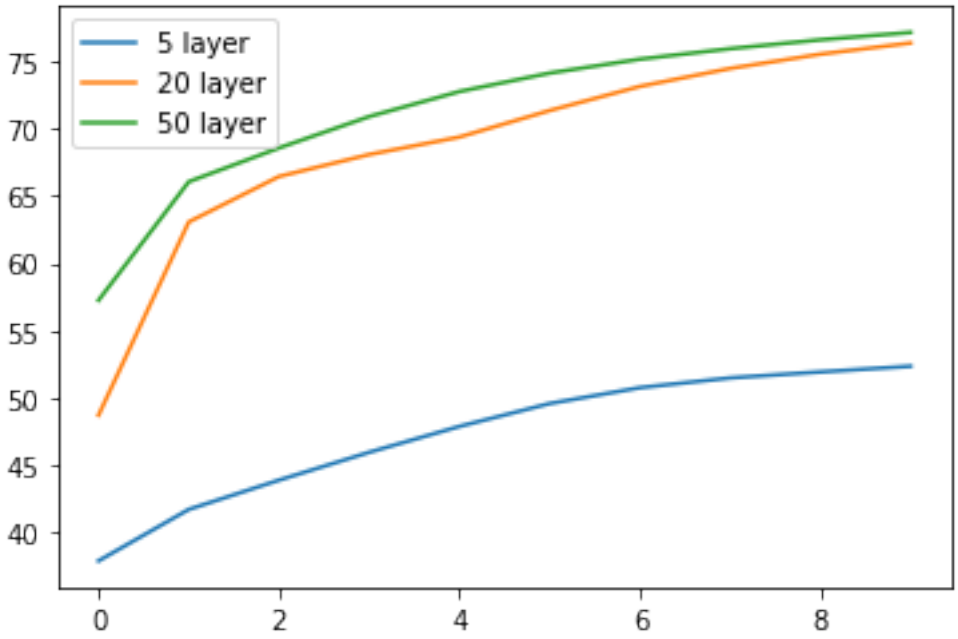
```
Test set: Average loss: 0.7651, Accuracy: 7840/10000 (78%)
```

[46]:
```
78.4
```

Plot the training accuracies over the epochs of the networks on the same figure (there should 3 line plots/scatter plots).

[47]:
```
plt.plot(train_5, label = '5 layer')
plt.plot(train_20, label = '20 layer')
plt.plot(train_50, label = '50 layer')
plt.legend()
```

[47]:
```
<matplotlib.legend.Legend at 0x7fb52c218fa0>
```



What is your conclustion on the effect of varying the hidden layer size on the performance of a neural network trained on the MNIST dataset?

It looks like the greater the layers size we have, the more effective the network becomes after we include more epochs, though the difference is more severe in adding the first few channels more.

1.4 Part b

Now, we will investigate the effect of varying the number of hidden layers. Create 3 networks with 1, 2, and 3 hidden layers, respectively. The size of all hidden layers should be 20 and the activation function is ReLU. We will call these the 1-network, the 2-network, and the 3-network.

[52]:
```
# Number of training epochs
epochs = 10
# Learning rate
lr = 0.001

# Define the training and testing loss
train_criterion = nn.CrossEntropyLoss()
test_criterion = nn.CrossEntropyLoss(reduction='sum')
```

Train the 1-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch,

---

record the current training accuracy of the network.

[53]:
```
model_1 = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')
optimizer = optim.SGD(model_1.parameters(),lr=lr)

train_1 = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_1, train_criterion, train_loader, optimizer, epoch)
        train_1.append(test(model_1, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.351165
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.237006
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.093750
Train Epoch: 1 [19200/60000 (32%)]      Loss: 1.929818
Train Epoch: 1 [25600/60000 (43%)]      Loss: 1.974377
Train Epoch: 1 [32000/60000 (53%)]      Loss: 1.875087
Train Epoch: 1 [38400/60000 (64%)]      Loss: 1.556570
Train Epoch: 1 [44800/60000 (75%)]      Loss: 1.673218
Train Epoch: 1 [51200/60000 (85%)]      Loss: 1.564982
Train Epoch: 1 [57600/60000 (96%)]      Loss: 1.275017

Test set: Average loss: 1.4415, Accuracy: 36236/60000 (60%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 1.484977
Train Epoch: 2 [6400/60000 (11%)]       Loss: 1.304284
Train Epoch: 2 [12800/60000 (21%)]      Loss: 1.314311
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.219861
Train Epoch: 2 [25600/60000 (43%)]      Loss: 1.491711
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.412746
Train Epoch: 2 [38400/60000 (64%)]      Loss: 1.019302
Train Epoch: 2 [44800/60000 (75%)]      Loss: 1.235540
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.194130
Train Epoch: 2 [57600/60000 (96%)]      Loss: 0.936995

Test set: Average loss: 1.1157, Accuracy: 40358/60000 (67%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 1.199095
Train Epoch: 3 [6400/60000 (11%)]       Loss: 0.981038
Train Epoch: 3 [12800/60000 (21%)]      Loss: 1.004695
Train Epoch: 3 [19200/60000 (32%)]      Loss: 0.984115
Train Epoch: 3 [25600/60000 (43%)]      Loss: 1.298311
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.197006
Train Epoch: 3 [38400/60000 (64%)]      Loss: 0.807701
Train Epoch: 3 [44800/60000 (75%)]      Loss: 1.035766
Train Epoch: 3 [51200/60000 (85%)]      Loss: 1.028638
Train Epoch: 3 [57600/60000 (96%)]      Loss: 0.810643

Test set: Average loss: 0.9677, Accuracy: 41877/60000 (70%)
```

```
Train Epoch: 4 [0/60000 (0%)]   Loss: 1.072543
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.834296
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.839809
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.861094
Train Epoch: 4 [25600/60000 (43%)]      Loss: 1.042221
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.997864
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.596406
Train Epoch: 4 [44800/60000 (75%)]      Loss: 0.840835
Train Epoch: 4 [51200/60000 (85%)]      Loss: 0.862188
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.621421

Test set: Average loss: 0.7775, Accuracy: 47279/60000 (79%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.874938
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.700077
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.613795
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.705972
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.930886
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.914301
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.497558
Train Epoch: 5 [44800/60000 (75%)]      Loss: 0.738562
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.768493
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.564168

Test set: Average loss: 0.7090, Accuracy: 47976/60000 (80%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 0.796533
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.618517
Train Epoch: 6 [12800/60000 (21%)]      Loss: 0.556085
Train Epoch: 6 [19200/60000 (32%)]      Loss: 0.639029
Train Epoch: 6 [25600/60000 (43%)]      Loss: 0.880295
Train Epoch: 6 [32000/60000 (53%)]      Loss: 0.858960
Train Epoch: 6 [38400/60000 (64%)]      Loss: 0.450602
Train Epoch: 6 [44800/60000 (75%)]      Loss: 0.685937
Train Epoch: 6 [51200/60000 (85%)]      Loss: 0.709698
Train Epoch: 6 [57600/60000 (96%)]      Loss: 0.533448

Test set: Average loss: 0.6645, Accuracy: 48341/60000 (81%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.745914
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.567364
Train Epoch: 7 [12800/60000 (21%)]      Loss: 0.519716
Train Epoch: 7 [19200/60000 (32%)]      Loss: 0.597483
Train Epoch: 7 [25600/60000 (43%)]      Loss: 0.838515
Train Epoch: 7 [32000/60000 (53%)]      Loss: 0.818532
Train Epoch: 7 [38400/60000 (64%)]      Loss: 0.419259
Train Epoch: 7 [44800/60000 (75%)]      Loss: 0.652005
```

```
Train Epoch: 7 [51200/60000 (85%)]      Loss: 0.670342
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.512509

Test set: Average loss: 0.6324, Accuracy: 48574/60000 (81%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.708307
Train Epoch: 8 [6400/60000 (11%)]       Loss: 0.532710
Train Epoch: 8 [12800/60000 (21%)]      Loss: 0.493294
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.571182
Train Epoch: 8 [25600/60000 (43%)]      Loss: 0.806093
Train Epoch: 8 [32000/60000 (53%)]      Loss: 0.789693
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.395340
Train Epoch: 8 [44800/60000 (75%)]      Loss: 0.627926
Train Epoch: 8 [51200/60000 (85%)]      Loss: 0.641709
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.496980

Test set: Average loss: 0.6083, Accuracy: 48752/60000 (81%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.679113
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.508260
Train Epoch: 9 [12800/60000 (21%)]      Loss: 0.473366
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.553933
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.780387
Train Epoch: 9 [32000/60000 (53%)]      Loss: 0.768840
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.376863
Train Epoch: 9 [44800/60000 (75%)]      Loss: 0.610905
Train Epoch: 9 [51200/60000 (85%)]      Loss: 0.619284
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.485507

Test set: Average loss: 0.5895, Accuracy: 48896/60000 (81%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.654934
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.490679
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.458022
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.541221
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.760644
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.752690
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.362441
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.598285
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.601211
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.475790

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)
```

Test the trained 1-network on the test data. Print out the accuracy.

[54]:
```
test(model_1, test_criterion, test_loader)
```

```
Test set: Average loss: 0.5615, Accuracy: 8222/10000 (82%)
```

[54]:
```
82.22
```

Train the 2-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[55]:
```
model_2 = mnist_network(num_hidden_layers=2,layer_size=20,activation='relu')
optimizer = optim.SGD(model_2.parameters(),lr=lr)

train_2 = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_2, train_criterion, train_loader, optimizer, epoch)
        train_2.append(test(model_1, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.325221
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.305492
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.301848
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.260953
Train Epoch: 1 [25600/60000 (43%)]      Loss: 2.275739
Train Epoch: 1 [32000/60000 (53%)]      Loss: 2.241323
Train Epoch: 1 [38400/60000 (64%)]      Loss: 2.214609
Train Epoch: 1 [44800/60000 (75%)]      Loss: 2.261936
Train Epoch: 1 [51200/60000 (85%)]      Loss: 2.209897
Train Epoch: 1 [57600/60000 (96%)]      Loss: 2.157484

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 2.153461
Train Epoch: 2 [6400/60000 (11%)]       Loss: 2.113545
Train Epoch: 2 [12800/60000 (21%)]      Loss: 2.206012
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.969792
Train Epoch: 2 [25600/60000 (43%)]      Loss: 2.003573
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.958805
Train Epoch: 2 [38400/60000 (64%)]      Loss: 1.955050
Train Epoch: 2 [44800/60000 (75%)]      Loss: 2.122275
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.954876
Train Epoch: 2 [57600/60000 (96%)]      Loss: 1.888553

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 1.820627
Train Epoch: 3 [6400/60000 (11%)]       Loss: 1.759484
Train Epoch: 3 [12800/60000 (21%)]      Loss: 1.991349
Train Epoch: 3 [19200/60000 (32%)]      Loss: 1.668006
```

```
Train Epoch: 3 [25600/60000 (43%)]      Loss: 1.688461
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.643827
Train Epoch: 3 [38400/60000 (64%)]      Loss: 1.612760
Train Epoch: 3 [44800/60000 (75%)]      Loss: 1.944686
Train Epoch: 3 [51200/60000 (85%)]      Loss: 1.683515
Train Epoch: 3 [57600/60000 (96%)]      Loss: 1.552363

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 1.531962
Train Epoch: 4 [6400/60000 (11%)]       Loss: 1.490770
Train Epoch: 4 [12800/60000 (21%)]      Loss: 1.727600
Train Epoch: 4 [19200/60000 (32%)]      Loss: 1.419731
Train Epoch: 4 [25600/60000 (43%)]      Loss: 1.418502
Train Epoch: 4 [32000/60000 (53%)]      Loss: 1.418193
Train Epoch: 4 [38400/60000 (64%)]      Loss: 1.333709
Train Epoch: 4 [44800/60000 (75%)]      Loss: 1.755605
Train Epoch: 4 [51200/60000 (85%)]      Loss: 1.451314
Train Epoch: 4 [57600/60000 (96%)]      Loss: 1.286769

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 1.314415
Train Epoch: 5 [6400/60000 (11%)]       Loss: 1.280101
Train Epoch: 5 [12800/60000 (21%)]      Loss: 1.512478
Train Epoch: 5 [19200/60000 (32%)]      Loss: 1.223100
Train Epoch: 5 [25600/60000 (43%)]      Loss: 1.227224
Train Epoch: 5 [32000/60000 (53%)]      Loss: 1.251781
Train Epoch: 5 [38400/60000 (64%)]      Loss: 1.156390
Train Epoch: 5 [44800/60000 (75%)]      Loss: 1.611154
Train Epoch: 5 [51200/60000 (85%)]      Loss: 1.304353
Train Epoch: 5 [57600/60000 (96%)]      Loss: 1.128117

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 1.171854
Train Epoch: 6 [6400/60000 (11%)]       Loss: 1.143544
Train Epoch: 6 [12800/60000 (21%)]      Loss: 1.372679
Train Epoch: 6 [19200/60000 (32%)]      Loss: 1.099954
Train Epoch: 6 [25600/60000 (43%)]      Loss: 1.122496
Train Epoch: 6 [32000/60000 (53%)]      Loss: 1.152708
Train Epoch: 6 [38400/60000 (64%)]      Loss: 1.057380
Train Epoch: 6 [44800/60000 (75%)]      Loss: 1.519413
Train Epoch: 6 [51200/60000 (85%)]      Loss: 1.212132
Train Epoch: 6 [57600/60000 (96%)]      Loss: 1.041719

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)
```

```
Train Epoch: 7 [0/60000 (0%)]   Loss: 1.075606
Train Epoch: 7 [6400/60000 (11%)]       Loss: 1.063624
Train Epoch: 7 [12800/60000 (21%)]      Loss: 1.289917
Train Epoch: 7 [19200/60000 (32%)]      Loss: 1.026551
Train Epoch: 7 [25600/60000 (43%)]      Loss: 1.052520
Train Epoch: 7 [32000/60000 (53%)]      Loss: 1.094956
Train Epoch: 7 [38400/60000 (64%)]      Loss: 1.000678
Train Epoch: 7 [44800/60000 (75%)]      Loss: 1.456095
Train Epoch: 7 [51200/60000 (85%)]      Loss: 1.143332
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.989034

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 1.006169
Train Epoch: 8 [6400/60000 (11%)]       Loss: 1.014295
Train Epoch: 8 [12800/60000 (21%)]      Loss: 1.236329
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.978822
Train Epoch: 8 [25600/60000 (43%)]      Loss: 1.003454
Train Epoch: 8 [32000/60000 (53%)]      Loss: 1.061629
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.965616
Train Epoch: 8 [44800/60000 (75%)]      Loss: 1.410911
Train Epoch: 8 [51200/60000 (85%)]      Loss: 1.090428
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.952378

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.950642
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.979889
Train Epoch: 9 [12800/60000 (21%)]      Loss: 1.197840
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.949107
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.966146
Train Epoch: 9 [32000/60000 (53%)]      Loss: 1.038035
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.942617
Train Epoch: 9 [44800/60000 (75%)]      Loss: 1.378685
Train Epoch: 9 [51200/60000 (85%)]      Loss: 1.048139
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.921625

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.906483
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.953474
Train Epoch: 10 [12800/60000 (21%)]     Loss: 1.168054
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.928759
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.937555
Train Epoch: 10 [32000/60000 (53%)]     Loss: 1.021101
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.924393
Train Epoch: 10 [44800/60000 (75%)]     Loss: 1.354418
Train Epoch: 10 [51200/60000 (85%)]     Loss: 1.015701
```

```
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.897101

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)
```

Test the trained 2-network on the test data. Print out the accuracy.

[56]:
```
test(model_2, test_criterion, test_loader)
```

```
Test set: Average loss: 1.0217, Accuracy: 6507/10000 (65%)
```

[56]:
```
65.07
```

Train the 3-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[57]:
```
model_3 = mnist_network(num_hidden_layers=3,layer_size=20,activation='relu')
optimizer = optim.SGD(model_3.parameters(),lr=lr)

train_3 = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_3, train_criterion, train_loader, optimizer, epoch)
        train_3.append(test(model_1, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.302562
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.303644
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.302970
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.314267
Train Epoch: 1 [25600/60000 (43%)]      Loss: 2.308614
Train Epoch: 1 [32000/60000 (53%)]      Loss: 2.293324
Train Epoch: 1 [38400/60000 (64%)]      Loss: 2.290712
Train Epoch: 1 [44800/60000 (75%)]      Loss: 2.297036
Train Epoch: 1 [51200/60000 (85%)]      Loss: 2.299915
Train Epoch: 1 [57600/60000 (96%)]      Loss: 2.299205

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 2.294710
Train Epoch: 2 [6400/60000 (11%)]       Loss: 2.294097
Train Epoch: 2 [12800/60000 (21%)]      Loss: 2.296114
Train Epoch: 2 [19200/60000 (32%)]      Loss: 2.309779
Train Epoch: 2 [25600/60000 (43%)]      Loss: 2.296661
Train Epoch: 2 [32000/60000 (53%)]      Loss: 2.280991
Train Epoch: 2 [38400/60000 (64%)]      Loss: 2.271788
Train Epoch: 2 [44800/60000 (75%)]      Loss: 2.274253
Train Epoch: 2 [51200/60000 (85%)]      Loss: 2.280630
```

```
Train Epoch: 2 [57600/60000 (96%)]      Loss: 2.274354

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 2.258916
Train Epoch: 3 [6400/60000 (11%)]       Loss: 2.260136
Train Epoch: 3 [12800/60000 (21%)]      Loss: 2.279160
Train Epoch: 3 [19200/60000 (32%)]      Loss: 2.270591
Train Epoch: 3 [25600/60000 (43%)]      Loss: 2.271703
Train Epoch: 3 [32000/60000 (53%)]      Loss: 2.248841
Train Epoch: 3 [38400/60000 (64%)]      Loss: 2.221462
Train Epoch: 3 [44800/60000 (75%)]      Loss: 2.222509
Train Epoch: 3 [51200/60000 (85%)]      Loss: 2.233599
Train Epoch: 3 [57600/60000 (96%)]      Loss: 2.223000

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 2.188464
Train Epoch: 4 [6400/60000 (11%)]       Loss: 2.190552
Train Epoch: 4 [12800/60000 (21%)]      Loss: 2.249191
Train Epoch: 4 [19200/60000 (32%)]      Loss: 2.188062
Train Epoch: 4 [25600/60000 (43%)]      Loss: 2.227818
Train Epoch: 4 [32000/60000 (53%)]      Loss: 2.188554
Train Epoch: 4 [38400/60000 (64%)]      Loss: 2.112998
Train Epoch: 4 [44800/60000 (75%)]      Loss: 2.122437
Train Epoch: 4 [51200/60000 (85%)]      Loss: 2.144279
Train Epoch: 4 [57600/60000 (96%)]      Loss: 2.126651

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 2.067403
Train Epoch: 5 [6400/60000 (11%)]       Loss: 2.069472
Train Epoch: 5 [12800/60000 (21%)]      Loss: 2.208099
Train Epoch: 5 [19200/60000 (32%)]      Loss: 2.095124
Train Epoch: 5 [25600/60000 (43%)]      Loss: 2.169632
Train Epoch: 5 [32000/60000 (53%)]      Loss: 2.114424
Train Epoch: 5 [38400/60000 (64%)]      Loss: 1.997360
Train Epoch: 5 [44800/60000 (75%)]      Loss: 2.018411
Train Epoch: 5 [51200/60000 (85%)]      Loss: 2.065549
Train Epoch: 5 [57600/60000 (96%)]      Loss: 2.056535

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 1.979900
Train Epoch: 6 [6400/60000 (11%)]       Loss: 1.965487
Train Epoch: 6 [12800/60000 (21%)]      Loss: 2.150515
Train Epoch: 6 [19200/60000 (32%)]      Loss: 2.040139
Train Epoch: 6 [25600/60000 (43%)]      Loss: 2.105255
```

```
Train Epoch: 6 [32000/60000 (53%)]      Loss: 2.031651
Train Epoch: 6 [38400/60000 (64%)]      Loss: 1.899890
Train Epoch: 6 [44800/60000 (75%)]      Loss: 1.911468
Train Epoch: 6 [51200/60000 (85%)]      Loss: 1.982568
Train Epoch: 6 [57600/60000 (96%)]      Loss: 1.994272

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 1.886821
Train Epoch: 7 [6400/60000 (11%)]       Loss: 1.843724
Train Epoch: 7 [12800/60000 (21%)]      Loss: 2.060527
Train Epoch: 7 [19200/60000 (32%)]      Loss: 1.976303
Train Epoch: 7 [25600/60000 (43%)]      Loss: 2.010366
Train Epoch: 7 [32000/60000 (53%)]      Loss: 1.918582
Train Epoch: 7 [38400/60000 (64%)]      Loss: 1.754093
Train Epoch: 7 [44800/60000 (75%)]      Loss: 1.763331
Train Epoch: 7 [51200/60000 (85%)]      Loss: 1.848082
Train Epoch: 7 [57600/60000 (96%)]      Loss: 1.886458

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 1.722288
Train Epoch: 8 [6400/60000 (11%)]       Loss: 1.634319
Train Epoch: 8 [12800/60000 (21%)]      Loss: 1.916767
Train Epoch: 8 [19200/60000 (32%)]      Loss: 1.803691
Train Epoch: 8 [25600/60000 (43%)]      Loss: 1.836590
Train Epoch: 8 [32000/60000 (53%)]      Loss: 1.719116
Train Epoch: 8 [38400/60000 (64%)]      Loss: 1.522881
Train Epoch: 8 [44800/60000 (75%)]      Loss: 1.605103
Train Epoch: 8 [51200/60000 (85%)]      Loss: 1.621219
Train Epoch: 8 [57600/60000 (96%)]      Loss: 1.699002

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 1.511993
Train Epoch: 9 [6400/60000 (11%)]       Loss: 1.376002
Train Epoch: 9 [12800/60000 (21%)]      Loss: 1.673275
Train Epoch: 9 [19200/60000 (32%)]      Loss: 1.577297
Train Epoch: 9 [25600/60000 (43%)]      Loss: 1.609170
Train Epoch: 9 [32000/60000 (53%)]      Loss: 1.463794
Train Epoch: 9 [38400/60000 (64%)]      Loss: 1.249712
Train Epoch: 9 [44800/60000 (75%)]      Loss: 1.425193
Train Epoch: 9 [51200/60000 (85%)]      Loss: 1.391497
Train Epoch: 9 [57600/60000 (96%)]      Loss: 1.343554

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 1.301243
```

```
Train Epoch: 10 [6400/60000 (11%)]      Loss: 1.156901
Train Epoch: 10 [12800/60000 (21%)]     Loss: 1.378169
Train Epoch: 10 [19200/60000 (32%)]     Loss: 1.381629
Train Epoch: 10 [25600/60000 (43%)]     Loss: 1.437446
Train Epoch: 10 [32000/60000 (53%)]     Loss: 1.292783
Train Epoch: 10 [38400/60000 (64%)]     Loss: 1.017623
Train Epoch: 10 [44800/60000 (75%)]     Loss: 1.284842
Train Epoch: 10 [51200/60000 (85%)]     Loss: 1.243222
Train Epoch: 10 [57600/60000 (96%)]     Loss: 1.135275

Test set: Average loss: 0.5746, Accuracy: 49030/60000 (82%)
```

Test the trained 3-network on the test data. Print out the accuracy.

[58]:
```
test(model_3, test_criterion, test_loader)
```

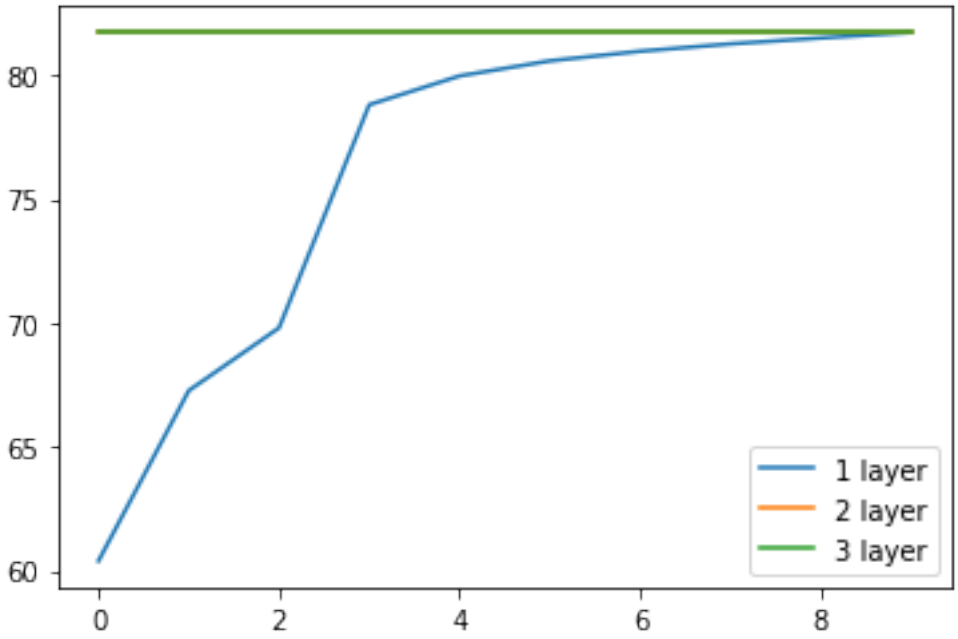
```
Test set: Average loss: 1.2027, Accuracy: 6328/10000 (63%)
```

[58]:
```
63.28
```

Plot the training accuracies over the epochs of the networks on the same figure (there should 3 line plots/scatter plots).

[59]:
```
plt.plot(train_1, label = '1 layer')
plt.plot(train_2, label = '2 layer')
plt.plot(train_3, label = '3 layer')
plt.legend()
```

[59]:
```
<matplotlib.legend.Legend at 0x7fb53012fee0>
```



What is your conclustion on the effect of varying the number of hidden layers on the performance of a neural network trained on the MNIST dataset?

My training accuracies for the 2nd and 3rd layer were approximately the same, so adding another hidden layer will train the model faster, but doesn't necessarily make the model more accurate after a certain number of epochs.

1.5 Part c

Next, we will investigate the effects of varying the activation functions on a neural network. Create 3 networks. The first network has Sigmoid activation (Sigmoid-network). The second network has ReLU activation (ReLU-network). The third network has Tanh activation (Tanh-network). All networks have one hidden layer with size 20.

[60]:
```
# Number of training epochs
epochs = 10
# Learning rate
lr = 0.001

# Define the training and testing loss
train_criterion = nn.CrossEntropyLoss()
test_criterion = nn.CrossEntropyLoss(reduction='sum')
```

Train the Sigmoid-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[61]:
```
model_s = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')
model_s.activation = torch.nn.Sigmoid()
optimizer = optim.SGD(model_s.parameters(),lr=lr)

train_s = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_s, train_criterion, train_loader, optimizer, epoch)
        train_s.append(test(model_s, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.301595
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.309707
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.303554
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.304775
Train Epoch: 1 [25600/60000 (43%)]      Loss: 2.297797
Train Epoch: 1 [32000/60000 (53%)]      Loss: 2.294732
Train Epoch: 1 [38400/60000 (64%)]      Loss: 2.285025
Train Epoch: 1 [44800/60000 (75%)]      Loss: 2.315665
Train Epoch: 1 [51200/60000 (85%)]      Loss: 2.292294
Train Epoch: 1 [57600/60000 (96%)]      Loss: 2.287698

Test set: Average loss: 2.2899, Accuracy: 5922/60000 (10%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 2.287932
Train Epoch: 2 [6400/60000 (11%)]       Loss: 2.295752
Train Epoch: 2 [12800/60000 (21%)]      Loss: 2.292507
Train Epoch: 2 [19200/60000 (32%)]      Loss: 2.288934
Train Epoch: 2 [25600/60000 (43%)]      Loss: 2.284253
Train Epoch: 2 [32000/60000 (53%)]      Loss: 2.282168
Train Epoch: 2 [38400/60000 (64%)]      Loss: 2.270616
Train Epoch: 2 [44800/60000 (75%)]      Loss: 2.304687
Train Epoch: 2 [51200/60000 (85%)]      Loss: 2.279253
Train Epoch: 2 [57600/60000 (96%)]      Loss: 2.274067

Test set: Average loss: 2.2768, Accuracy: 8358/60000 (14%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 2.274616
Train Epoch: 3 [6400/60000 (11%)]       Loss: 2.281997
Train Epoch: 3 [12800/60000 (21%)]      Loss: 2.281560
Train Epoch: 3 [19200/60000 (32%)]      Loss: 2.273484
Train Epoch: 3 [25600/60000 (43%)]      Loss: 2.271089
Train Epoch: 3 [32000/60000 (53%)]      Loss: 2.269976
Train Epoch: 3 [38400/60000 (64%)]      Loss: 2.256835
Train Epoch: 3 [44800/60000 (75%)]      Loss: 2.294069
Train Epoch: 3 [51200/60000 (85%)]      Loss: 2.266760
Train Epoch: 3 [57600/60000 (96%)]      Loss: 2.261118
```

```
Test set: Average loss: 2.2642, Accuracy: 11930/60000 (20%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 2.262172
Train Epoch: 4 [6400/60000 (11%)]       Loss: 2.268850
Train Epoch: 4 [12800/60000 (21%)]      Loss: 2.270884
Train Epoch: 4 [19200/60000 (32%)]      Loss: 2.259145
Train Epoch: 4 [25600/60000 (43%)]      Loss: 2.258778
Train Epoch: 4 [32000/60000 (53%)]      Loss: 2.258431
Train Epoch: 4 [38400/60000 (64%)]      Loss: 2.243957
Train Epoch: 4 [44800/60000 (75%)]      Loss: 2.284070
Train Epoch: 4 [51200/60000 (85%)]      Loss: 2.255064
Train Epoch: 4 [57600/60000 (96%)]      Loss: 2.249102

Test set: Average loss: 2.2525, Accuracy: 14705/60000 (25%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 2.250757
Train Epoch: 5 [6400/60000 (11%)]       Loss: 2.256540
Train Epoch: 5 [12800/60000 (21%)]      Loss: 2.260612
Train Epoch: 5 [19200/60000 (32%)]      Loss: 2.246204
Train Epoch: 5 [25600/60000 (43%)]      Loss: 2.247457
Train Epoch: 5 [32000/60000 (53%)]      Loss: 2.247637
Train Epoch: 5 [38400/60000 (64%)]      Loss: 2.232019
Train Epoch: 5 [44800/60000 (75%)]      Loss: 2.274724
Train Epoch: 5 [51200/60000 (85%)]      Loss: 2.244221
Train Epoch: 5 [57600/60000 (96%)]      Loss: 2.238004

Test set: Average loss: 2.2417, Accuracy: 18829/60000 (31%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 2.240258
Train Epoch: 6 [6400/60000 (11%)]       Loss: 2.245085
Train Epoch: 6 [12800/60000 (21%)]      Loss: 2.250801
Train Epoch: 6 [19200/60000 (32%)]      Loss: 2.234568
Train Epoch: 6 [25600/60000 (43%)]      Loss: 2.237023
Train Epoch: 6 [32000/60000 (53%)]      Loss: 2.237560
Train Epoch: 6 [38400/60000 (64%)]      Loss: 2.220931
Train Epoch: 6 [44800/60000 (75%)]      Loss: 2.265945
Train Epoch: 6 [51200/60000 (85%)]      Loss: 2.234161
Train Epoch: 6 [57600/60000 (96%)]      Loss: 2.227687

Test set: Average loss: 2.2316, Accuracy: 23221/60000 (39%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 2.230493
Train Epoch: 7 [6400/60000 (11%)]       Loss: 2.234391
Train Epoch: 7 [12800/60000 (21%)]      Loss: 2.241434
Train Epoch: 7 [19200/60000 (32%)]      Loss: 2.223974
Train Epoch: 7 [25600/60000 (43%)]      Loss: 2.227299
Train Epoch: 7 [32000/60000 (53%)]      Loss: 2.228103
Train Epoch: 7 [38400/60000 (64%)]      Loss: 2.210568
```

```
Train Epoch: 7 [44800/60000 (75%)]      Loss: 2.257609
Train Epoch: 7 [51200/60000 (85%)]      Loss: 2.224752
Train Epoch: 7 [57600/60000 (96%)]      Loss: 2.217993

Test set: Average loss: 2.2221, Accuracy: 26780/60000 (45%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 2.221304
Train Epoch: 8 [6400/60000 (11%)]       Loss: 2.224339
Train Epoch: 8 [12800/60000 (21%)]      Loss: 2.232463
Train Epoch: 8 [19200/60000 (32%)]      Loss: 2.214160
Train Epoch: 8 [25600/60000 (43%)]      Loss: 2.218123
Train Epoch: 8 [32000/60000 (53%)]      Loss: 2.219155
Train Epoch: 8 [38400/60000 (64%)]      Loss: 2.200808
Train Epoch: 8 [44800/60000 (75%)]      Loss: 2.249600
Train Epoch: 8 [51200/60000 (85%)]      Loss: 2.215869
Train Epoch: 8 [57600/60000 (96%)]      Loss: 2.208789

Test set: Average loss: 2.2131, Accuracy: 29391/60000 (49%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 2.212569
Train Epoch: 9 [6400/60000 (11%)]       Loss: 2.214814
Train Epoch: 9 [12800/60000 (21%)]      Loss: 2.223830
Train Epoch: 9 [19200/60000 (32%)]      Loss: 2.204919
Train Epoch: 9 [25600/60000 (43%)]      Loss: 2.209368
Train Epoch: 9 [32000/60000 (53%)]      Loss: 2.210621
Train Epoch: 9 [38400/60000 (64%)]      Loss: 2.191550
Train Epoch: 9 [44800/60000 (75%)]      Loss: 2.241825
Train Epoch: 9 [51200/60000 (85%)]      Loss: 2.207404
Train Epoch: 9 [57600/60000 (96%)]      Loss: 2.199971

Test set: Average loss: 2.2045, Accuracy: 31297/60000 (52%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 2.204198
Train Epoch: 10 [6400/60000 (11%)]      Loss: 2.205724
Train Epoch: 10 [12800/60000 (21%)]     Loss: 2.215479
Train Epoch: 10 [19200/60000 (32%)]     Loss: 2.196106
Train Epoch: 10 [25600/60000 (43%)]     Loss: 2.200945
Train Epoch: 10 [32000/60000 (53%)]     Loss: 2.202423
Train Epoch: 10 [38400/60000 (64%)]     Loss: 2.182712
Train Epoch: 10 [44800/60000 (75%)]     Loss: 2.234219
Train Epoch: 10 [51200/60000 (85%)]     Loss: 2.199275
Train Epoch: 10 [57600/60000 (96%)]     Loss: 2.191464

Test set: Average loss: 2.1961, Accuracy: 32870/60000 (55%)
```

Test the trained Sigmoid-network on the test data. Print out the accuracy.

[62]:
```
test(model_s, test_criterion, test_loader)
```

```
Test set: Average loss: 2.1945, Accuracy: 5561/10000 (56%)
```

[62]:
```
55.61
```

Train the ReLU-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[63]:
```
model_r = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')
optimizer = optim.SGD(model_r.parameters(),lr=lr)

train_r = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_r, train_criterion, train_loader, optimizer, epoch)
        train_r.append(test(model_r, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.281887
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.217042
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.294579
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.148053
Train Epoch: 1 [25600/60000 (43%)]      Loss: 2.133491
Train Epoch: 1 [32000/60000 (53%)]      Loss: 2.118838
Train Epoch: 1 [38400/60000 (64%)]      Loss: 2.027254
Train Epoch: 1 [44800/60000 (75%)]      Loss: 2.099699
Train Epoch: 1 [51200/60000 (85%)]      Loss: 1.917426
Train Epoch: 1 [57600/60000 (96%)]      Loss: 1.996129

Test set: Average loss: 1.8640, Accuracy: 24642/60000 (41%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 1.756914
Train Epoch: 2 [6400/60000 (11%)]       Loss: 1.699863
Train Epoch: 2 [12800/60000 (21%)]      Loss: 1.960794
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.649650
Train Epoch: 2 [25600/60000 (43%)]      Loss: 1.533370
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.644837
Train Epoch: 2 [38400/60000 (64%)]      Loss: 1.634581
Train Epoch: 2 [44800/60000 (75%)]      Loss: 1.531609
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.450205
Train Epoch: 2 [57600/60000 (96%)]      Loss: 1.526274

Test set: Average loss: 1.3516, Accuracy: 37826/60000 (63%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 1.316581
Train Epoch: 3 [6400/60000 (11%)]       Loss: 1.239222
```

```
Train Epoch: 3 [12800/60000 (21%)]      Loss: 1.241424
Train Epoch: 3 [19200/60000 (32%)]      Loss: 1.120924
Train Epoch: 3 [25600/60000 (43%)]      Loss: 0.940467
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.094005
Train Epoch: 3 [38400/60000 (64%)]      Loss: 1.064625
Train Epoch: 3 [44800/60000 (75%)]      Loss: 1.016214
Train Epoch: 3 [51200/60000 (85%)]      Loss: 0.966735
Train Epoch: 3 [57600/60000 (96%)]      Loss: 1.014855

Test set: Average loss: 0.9268, Accuracy: 45260/60000 (75%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 0.936652
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.924614
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.890812
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.777797
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.692548
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.866574
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.875229
Train Epoch: 4 [44800/60000 (75%)]      Loss: 0.816575
Train Epoch: 4 [51200/60000 (85%)]      Loss: 0.802185
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.883845

Test set: Average loss: 0.7927, Accuracy: 46121/60000 (77%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.797789
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.807727
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.766917
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.682856
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.597885
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.769941
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.796269
Train Epoch: 5 [44800/60000 (75%)]      Loss: 0.735836
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.723555
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.824592

Test set: Average loss: 0.7276, Accuracy: 46612/60000 (78%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 0.723001
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.749109
Train Epoch: 6 [12800/60000 (21%)]      Loss: 0.704244
Train Epoch: 6 [19200/60000 (32%)]      Loss: 0.633270
Train Epoch: 6 [25600/60000 (43%)]      Loss: 0.544363
Train Epoch: 6 [32000/60000 (53%)]      Loss: 0.718154
Train Epoch: 6 [38400/60000 (64%)]      Loss: 0.751379
Train Epoch: 6 [44800/60000 (75%)]      Loss: 0.692432
Train Epoch: 6 [51200/60000 (85%)]      Loss: 0.676200
Train Epoch: 6 [57600/60000 (96%)]      Loss: 0.791701
```

```
Test set: Average loss: 0.6883, Accuracy: 46962/60000 (78%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.674238
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.713985
Train Epoch: 7 [12800/60000 (21%)]      Loss: 0.664369
Train Epoch: 7 [19200/60000 (32%)]      Loss: 0.601350
Train Epoch: 7 [25600/60000 (43%)]      Loss: 0.508732
Train Epoch: 7 [32000/60000 (53%)]      Loss: 0.685534
Train Epoch: 7 [38400/60000 (64%)]      Loss: 0.721416
Train Epoch: 7 [44800/60000 (75%)]      Loss: 0.665319
Train Epoch: 7 [51200/60000 (85%)]      Loss: 0.645631
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.771619

Test set: Average loss: 0.6615, Accuracy: 47194/60000 (79%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.638965
Train Epoch: 8 [6400/60000 (11%)]       Loss: 0.690548
Train Epoch: 8 [12800/60000 (21%)]      Loss: 0.636072
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.578550
Train Epoch: 8 [25600/60000 (43%)]      Loss: 0.482469
Train Epoch: 8 [32000/60000 (53%)]      Loss: 0.662586
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.700130
Train Epoch: 8 [44800/60000 (75%)]      Loss: 0.646281
Train Epoch: 8 [51200/60000 (85%)]      Loss: 0.623952
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.758897

Test set: Average loss: 0.6418, Accuracy: 47407/60000 (79%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.612075
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.674097
Train Epoch: 9 [12800/60000 (21%)]      Loss: 0.614408
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.561181
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.462030
Train Epoch: 9 [32000/60000 (53%)]      Loss: 0.646933
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.683693
Train Epoch: 9 [44800/60000 (75%)]      Loss: 0.632435
Train Epoch: 9 [51200/60000 (85%)]      Loss: 0.607484
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.749298

Test set: Average loss: 0.6264, Accuracy: 47583/60000 (79%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.590189
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.662012
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.597644
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.547133
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.445923
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.633784
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.670642
```

```
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.622273
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.593935
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.742145

Test set: Average loss: 0.6140, Accuracy: 47711/60000 (80%)
```

Test the trained ReLU-network on the test data. Print out the accuracy.

[64]:
```
test(model_r, test_criterion, test_loader)
```

```
Test set: Average loss: 0.6020, Accuracy: 7980/10000 (80%)
```

[64]:
```
79.8
```

Train the Tanh-network on the MNIST dataset for 10 epochs with learning rate 0.001. After each epoch, record the current training accuracy of the network.

[65]:
```
model_t = mnist_network(num_hidden_layers=1,layer_size=20,activation='tanh')
optimizer = optim.SGD(model_t.parameters(),lr=lr)

train_t = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_t, train_criterion, train_loader, optimizer, epoch)
        train_t.append(test(model_t, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.339976
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.083477
Train Epoch: 1 [12800/60000 (21%)]      Loss: 1.929702
Train Epoch: 1 [19200/60000 (32%)]      Loss: 1.899072
Train Epoch: 1 [25600/60000 (43%)]      Loss: 1.778455
Train Epoch: 1 [32000/60000 (53%)]      Loss: 1.760510
Train Epoch: 1 [38400/60000 (64%)]      Loss: 1.704436
Train Epoch: 1 [44800/60000 (75%)]      Loss: 1.745781
Train Epoch: 1 [51200/60000 (85%)]      Loss: 1.733356
Train Epoch: 1 [57600/60000 (96%)]      Loss: 1.657083

Test set: Average loss: 1.6586, Accuracy: 43411/60000 (72%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 1.653630
Train Epoch: 2 [6400/60000 (11%)]       Loss: 1.630653
Train Epoch: 2 [12800/60000 (21%)]      Loss: 1.619556
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.623303
Train Epoch: 2 [25600/60000 (43%)]      Loss: 1.578895
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.604777
Train Epoch: 2 [38400/60000 (64%)]      Loss: 1.546601
```

```
Train Epoch: 2 [44800/60000 (75%)]      Loss: 1.602877
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.608471
Train Epoch: 2 [57600/60000 (96%)]      Loss: 1.542407

Test set: Average loss: 1.5460, Accuracy: 47812/60000 (80%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 1.541033
Train Epoch: 3 [6400/60000 (11%)]       Loss: 1.518264
Train Epoch: 3 [12800/60000 (21%)]      Loss: 1.525522
Train Epoch: 3 [19200/60000 (32%)]      Loss: 1.526270
Train Epoch: 3 [25600/60000 (43%)]      Loss: 1.500430
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.529891
Train Epoch: 3 [38400/60000 (64%)]      Loss: 1.465365
Train Epoch: 3 [44800/60000 (75%)]      Loss: 1.530134
Train Epoch: 3 [51200/60000 (85%)]      Loss: 1.534062
Train Epoch: 3 [57600/60000 (96%)]      Loss: 1.474018

Test set: Average loss: 1.4782, Accuracy: 49686/60000 (83%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 1.471897
Train Epoch: 4 [6400/60000 (11%)]       Loss: 1.448209
Train Epoch: 4 [12800/60000 (21%)]      Loss: 1.465878
Train Epoch: 4 [19200/60000 (32%)]      Loss: 1.464414
Train Epoch: 4 [25600/60000 (43%)]      Loss: 1.446960
Train Epoch: 4 [32000/60000 (53%)]      Loss: 1.474476
Train Epoch: 4 [38400/60000 (64%)]      Loss: 1.407470
Train Epoch: 4 [44800/60000 (75%)]      Loss: 1.478619
Train Epoch: 4 [51200/60000 (85%)]      Loss: 1.479325
Train Epoch: 4 [57600/60000 (96%)]      Loss: 1.422948

Test set: Average loss: 1.4276, Accuracy: 50649/60000 (84%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 1.419244
Train Epoch: 5 [6400/60000 (11%)]       Loss: 1.396086
Train Epoch: 5 [12800/60000 (21%)]      Loss: 1.419674
Train Epoch: 5 [19200/60000 (32%)]      Loss: 1.418471
Train Epoch: 5 [25600/60000 (43%)]      Loss: 1.403967
Train Epoch: 5 [32000/60000 (53%)]      Loss: 1.428925
Train Epoch: 5 [38400/60000 (64%)]      Loss: 1.361804
Train Epoch: 5 [44800/60000 (75%)]      Loss: 1.437657
Train Epoch: 5 [51200/60000 (85%)]      Loss: 1.435258
Train Epoch: 5 [57600/60000 (96%)]      Loss: 1.381601

Test set: Average loss: 1.3863, Accuracy: 51214/60000 (85%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 1.375681
Train Epoch: 6 [6400/60000 (11%)]       Loss: 1.353973
Train Epoch: 6 [12800/60000 (21%)]      Loss: 1.380950
```

```
Train Epoch: 6 [19200/60000 (32%)]      Loss: 1.381694
Train Epoch: 6 [25600/60000 (43%)]      Loss: 1.367401
Train Epoch: 6 [32000/60000 (53%)]      Loss: 1.390481
Train Epoch: 6 [38400/60000 (64%)]      Loss: 1.324276
Train Epoch: 6 [44800/60000 (75%)]      Loss: 1.403409
Train Epoch: 6 [51200/60000 (85%)]      Loss: 1.397951
Train Epoch: 6 [57600/60000 (96%)]      Loss: 1.346887

Test set: Average loss: 1.3515, Accuracy: 51648/60000 (86%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 1.338529
Train Epoch: 7 [6400/60000 (11%)]       Loss: 1.318748
Train Epoch: 7 [12800/60000 (21%)]      Loss: 1.347319
Train Epoch: 7 [19200/60000 (32%)]      Loss: 1.351182
Train Epoch: 7 [25600/60000 (43%)]      Loss: 1.335975
Train Epoch: 7 [32000/60000 (53%)]      Loss: 1.358059
Train Epoch: 7 [38400/60000 (64%)]      Loss: 1.292866
Train Epoch: 7 [44800/60000 (75%)]      Loss: 1.374063
Train Epoch: 7 [51200/60000 (85%)]      Loss: 1.365597
Train Epoch: 7 [57600/60000 (96%)]      Loss: 1.317237

Test set: Average loss: 1.3215, Accuracy: 51971/60000 (87%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 1.306477
Train Epoch: 8 [6400/60000 (11%)]       Loss: 1.288790
Train Epoch: 8 [12800/60000 (21%)]      Loss: 1.317505
Train Epoch: 8 [19200/60000 (32%)]      Loss: 1.325275
Train Epoch: 8 [25600/60000 (43%)]      Loss: 1.308811
Train Epoch: 8 [32000/60000 (53%)]      Loss: 1.330597
Train Epoch: 8 [38400/60000 (64%)]      Loss: 1.266326
Train Epoch: 8 [44800/60000 (75%)]      Loss: 1.348563
Train Epoch: 8 [51200/60000 (85%)]      Loss: 1.337181
Train Epoch: 8 [57600/60000 (96%)]      Loss: 1.291643

Test set: Average loss: 1.2954, Accuracy: 52252/60000 (87%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 1.278593
Train Epoch: 9 [6400/60000 (11%)]       Loss: 1.263040
Train Epoch: 9 [12800/60000 (21%)]      Loss: 1.290709
Train Epoch: 9 [19200/60000 (32%)]      Loss: 1.302788
Train Epoch: 9 [25600/60000 (43%)]      Loss: 1.285059
Train Epoch: 9 [32000/60000 (53%)]      Loss: 1.307066
Train Epoch: 9 [38400/60000 (64%)]      Loss: 1.243746
Train Epoch: 9 [44800/60000 (75%)]      Loss: 1.326226
Train Epoch: 9 [51200/60000 (85%)]      Loss: 1.312101
Train Epoch: 9 [57600/60000 (96%)]      Loss: 1.269385

Test set: Average loss: 1.2724, Accuracy: 52532/60000 (88%)
```

```
Train Epoch: 10 [0/60000 (0%)]  Loss: 1.254116
Train Epoch: 10 [6400/60000 (11%)]      Loss: 1.240688
Train Epoch: 10 [12800/60000 (21%)]     Loss: 1.266507
Train Epoch: 10 [19200/60000 (32%)]     Loss: 1.282884
Train Epoch: 10 [25600/60000 (43%)]     Loss: 1.264085
Train Epoch: 10 [32000/60000 (53%)]     Loss: 1.286670
Train Epoch: 10 [38400/60000 (64%)]     Loss: 1.224357
Train Epoch: 10 [44800/60000 (75%)]     Loss: 1.306602
Train Epoch: 10 [51200/60000 (85%)]     Loss: 1.289986
Train Epoch: 10 [57600/60000 (96%)]     Loss: 1.249912

Test set: Average loss: 1.2522, Accuracy: 52733/60000 (88%)
```

Test the trained Tanh-network on the test data. Print out the accuracy.

[66]:
```
test(model_t, test_criterion, test_loader)
```

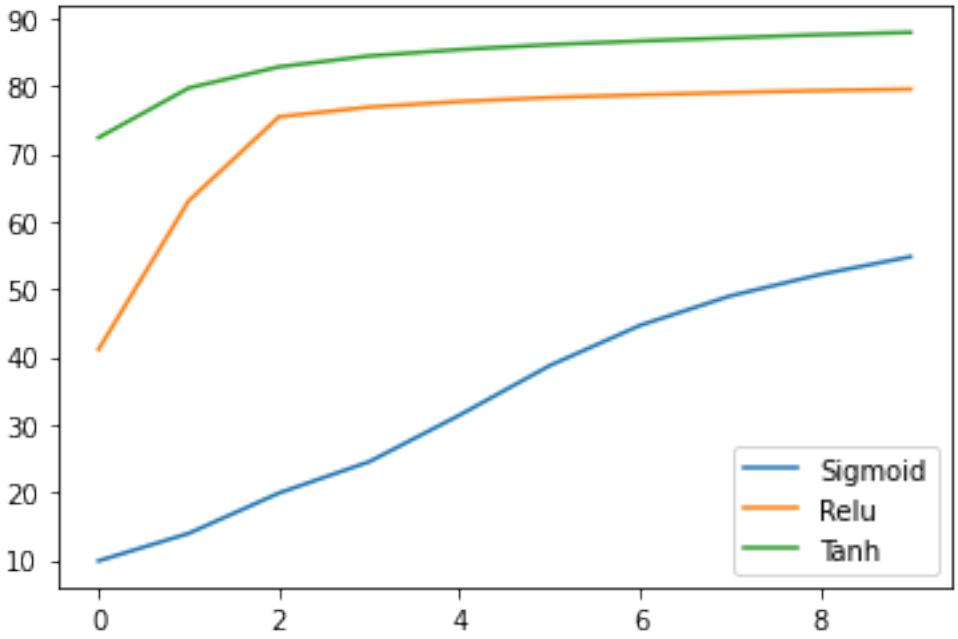
```
Test set: Average loss: 1.2454, Accuracy: 8842/10000 (88%)
```

[66]:
```
88.42
```

Plot the training accuracies over the epochs of the networks on the same figure (there should 3 line plots/scatter plots).

[67]:
```
plt.plot(train_s, label = 'Sigmoid')
plt.plot(train_r, label = 'Relu')
plt.plot(train_t, label = 'Tanh')
plt.legend()
```

[67]:
```
<matplotlib.legend.Legend at 0x7fb52f05eeb0>
```



What is your conclustion on the effect of varying the activation functions on the performance of a neural network trained on MNIST dataset?

The tanh function appeared to give the highest accuracy, while the relu converged to a lower point and the sigmoid did not seem converge after 10 epochs, so perhaps the sigmoid still has potential as a slow learner.

1.6 Part d

Finally, we will look into the effect of varying the value of the learning rate on the performance of a neural network. Create a network with one hidden layer of size 20 and ReLU activation.

[68]:
```
# Number of training epochs
epochs = 10
# Learning rate
lr = 0.1

# Define the training and testing loss
train_criterion = nn.CrossEntropyLoss()
test_criterion = nn.CrossEntropyLoss(reduction='sum')
```

Train the network on the MNIST dataset for 10 epochs. Set the learning rate to be 0.1. After each epoch, record the current training accuracy of the network.

[69]:
```
model_a = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')
lr = 0.1
optimizer = optim.SGD(model_a.parameters(),lr=lr)
```

```
train_a = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_a, train_criterion, train_loader, optimizer, epoch)
        train_a.append(test(model_a, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.324512
Train Epoch: 1 [6400/60000 (11%)]       Loss: 0.312341
Train Epoch: 1 [12800/60000 (21%)]      Loss: 0.254524
Train Epoch: 1 [19200/60000 (32%)]      Loss: 0.313385
Train Epoch: 1 [25600/60000 (43%)]      Loss: 0.211388
Train Epoch: 1 [32000/60000 (53%)]      Loss: 0.328997
Train Epoch: 1 [38400/60000 (64%)]      Loss: 0.222577
Train Epoch: 1 [44800/60000 (75%)]      Loss: 0.268873
Train Epoch: 1 [51200/60000 (85%)]      Loss: 0.236029
Train Epoch: 1 [57600/60000 (96%)]      Loss: 0.303315

Test set: Average loss: 0.2253, Accuracy: 55840/60000 (93%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 0.139828
Train Epoch: 2 [6400/60000 (11%)]       Loss: 0.197228
Train Epoch: 2 [12800/60000 (21%)]      Loss: 0.132884
Train Epoch: 2 [19200/60000 (32%)]      Loss: 0.228968
Train Epoch: 2 [25600/60000 (43%)]      Loss: 0.161075
Train Epoch: 2 [32000/60000 (53%)]      Loss: 0.236868
Train Epoch: 2 [38400/60000 (64%)]      Loss: 0.160275
Train Epoch: 2 [44800/60000 (75%)]      Loss: 0.241555
Train Epoch: 2 [51200/60000 (85%)]      Loss: 0.163595
Train Epoch: 2 [57600/60000 (96%)]      Loss: 0.269563

Test set: Average loss: 0.1685, Accuracy: 56862/60000 (95%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 0.112471
Train Epoch: 3 [6400/60000 (11%)]       Loss: 0.124982
Train Epoch: 3 [12800/60000 (21%)]      Loss: 0.110417
Train Epoch: 3 [19200/60000 (32%)]      Loss: 0.224524
Train Epoch: 3 [25600/60000 (43%)]      Loss: 0.150086
Train Epoch: 3 [32000/60000 (53%)]      Loss: 0.196960
Train Epoch: 3 [38400/60000 (64%)]      Loss: 0.100432
Train Epoch: 3 [44800/60000 (75%)]      Loss: 0.195327
Train Epoch: 3 [51200/60000 (85%)]      Loss: 0.163248
Train Epoch: 3 [57600/60000 (96%)]      Loss: 0.251062

Test set: Average loss: 0.1504, Accuracy: 57184/60000 (95%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 0.111182
```

```
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.092998
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.098265
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.173106
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.126528
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.194421
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.112932
Train Epoch: 4 [44800/60000 (75%)]      Loss: 0.166068
Train Epoch: 4 [51200/60000 (85%)]      Loss: 0.156627
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.228048

Test set: Average loss: 0.1362, Accuracy: 57454/60000 (96%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.104668
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.077030
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.079599
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.141657
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.112371
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.201750
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.135284
Train Epoch: 5 [44800/60000 (75%)]      Loss: 0.148029
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.161087
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.209256

Test set: Average loss: 0.1322, Accuracy: 57533/60000 (96%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 0.115813
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.071467
Train Epoch: 6 [12800/60000 (21%)]      Loss: 0.068759
Train Epoch: 6 [19200/60000 (32%)]      Loss: 0.116211
Train Epoch: 6 [25600/60000 (43%)]      Loss: 0.094429
Train Epoch: 6 [32000/60000 (53%)]      Loss: 0.208769
Train Epoch: 6 [38400/60000 (64%)]      Loss: 0.145213
Train Epoch: 6 [44800/60000 (75%)]      Loss: 0.127880
Train Epoch: 6 [51200/60000 (85%)]      Loss: 0.151395
Train Epoch: 6 [57600/60000 (96%)]      Loss: 0.181021

Test set: Average loss: 0.1244, Accuracy: 57652/60000 (96%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.112015
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.057496
Train Epoch: 7 [12800/60000 (21%)]      Loss: 0.071202
Train Epoch: 7 [19200/60000 (32%)]      Loss: 0.098883
Train Epoch: 7 [25600/60000 (43%)]      Loss: 0.096744
Train Epoch: 7 [32000/60000 (53%)]      Loss: 0.195164
Train Epoch: 7 [38400/60000 (64%)]      Loss: 0.140555
Train Epoch: 7 [44800/60000 (75%)]      Loss: 0.116852
Train Epoch: 7 [51200/60000 (85%)]      Loss: 0.144675
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.171584
```

```
Test set: Average loss: 0.1157, Accuracy: 57846/60000 (96%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.100874
Train Epoch: 8 [6400/60000 (11%)]       Loss: 0.049888
Train Epoch: 8 [12800/60000 (21%)]      Loss: 0.065431
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.098656
Train Epoch: 8 [25600/60000 (43%)]      Loss: 0.091312
Train Epoch: 8 [32000/60000 (53%)]      Loss: 0.178458
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.141795
Train Epoch: 8 [44800/60000 (75%)]      Loss: 0.101325
Train Epoch: 8 [51200/60000 (85%)]      Loss: 0.149491
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.160454

Test set: Average loss: 0.1144, Accuracy: 57871/60000 (96%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.097544
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.042563
Train Epoch: 9 [12800/60000 (21%)]      Loss: 0.057320
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.081184
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.093119
Train Epoch: 9 [32000/60000 (53%)]      Loss: 0.171133
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.133179
Train Epoch: 9 [44800/60000 (75%)]      Loss: 0.097496
Train Epoch: 9 [51200/60000 (85%)]      Loss: 0.144246
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.158419

Test set: Average loss: 0.1114, Accuracy: 57920/60000 (97%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.088668
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.036727
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.055895
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.075764
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.089220
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.145999
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.131782
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.099017
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.146621
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.144274

Test set: Average loss: 0.1098, Accuracy: 57939/60000 (97%)
```

Test the trained network on the test data. Print out the accuracy.

[70]:
```
test(model_a, test_criterion, test_loader)
```

```
Test set: Average loss: 0.1575, Accuracy: 9559/10000 (96%)
```

[70]: 95.59

Train the network on the MNIST dataset for 10 epochs. Set the learning rate to be 0.01. After each epoch, record the current training accuracy of the network.

[71]:
```
model_b = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')
lr = 0.01
optimizer = optim.SGD(model_b.parameters(),lr=lr)

train_b = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_b, train_criterion, train_loader, optimizer, epoch)
        train_b.append(test(model_b, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.298980
Train Epoch: 1 [6400/60000 (11%)]       Loss: 1.415908
Train Epoch: 1 [12800/60000 (21%)]      Loss: 1.323044
Train Epoch: 1 [19200/60000 (32%)]      Loss: 1.202785
Train Epoch: 1 [25600/60000 (43%)]      Loss: 0.837407
Train Epoch: 1 [32000/60000 (53%)]      Loss: 0.798797
Train Epoch: 1 [38400/60000 (64%)]      Loss: 0.765324
Train Epoch: 1 [44800/60000 (75%)]      Loss: 0.963126
Train Epoch: 1 [51200/60000 (85%)]      Loss: 0.911553
Train Epoch: 1 [57600/60000 (96%)]      Loss: 0.979912

Test set: Average loss: 0.8162, Accuracy: 43478/60000 (72%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 0.675400
Train Epoch: 2 [6400/60000 (11%)]       Loss: 0.627203
Train Epoch: 2 [12800/60000 (21%)]      Loss: 0.695437
Train Epoch: 2 [19200/60000 (32%)]      Loss: 0.908638
Train Epoch: 2 [25600/60000 (43%)]      Loss: 0.662911
Train Epoch: 2 [32000/60000 (53%)]      Loss: 0.675273
Train Epoch: 2 [38400/60000 (64%)]      Loss: 0.636540
Train Epoch: 2 [44800/60000 (75%)]      Loss: 0.893298
Train Epoch: 2 [51200/60000 (85%)]      Loss: 0.800239
Train Epoch: 2 [57600/60000 (96%)]      Loss: 0.931625

Test set: Average loss: 0.7446, Accuracy: 44339/60000 (74%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 0.583531
Train Epoch: 3 [6400/60000 (11%)]       Loss: 0.592212
Train Epoch: 3 [12800/60000 (21%)]      Loss: 0.620640
Train Epoch: 3 [19200/60000 (32%)]      Loss: 0.878048
Train Epoch: 3 [25600/60000 (43%)]      Loss: 0.623821
```

```
Train Epoch: 3 [32000/60000 (53%)]      Loss: 0.640307
Train Epoch: 3 [38400/60000 (64%)]      Loss: 0.600952
Train Epoch: 3 [44800/60000 (75%)]      Loss: 0.887341
Train Epoch: 3 [51200/60000 (85%)]      Loss: 0.756342
Train Epoch: 3 [57600/60000 (96%)]      Loss: 0.910970

Test set: Average loss: 0.7137, Accuracy: 44703/60000 (75%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 0.550697
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.568239
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.588444
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.865209
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.600921
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.621725
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.584999
Train Epoch: 4 [44800/60000 (75%)]      Loss: 0.888465
Train Epoch: 4 [51200/60000 (85%)]      Loss: 0.731478
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.893326

Test set: Average loss: 0.6933, Accuracy: 44926/60000 (75%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.536483
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.543123
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.567843
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.850565
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.588987
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.613429
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.573629
Train Epoch: 5 [44800/60000 (75%)]      Loss: 0.886284
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.710770
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.879190

Test set: Average loss: 0.6778, Accuracy: 45138/60000 (75%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 0.528496
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.522225
Train Epoch: 6 [12800/60000 (21%)]      Loss: 0.554385
Train Epoch: 6 [19200/60000 (32%)]      Loss: 0.831553
Train Epoch: 6 [25600/60000 (43%)]      Loss: 0.574143
Train Epoch: 6 [32000/60000 (53%)]      Loss: 0.607277
Train Epoch: 6 [38400/60000 (64%)]      Loss: 0.562679
Train Epoch: 6 [44800/60000 (75%)]      Loss: 0.885489
Train Epoch: 6 [51200/60000 (85%)]      Loss: 0.696746
Train Epoch: 6 [57600/60000 (96%)]      Loss: 0.866379

Test set: Average loss: 0.6653, Accuracy: 45300/60000 (76%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.517867
```

```
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.509298
Train Epoch: 7 [12800/60000 (21%)]      Loss: 0.544725
Train Epoch: 7 [19200/60000 (32%)]      Loss: 0.813968
Train Epoch: 7 [25600/60000 (43%)]      Loss: 0.563698
Train Epoch: 7 [32000/60000 (53%)]      Loss: 0.588916
Train Epoch: 7 [38400/60000 (64%)]      Loss: 0.551228
Train Epoch: 7 [44800/60000 (75%)]      Loss: 0.879637
Train Epoch: 7 [51200/60000 (85%)]      Loss: 0.675821
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.856169

Test set: Average loss: 0.6543, Accuracy: 45436/60000 (76%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.509052
Train Epoch: 8 [6400/60000 (11%)]       Loss: 0.495396
Train Epoch: 8 [12800/60000 (21%)]      Loss: 0.537554
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.799301
Train Epoch: 8 [25600/60000 (43%)]      Loss: 0.552130
Train Epoch: 8 [32000/60000 (53%)]      Loss: 0.579908
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.541948
Train Epoch: 8 [44800/60000 (75%)]      Loss: 0.874071
Train Epoch: 8 [51200/60000 (85%)]      Loss: 0.661048
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.845504

Test set: Average loss: 0.6451, Accuracy: 45549/60000 (76%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.497810
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.481755
Train Epoch: 9 [12800/60000 (21%)]      Loss: 0.533681
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.784368
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.544574
Train Epoch: 9 [32000/60000 (53%)]      Loss: 0.570323
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.535745
Train Epoch: 9 [44800/60000 (75%)]      Loss: 0.865992
Train Epoch: 9 [51200/60000 (85%)]      Loss: 0.650129
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.834754

Test set: Average loss: 0.6371, Accuracy: 45663/60000 (76%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.486285
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.469220
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.528691
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.770642
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.537360
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.562156
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.531304
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.856225
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.638969
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.825931
```

```
Test set: Average loss: 0.6304, Accuracy: 45740/60000 (76%)
```

Test the trained network on the test data. Print out the accuracy.

[72]:
```
test(model_b, test_criterion, test_loader)
```

```
Test set: Average loss: 0.6322, Accuracy: 7638/10000 (76%)
```

[72]:
```
76.38
```

Train the network on the MNIST dataset for 10 epochs. Set the learning rate to 0.001. After each epoch, record the current training accuracy of the network.

[73]:
```
model_c = mnist_network(num_hidden_layers=1,layer_size=20,activation='relu')
lr = 0.001
optimizer = optim.SGD(model_c.parameters(),lr=lr)

train_c = []

for epoch in range(1, epochs + 1):
        # Training
        train(model_c, train_criterion, train_loader, optimizer, epoch)
        train_c.append(test(model_c, test_criterion, train_loader))
```

```
Train Epoch: 1 [0/60000 (0%)]   Loss: 2.353023
Train Epoch: 1 [6400/60000 (11%)]       Loss: 2.293145
Train Epoch: 1 [12800/60000 (21%)]      Loss: 2.178953
Train Epoch: 1 [19200/60000 (32%)]      Loss: 2.157175
Train Epoch: 1 [25600/60000 (43%)]      Loss: 2.056528
Train Epoch: 1 [32000/60000 (53%)]      Loss: 2.012466
Train Epoch: 1 [38400/60000 (64%)]      Loss: 1.778736
Train Epoch: 1 [44800/60000 (75%)]      Loss: 1.817686
Train Epoch: 1 [51200/60000 (85%)]      Loss: 1.699716
Train Epoch: 1 [57600/60000 (96%)]      Loss: 1.477475

Test set: Average loss: 1.5282, Accuracy: 39361/60000 (66%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 1.623578
Train Epoch: 2 [6400/60000 (11%)]       Loss: 1.456735
Train Epoch: 2 [12800/60000 (21%)]      Loss: 1.307255
Train Epoch: 2 [19200/60000 (32%)]      Loss: 1.257192
Train Epoch: 2 [25600/60000 (43%)]      Loss: 1.343965
Train Epoch: 2 [32000/60000 (53%)]      Loss: 1.304283
Train Epoch: 2 [38400/60000 (64%)]      Loss: 0.936449
Train Epoch: 2 [44800/60000 (75%)]      Loss: 1.201425
Train Epoch: 2 [51200/60000 (85%)]      Loss: 1.149564
```

```
Train Epoch: 2 [57600/60000 (96%)]      Loss: 0.922443

Test set: Average loss: 1.0156, Accuracy: 45395/60000 (76%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 1.113865
Train Epoch: 3 [6400/60000 (11%)]       Loss: 0.934241
Train Epoch: 3 [12800/60000 (21%)]      Loss: 0.871322
Train Epoch: 3 [19200/60000 (32%)]      Loss: 0.878541
Train Epoch: 3 [25600/60000 (43%)]      Loss: 1.080655
Train Epoch: 3 [32000/60000 (53%)]      Loss: 1.046979
Train Epoch: 3 [38400/60000 (64%)]      Loss: 0.659038
Train Epoch: 3 [44800/60000 (75%)]      Loss: 0.930863
Train Epoch: 3 [51200/60000 (85%)]      Loss: 0.927686
Train Epoch: 3 [57600/60000 (96%)]      Loss: 0.744721

Test set: Average loss: 0.8352, Accuracy: 46789/60000 (78%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 0.937301
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.753811
Train Epoch: 4 [12800/60000 (21%)]      Loss: 0.700586
Train Epoch: 4 [19200/60000 (32%)]      Loss: 0.741067
Train Epoch: 4 [25600/60000 (43%)]      Loss: 0.973972
Train Epoch: 4 [32000/60000 (53%)]      Loss: 0.922415
Train Epoch: 4 [38400/60000 (64%)]      Loss: 0.550535
Train Epoch: 4 [44800/60000 (75%)]      Loss: 0.804712
Train Epoch: 4 [51200/60000 (85%)]      Loss: 0.815086
Train Epoch: 4 [57600/60000 (96%)]      Loss: 0.666530

Test set: Average loss: 0.7466, Accuracy: 47537/60000 (79%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.844682
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.662456
Train Epoch: 5 [12800/60000 (21%)]      Loss: 0.614482
Train Epoch: 5 [19200/60000 (32%)]      Loss: 0.673963
Train Epoch: 5 [25600/60000 (43%)]      Loss: 0.908774
Train Epoch: 5 [32000/60000 (53%)]      Loss: 0.849929
Train Epoch: 5 [38400/60000 (64%)]      Loss: 0.489985
Train Epoch: 5 [44800/60000 (75%)]      Loss: 0.732203
Train Epoch: 5 [51200/60000 (85%)]      Loss: 0.745532
Train Epoch: 5 [57600/60000 (96%)]      Loss: 0.622307

Test set: Average loss: 0.6925, Accuracy: 47901/60000 (80%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 0.782359
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.606320
Train Epoch: 6 [12800/60000 (21%)]      Loss: 0.562650
Train Epoch: 6 [19200/60000 (32%)]      Loss: 0.632474
Train Epoch: 6 [25600/60000 (43%)]      Loss: 0.862520
```

```
Train Epoch: 6 [32000/60000 (53%)]      Loss: 0.803974
Train Epoch: 6 [38400/60000 (64%)]      Loss: 0.450305
Train Epoch: 6 [44800/60000 (75%)]      Loss: 0.685472
Train Epoch: 6 [51200/60000 (85%)]      Loss: 0.698506
Train Epoch: 6 [57600/60000 (96%)]      Loss: 0.593154

Test set: Average loss: 0.6556, Accuracy: 48193/60000 (80%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.737935
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.569005
Train Epoch: 7 [12800/60000 (21%)]      Loss: 0.527740
Train Epoch: 7 [19200/60000 (32%)]      Loss: 0.604454
Train Epoch: 7 [25600/60000 (43%)]      Loss: 0.828974
Train Epoch: 7 [32000/60000 (53%)]      Loss: 0.773334
Train Epoch: 7 [38400/60000 (64%)]      Loss: 0.421254
Train Epoch: 7 [44800/60000 (75%)]      Loss: 0.652739
Train Epoch: 7 [51200/60000 (85%)]      Loss: 0.664618
Train Epoch: 7 [57600/60000 (96%)]      Loss: 0.572006

Test set: Average loss: 0.6288, Accuracy: 48440/60000 (81%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.704114
Train Epoch: 8 [6400/60000 (11%)]       Loss: 0.543282
Train Epoch: 8 [12800/60000 (21%)]      Loss: 0.501873
Train Epoch: 8 [19200/60000 (32%)]      Loss: 0.584986
Train Epoch: 8 [25600/60000 (43%)]      Loss: 0.802990
Train Epoch: 8 [32000/60000 (53%)]      Loss: 0.751637
Train Epoch: 8 [38400/60000 (64%)]      Loss: 0.399112
Train Epoch: 8 [44800/60000 (75%)]      Loss: 0.628883
Train Epoch: 8 [51200/60000 (85%)]      Loss: 0.639128
Train Epoch: 8 [57600/60000 (96%)]      Loss: 0.556578

Test set: Average loss: 0.6085, Accuracy: 48618/60000 (81%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.677444
Train Epoch: 9 [6400/60000 (11%)]       Loss: 0.524357
Train Epoch: 9 [12800/60000 (21%)]      Loss: 0.481715
Train Epoch: 9 [19200/60000 (32%)]      Loss: 0.570264
Train Epoch: 9 [25600/60000 (43%)]      Loss: 0.782828
Train Epoch: 9 [32000/60000 (53%)]      Loss: 0.735200
Train Epoch: 9 [38400/60000 (64%)]      Loss: 0.381736
Train Epoch: 9 [44800/60000 (75%)]      Loss: 0.611363
Train Epoch: 9 [51200/60000 (85%)]      Loss: 0.618929
Train Epoch: 9 [57600/60000 (96%)]      Loss: 0.544350

Test set: Average loss: 0.5924, Accuracy: 48791/60000 (81%)

Train Epoch: 10 [0/60000 (0%)]  Loss: 0.655728
```

```
Train Epoch: 10 [6400/60000 (11%)]      Loss: 0.510754
Train Epoch: 10 [12800/60000 (21%)]     Loss: 0.465433
Train Epoch: 10 [19200/60000 (32%)]     Loss: 0.558340
Train Epoch: 10 [25600/60000 (43%)]     Loss: 0.766528
Train Epoch: 10 [32000/60000 (53%)]     Loss: 0.722264
Train Epoch: 10 [38400/60000 (64%)]     Loss: 0.367999
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.597598
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.602003
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.534860

Test set: Average loss: 0.5793, Accuracy: 48920/60000 (82%)
```

Test the trained network on the test data. Print out the accuracy.

[74]:
```
test(model_c, test_criterion, test_loader)
```

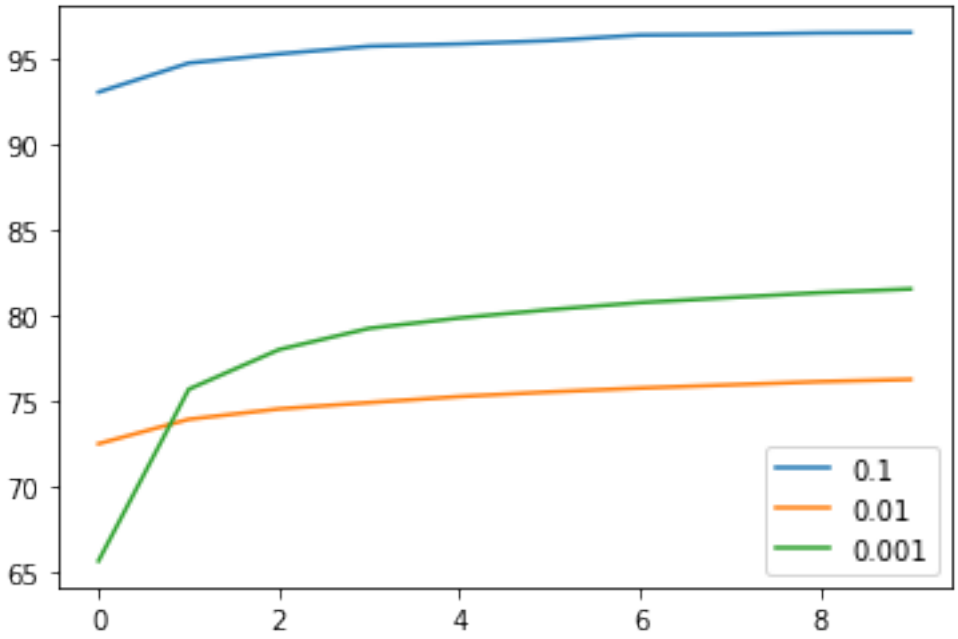
```
Test set: Average loss: 0.5652, Accuracy: 8212/10000 (82%)
```

[74]:
```
82.12
```

Plot the training accuracies over the epochs of the scenarios on the same figure (there should 3 line plots/scatter plots).

[75]:
```
plt.plot(train_a, label = '0.1')
plt.plot(train_b, label = '0.01')
plt.plot(train_c, label = '0.001')
plt.legend()
```

[75]:
```
<matplotlib.legend.Legend at 0x7fb55641b940>
```



What is your conclustion on the effect of varying the learning rate on the performance of a neural network?

The learning rate of 0.1 produced the best results, and the general trend that I saw in these three learning rates was that the smaller the learning rate, the longer it took the network to converge to a value of accuracy.

1.7 REMARK for Problem 2

You have observed the effects of varying different hyperparameters on the performance of a neural network **on the MNIST dataset**. However, keep in mind that these trends only apply for **the MNIST dataset** and should not be carried to another problem. There is no single hyperparameter settings that works for all problems. As you do more problems, you will build up your intuitions about the hyperparameters so that you can quickly deploy a good model. For example, people observed that setting the learning rate = 0.001 often works the best, though it is not always the case.

1.8 Problem 3

Experimenting with **k-anonimity, i-diversity, and t-closeness**.

Consider a dataset, for example, with 3 ordinary attributes and 1 sensitive attribute. Let the 3 ordinary attributes be Age, Sex, and Education and the sensitive attribute be Income, each row in this dataset is of the form:

$$[Age, Sex, Education, Income]$$

A hacker is interested in knowing the sensitive attribute Income. When the dataset is designed so that if complies with either **k-anomity**, **i-diversity**, and/or **t-closeness**, even if he or she somehow figures out the values of the three, the hacker may not retrive the sensitive information accurately. In general, **k-anomity** is weaker than **i-diversity**, which, in turn, is weaker than **t-closeness**.

By definition, **k-anomity** means that there is at least **k** different rows in the table of which ordinary values are a particular combination of Age, Sex, and Education. For example, the hacker knows the information of the person of interest is Age = 31, Sex = Female, and Education = BS. He or she looks into the data table and found that there are 3 rows with that combination:

$$[Age = 31, Sex = Female, Education = BS, Income = 300k]$$
$$[Age = 31, Sex = Female, Education = BS, Income = 70k]$$
$$[Age = 31, Sex = Female, Education = BS, Income = 20k]$$

The hacker cannot tell accurately what the income of the person is because it can be one of the 3 values shown. This particular combination of information has 3-anomity. If every combination corresponds to at least 3 rows, then the dataset has 3-anomity.

a) Let's look at the dataset **"table.csv"**. Let the sensitive attribute be **education** and others be ordinary attributes. Calculate the anomity of the dataset (the value **k**). First, find all the posible combinations of the ordinary attributes that exists in the dataset. After that, determine the anomity for each combination. The anomity of the dataset is the smallest anomity among the combinations.

[90]:
```
import pandas as pd
table = pd.read_csv('table.csv')
table.head(5)
```

[90]:

	Unnamed: 0	age	education	race	sex
0	0	39	Bachelors	White	Male
1	1	50	Bachelors	White	Male
2	2	38	HS-grad	White	Male
3	3	53	11th	Black	Male
4	4	28	Bachelors	Black	Female

[107]:
```
anonimity = min(table.groupby(['age', 'race', 'sex']).count()['education'])
print(f'Anonimity is {anonimity}')
```

```
Anonimity is 1
```

We can improve the **k-anomity** of the dataset by "suppressing" the ordinary attributes. Suppressing means reducing the resolution of the attribute's value. For this problem, let's suppress Age by replacing the exact age with an age range. For example, instead of leaving age = 32, replace it with age = 30-40. Apply this to **"table.csv"** with the ranges {<20, 20-30, 30-50, >50}. Check if the anomity improves.

[111]:
```
def ageRange(age):
    if age < 20:
```

```
        return '<20'
    elif age <= 30:
        return '20-30'
    elif age <=50:
        return '30-50'
    else:
        return '>50'

table['age_range'] = table['age'].apply(ageRange)
```

[113]:
```
anonimity = min(table.groupby(['age_range', 'race', 'sex']).
 ↪count()['education'])
print(f'Anonimity is {anonimity}')
```

```
Anonimity is 4
```

The anonimity improves by 3.

K-anomity is nice, however, it fails in many cases. If the rows which share a combination of ordinary attributes have only a few values for the sensitive attribute, then it is not much better than having no anomity at all. For example, consider:

$$[Age = 31, Sex = Female, Education = BS, Income = 300k]$$
$$[Age = 31, Sex = Female, Education = BS, Income = 20k]$$
$$[Age = 31, Sex = Female, Education = BS, Income = 20k]$$
$$[Age = 31, Sex = Female, Education = BS, Income = 20k]$$

When **k-anomity** fails in the second case, **i-diversity** comes to the rescue. **I-diversity** states that the rows of a particular combination of information must have at least i different values for the sensitive attribute. The above example has 2-diversity, which is not good.

b) Calculate the **i-diversity** of the dataset **"table.csv"**. Follow similar steps as in part a.

[127]:
```
table = pd.read_csv('table.csv')
table.head(5)
```

[127]:

	Unnamed: 0	age	education	race	sex
0	0	39	Bachelors	White	Male
1	1	50	Bachelors	White	Male
2	2	38	HS-grad	White	Male
3	3	53	11th	Black	Male
4	4	28	Bachelors	Black	Female

[142]:
```
table.groupby(["age","race", "sex"]).agg({"education":"nunique"}).
 ↪sort_values('education').head(5)
```

[142]:

			education
age	race	sex	

```
41   Amer-Indian-Eskimo Female          1
67   Asian-Pac-Islander Male            1
     Amer-Indian-Eskimo Male            1
39   Amer-Indian-Eskimo Male            1
66   Other              Female          1
```

The i-diversity is 1.

Suppressing an attribute can also improve the **i-diversity** of the dataset. Repeat the suppression as in **part a** and check if the diversity improves. If it does not, consider further suppress age by using the range {<20, 20-50, >50}.

[143]:
```
def ageRange_i (age):
    if age < 20:
        return '<20'
    elif age <=50:
        return '20-50'
    else:
        return '>50'

table['age_range'] = table['age'].apply(ageRange_i)
table.groupby(["age_range","race", "sex"]).agg({"education":"nunique"}).
 ↪sort_values('education').head(5)
```

[143]:
```
                                    education
age_range race              sex
<20       Amer-Indian-Eskimo Male           3
>50       Other              Female         4
<20       Amer-Indian-Eskimo Female         4
          Asian-Pac-Islander Male           4
          Other              Male           5
```

The i-diversity is improved from 1 to 3.

T-closeness is even better than **i-diversity**. **T-closeness** requires that for every combination of information, the distribution of the sensitive attribute's value among the corresponding rows must be close to the overall distribution of the sensitive attribute's value for the whole dataset. Distance between distribution is calculated using the Earth Mover Distance (EMD). The dataset has **t-closeness** if no distance exceeds **t**.

c) Calculate the overall distribution of **education**. Find the **t-closeness** of the dataset (largest distance between any combination's distribution of marital-status and the overall distribution).

You can use **scipy.stats.wasserstein_distance** to calculate the EMD.

[144]:
```
from scipy.stats import wasserstein_distance

table = pd.read_csv('table.csv')
```

[148]:
```
table['race'] = pd.factorize(table['race'])[0]
table['sex'] = pd.factorize(table['sex'])[0]
table['education'] = pd.factorize(table['education'])[0]

ages = np.unique(table['age'])
races = np.unique(table['race'])
sexes = np.unique(table['sex'])
```

[149]:
```
education = (table['education'])
distances = []

for age in range(len(ages)):
    for race in range(len(races)):
        for sex in range(len(sexes)):
            temp = table[(table['age']==age) & (table['race']==race) &␣
 ↪(table['sex']==sex)]
            if temp.empty:
                distances.append(wasserstein_distance([0], education))
            else:
                use = temp.iloc[:, 2]
                distances.append(wasserstein_distance(use, education))
```

[155]:
```
closeness = max(distances) - min(distances)
closeness
```

[155]:
```
11.44312068583628
```

[ ]:
```

```