MLP_Assignment1

September 10, 2023

Name: Siddharth Betala

Roll Number: BE19B032

PS: **In this assignment, whenever I have used the term SGD to represent mini-batch/batch gradient descent. Please excuse this misnomer.**

1 Imports

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
from torch.utils.data import DataLoader, TensorDataset
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
```

[2]:
```
import torch
import numpy as np
import random

def set_seed(seed_value=42):
    random.seed(seed_value)
    np.random.seed(seed_value)
    torch.manual_seed(seed_value)
    torch.cuda.manual_seed_all(seed_value)

set_seed(42)
```

2 Activation Functions

[3]:
```
class Activation:

    def __init__(self, *args, **kwargs):
        self.grads = {}
        self.backprop_cache = {}
```

```
    def __call__(self, *args, **kwargs):
        op = self.forward(*args, **kwargs)
        self.grads = self.calc_grads(*args, **kwargs)
        return op

    def forward(self, *args, **kwargs):
        pass

    def calc_grads(self, *args, **kwargs):
        pass

    def backward(self, *args, **kwargs):
        pass
```

[4]:
```
class Sigmoid(Activation):

    def __init__(self):
        super().__init__()

    def forward(self, x):
        self.backprop_cache = 1/(1+np.exp(-x))
        return self.backprop_cache

    def calc_grads(self, x):
        id = "x"
        y = self.backprop_cache
        delta = y*(1-y)
        return {id:delta}

    def backward(self, yhat):
        return self.grads['x']*yhat

class Tanh(Activation):

    def __init__(self):
        super().__init__()

    def forward(self, x):
        self.backprop_cache = (np.exp(x) - np.exp(-x))/(np.exp(x)+np.exp(-x))
        return self.backprop_cache

    def calc_grads(self, x):
        id = "x"
        y = self.backprop_cache
        delta = 1-y**2
        return {id:delta}
```

```
    def backward(self, yhat):
        return self.grads['x']*yhat

class ReLU(Activation):

    def __init__(self):
        super().__init__()

    def forward(self, x):
        self.backprop_cache = np.maximum(x, 0.0)
        return self.backprop_cache

    def calc_grads(self, x):
        id = "x"
        y = self.backprop_cache
        delta = (y>0).astype("float")
        return {id:delta}

    def backward(self, yhat):
        return self.grads['x']*yhat

class LeakyReLU(Activation):

    def __init__(self, alpha = 0.1):
        super().__init__()
        self.alpha = alpha

    def forward(self, x):
        self.backprop_cache = np.maximum(x, self.alpha*x)
        return self.backprop_cache

    def calc_grads(self, x):
        id = "x"
        y = self.backprop_cache
        delta = np.where(y > 0, 1, self.alpha)
        return {id:delta}

    def backward(self, yhat):
        return self.grads['x']*yhat

class Linear(Activation):

    def __init__(self):
        super().__init__()

    def forward(self, x):
        return x
```

```
    def calc_grads(self, x):
        id = "x"
        delta = np.ones_like(x)
        return {id:delta}

    def backward(self, yhat):
        return self.grads['x'] * yhat
```

3 Loss Functions

3.1 Helper Function: Softmax for Output Layer

[5]:
```
def softmax(x):
    exp_x = np.exp(x - np.max(x, axis= -1, keepdims=True))
    return exp_x/ np.sum(exp_x, axis = -1, keepdims=True)
```

[6]:
```
class Loss:

    def __init__(self, *args, **kwargs):
        self.grads = {}
        self.backprop_cache = {}

    def __call__(self, y_pred, y_true, *args, **kwargs):
        op = self.forward(y_pred, y_true, *args, **kwargs)
        self.grads = self.calc_grads(y_pred, y_true, *args, **kwargs)
        return op

    def forward(self, y_pred, y_true, *args, **kwargs):
        pass

    def calc_grads(self, y_pred, y_true, *args, **kwargs):
        pass

    def backward(self, *args, **kwargs):
        return self.grads['x']
```

[7]:
```
class MSE(Loss):

    def __init__(self):
        super().__init__()

    def forward(self, y_pred, y_true):
        num_classes = y_pred.shape[-1]
        probabs = softmax(y_pred)
```

```
        y_true_encoding = np.eye(num_classes, dtype = int)[np.array(y_true).
↪astype(int)]
        self.backprop_cache['y_true'] = y_true_encoding
        loss = np.mean(np.sum((probabs-y_true_encoding)**2, axis =1))
        self.backprop_cache['probabs'] = probabs
        return loss

    def calc_grads(self, y_pred, y_true):
        batch_size = y_pred.shape[0]
        sub_term = self.backprop_cache["probabs"] - self.
↪backprop_cache["y_true"]
        grad = (sub_term - (sub_term*self.backprop_cache["probabs"]).sum(axis =␣
↪1, keepdims = True))*self.backprop_cache["probabs"]
        grad = grad/batch_size
        return {'x': grad}


class LogLoss(Loss):

    def __init__(self):
        super().__init__()

    def forward(self, y_pred, y_true):
        num_classes = y_pred.shape[-1]
        probabs = softmax(y_pred)
        y_true_encoding = np.eye(num_classes, dtype = int)[np.array(y_true).
↪astype(int)]
        self.backprop_cache['y_true'] = y_true_encoding
        loss = np.mean(- np.log(probabs[np.arange(y_pred.shape[0]), y_true] +␣
↪1e-16)) #added the 1e-16 term to avoid division by 0 error in log
        self.backprop_cache['probabs'] = probabs
        return loss

    def calc_grads(self, y_pred, y_true):
        batch_size = y_pred.shape[0]
        grad = self.backprop_cache['probabs'] - self.backprop_cache['y_true']
        grad = grad/batch_size
        return {'x': grad}
```

4 Optimizers

Formulae for different optimizers taken from class slides and online resources

[8]:
```
class Optimizer:

    def __init__(self, *args, **kwargs):
```

```
        self.history = {}
        pass

    def update_weights(self, layer, *args, **kwargs):

        update = self.calc_update(layer)
        for k, v in layer.weights.items():
            layer.weights[k] = layer.weights[k] + update[k]

    def calc_update(self, layer, *args, **kwargs):
        pass

    def _get_unique_key(self, layer, key):
        """Helper function to get a unique key for the layer and attribute."""
        return (id(layer), key)
```

[9]:
```
class SGD(Optimizer):
    '''
    Although this has been named as SGD, it acts as mini-batch gradient descent.
    I have used the terms interchangeably in this assignment.
    '''

    def __init__(self, learning_rate = 0.01):

        super().__init__()
        self.learning_rate = learning_rate

    def calc_update(self, layer):

        update = {}

        for k, v in layer.weights.items():
            update[k] = -self.learning_rate*layer.del_theta[k]
        return update

class Momentum(Optimizer):

    def __init__(self, learning_rate=0.01, beta=0.9):

        super().__init__()
        self.learning_rate = learning_rate
        self.beta = beta

    def calc_update(self, layer):

        update = {}
```

```
        for k, v in layer.weights.items():
            unique_key = self._get_unique_key(layer, k)

            if unique_key in self.history:
                self.history[unique_key]['u'] = self.beta * self.
 ↪history[unique_key]['u'] + self.learning_rate * layer.del_theta[k]
            else:
                self.history[unique_key] = {}
                self.history[unique_key]['u'] = self.learning_rate * layer.
 ↪del_theta[k]

            update[k] = -self.history[unique_key]['u']

        return update

class RMSProp(Optimizer):

    def __init__(self, learning_rate = 0.01, beta = 0.9, epsilon = 1e-7):

        super().__init__()
        self.learning_rate = learning_rate
        self.beta = beta
        self.epsilon = epsilon

    def calc_update(self, layer):

        update = {}

        for k, v in layer.weights.items():
            unique_key = self._get_unique_key(layer, k)

            if unique_key in self.history:
                self.history[unique_key]['u'] = self.beta*self.
 ↪history[unique_key]['u'] + (1-self.beta)*(layer.del_theta[k]**2)
            else:
                self.history[unique_key] = {}
                self.history[unique_key]['u'] = (1-self.beta)*(layer.
 ↪del_theta[k]**2)

            sqrt_term = np.sqrt(self.history[unique_key]['u'] + self.epsilon)
            update[k] = -(self.learning_rate*layer.del_theta[k]/sqrt_term)
        return update

class Adam(Optimizer):

    def __init__(self, learning_rate = 0.01, epsilon = 1e-7, beta1 = 0.9, beta2␣
 ↪= 0.999):
```

```
        super().__init__()
        self.learning_rate = learning_rate
        self.epsilon = epsilon
        self.beta1 = beta1
        self.beta2 = beta2
        self.steps = 1

    def calc_update(self, layer):

        update = {}

        for k, v in layer.weights.items():
            unique_key = self._get_unique_key(layer, k)

            if unique_key in self.history:
                self.history[unique_key]['m'] = self.beta1*self.
↪history[unique_key]['m'] + (1-self.beta1)*layer.del_theta[k]
                self.history[unique_key]['u'] = self.beta2*self.
↪history[unique_key]['u'] + (1-self.beta2)*(layer.del_theta[k]**2)

            else:
                self.history[unique_key] = {}
                self.history[unique_key]['m'] = (1-self.beta1)*layer.
↪del_theta[k]
                self.history[unique_key]['u'] = (1-self.beta2)*(layer.
↪del_theta[k]**2)

            corrected_avg = self.history[unique_key]['m']/(1-(self.beta1)**self.
↪steps)
            corrected_sq_avg = self.history[unique_key]['u']/(1-(self.
↪beta2)**self.steps)

            sqrt_term = np.sqrt(corrected_sq_avg) + self.epsilon
            update[k] = -self.learning_rate*corrected_avg/sqrt_term

        self.steps+=1
        return update
```

5 Layer Class

5.1 Helper Class: Weight Initialization

[10]:
```python
def xavier_weight_init(in_dim, out_dim):
    limit = np.sqrt(6 / (in_dim + out_dim))
    weights = np.random.uniform(low=-limit, high=limit, size=(in_dim, out_dim))
    biases = np.zeros((1, out_dim))
    return weights, biases
```

[11]:
```python
class Layer:

    def __init__(self, *args, **kwargs):
        self.grads = {}
        self.weights = {}
        self.backprop_cache = {}
        self.optimizer = None

    def __call__(self, *args, **kwargs):

        op = self.forward(*args, **kwargs)
        self.grads = self.calc_grads(*args, **kwargs)
        return op

    def init_weights(self, *args, **kwargs):
        pass

    def forward(self, *args, **kwargs):
        pass

    def calc_grads(self, *args, **kwargs):
        pass

    def backward(self, *args, **kwargs):
        pass

    def update_weights(self, *args, **kwargs):
        self.optimizer.update_weights(self)
```

[12]:
```python
class FNNLayer(Layer):

    def __init__(self, in_dim, out_dim, weight_decay = None):
        super().__init__()
        self.in_dim = in_dim
        self.out_dim = out_dim
        self.weight_decay = weight_decay
        self.init_weights()
```

```
    def init_weights(self):
        self.weights['w'], self.weights['b'] = xavier_weight_init(self.in_dim,␣
↪self.out_dim)

    def forward(self, x):
        self.backprop_cache['x'] = x
        op = np.einsum('ij,jk->ik', x, self.weights["w"]) + self.weights["b"]
        return op

    def calc_grads(self, x):
        dels = {}
        dels['w'] = np.einsum('ij -> ji', self.backprop_cache['x'])
        dels['x'] = np.einsum('ij -> ji', self.weights['w'])

        return dels

    def backward(self, yhat):
        xhat = np.einsum('ij,jk->ik', yhat, self.grads["x"])
        what = np.einsum('ij,jk->ik', self.grads["w"], yhat)
        bhat = np.sum(yhat, axis=0, keepdims=True)
        if self.weight_decay:
            what = what + 2 * self.weight_decay * self.weights["w"]
            bhat = bhat + 2 * self.weight_decay * self.weights["b"]
        self.del_theta = {'w': what, 'b': bhat}
        return xhat

    def update_weights(self):
        self.optimizer.update_weights(self)
```

6 Putting it all together as a NN

6.1 Helper Function for Copy

[13]:
```
from copy import deepcopy as std_deepcopy

def custom_deepcopy(arr):
    if isinstance(arr, np.ndarray):
        return np.array([custom_deepcopy(elem) for elem in arr])
    else:
        return std_deepcopy(arr)
```

[14]:
```
class NN():
    def __init__(self, layers):
        self.layers = layers
        self.history = []
```

```
        self.iteration_losses = []
        self.iteration_test_losses = []

    def __call__(self, *args, **kwargs):
        return self.forward(*args, **kwargs)

    def compile(self, loss, optimizer):
        self.loss = loss
        for layer in self.layers:
            if isinstance(layer, (Layer, FNNLayer)) and layer.optimizer is None:
                if isinstance(optimizer, np.ndarray):
                    layer.optimizer = custom_deepcopy(optimizer)
                else:
                    layer.optimizer = std_deepcopy(optimizer)

    def forward(self, x):
        for layer in self.layers:
            x = layer(x)
        return x

    def backward(self):
        grad = self.loss.backward()
        for layer in reversed(self.layers):
            grad = layer.backward(grad)
        return grad

    def update_weights(self):
        for layer in reversed(self.layers):
            if isinstance(layer, (Layer, FNNLayer)):
                layer.update_weights()

    def fit(self, X_train, Y_train, X_test, Y_test, batch_size=64, epochs=15,␣
↪lr=0.01, log_interval=200):
        train_dataset = TensorDataset(torch.tensor(X_train), torch.
↪tensor(Y_train))
        test_dataset = TensorDataset(torch.tensor(X_test), torch.tensor(Y_test))

        train_loader = DataLoader(train_dataset, batch_size=batch_size,␣
↪shuffle=True)
        test_loader = DataLoader(test_dataset, batch_size=batch_size,␣
↪shuffle=False)

        iteration_counter = 0
        for epoch in range(1, epochs+1):
            tr_loss_epoch, tr_acc = 0, 0

            # Training phase
```

```
            for batch_idx, (X_t, Y_t) in enumerate(train_loader):
                X_t, Y_t = X_t.numpy(), Y_t.numpy()
                preds = self(X_t)
                tr_loss_batch = self.loss(preds, Y_t)
                tr_loss_epoch += tr_loss_batch
                tr_acc += np.mean(np.argmax(preds, axis=1) == Y_t)
                self.backward()
                self.update_weights()

                iteration_counter += 1
                if iteration_counter % log_interval == 0:
                    te_loss = 0
                    test_batches = 0  # Counter for the number of test batches
                    for X_te, Y_te in test_loader:
                        X_te, Y_te = X_te.numpy(), Y_te.numpy()
                        te_preds = self(X_te)
                        te_loss += self.loss(te_preds, Y_te)
                        test_batches += 1

                    average_te_loss = te_loss / test_batches  # Average the␣
↪test loss over batches
                    self.iteration_losses.append(tr_loss_batch)  # Store the␣
↪training loss
                    self.iteration_test_losses.append(average_te_loss)  # Store␣
↪the test loss

                    print(f"Iteration: {iteration_counter} Train Loss:␣
↪{tr_loss_batch} Test Loss: {average_te_loss}")

            mean_tr_loss_epoch = tr_loss_epoch / (batch_idx + 1)  # Average␣
↪loss per batch
            mean_tr_acc = tr_acc / (batch_idx + 1)  # Average accuracy per batch
            self.history.append({
                "Epoch": epoch,
                "Train Loss": mean_tr_loss_epoch,
                "Train Accuracy": mean_tr_acc
            })

            print(f"Epoch: {epoch} Mean Train Loss: {mean_tr_loss_epoch} Train␣
↪Accuracy: {mean_tr_acc * 100:.2f}%")

        print('Model trained!')

    def evaluate(self, X_train, Y_train, X_test, Y_test):
    # Train accuracy
        train_preds = self(X_train)
```

```
        train_loss = self.loss(train_preds, Y_train)
        train_accuracy = np.mean(np.argmax(train_preds, axis=1) == Y_train)

        # Test accuracy
        test_preds = self(X_test)
        test_loss = self.loss(test_preds, Y_test)
        test_accuracy = np.mean(np.argmax(test_preds, axis=1) == Y_test)

        true_labels = Y_test
        predicted_labels = np.argmax(test_preds, axis=1)
        cm = confusion_matrix(true_labels, predicted_labels)
        cr = classification_report(true_labels, predicted_labels)

        print(f"Train loss: {train_loss} Train accuracy: {train_accuracy * 100:.
↪2f}%")
        print(f"Test loss: {test_loss} Test accuracy: {test_accuracy * 100:.
↪2f}%")
        print("\nConfusion Matrix:\n", cm)
        print("\nClassification Report:\n", cr)


    def plot_losses(self):
        plt.figure(figsize=(10, 5))
        plt.plot(self.iteration_losses, label='Training Loss', color='blue')
        plt.plot(self.iteration_test_losses, label='Test Loss', color='red',␣
↪linestyle='--')
        plt.xlabel('Iterations (x200)')
        plt.ylabel('Loss')
        plt.title('Loss Convergence')
        plt.legend()
        plt.grid(True)
        plt.show()

    def plot_confusion_matrix(self, X_test, Y_test):
        preds = self(X_test)
        predicted_labels = np.argmax(preds, axis=1)
        cm = confusion_matrix(Y_test, predicted_labels)

        cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

        plt.figure(figsize=(10, 8))

        sns.heatmap(cm_normalized, annot=cm, fmt='g', cmap='Blues',
                    xticklabels=[str(i) for i in range(10)],
                    yticklabels=[str(i) for i in range(10)])

        plt.xlabel('Predicted Labels')
```

```
        plt.ylabel('True Labels')
        plt.title('Confusion Matrix')
        plt.show()
```

6.2 Loading the dataset

[15]:
```
import torchvision.transforms as transforms
from torchvision.datasets import MNIST
from sklearn.preprocessing import StandardScaler

transform = transforms.Compose([transforms.ToTensor(), transforms.Lambda(lambda␣
 ↪x: x.numpy().reshape(28*28))])
mnist_train = MNIST(root='./data', train=True, download=True,␣
 ↪transform=transform)
mnist_test = MNIST(root='./data', train=False, download=True,␣
 ↪transform=transform)

# Extract transformed data
X_train = [data[0] for data in mnist_train]
Y_train = [data[1] for data in mnist_train]
X_test = [data[0] for data in mnist_test]
Y_test = [data[1] for data in mnist_test]

X_train = np.vstack(X_train)
Y_train = np.array(Y_train)
X_test = np.vstack(X_test)
Y_test = np.array(Y_test)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to
./data/MNIST/raw/train-images-idx3-ubyte.gz
```

```
100%|      | 9912422/9912422 [00:00<00:00, 115686038.46it/s]
```

```
Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to
./data/MNIST/raw/train-labels-idx1-ubyte.gz
```

```
100%|      | 28881/28881 [00:00<00:00, 29119157.17it/s]
```

```
Extracting ./data/MNIST/raw/train-labels-idx1-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
```

```
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to
./data/MNIST/raw/t10k-images-idx3-ubyte.gz


100%|      | 1648877/1648877 [00:00<00:00, 28049299.55it/s]

Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to
./data/MNIST/raw/t10k-labels-idx1-ubyte.gz

100%|      | 4542/4542 [00:00<00:00, 15475652.94it/s]

Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw
```

7 The MLP

7.1 Model Setup

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    Sigmoid(),
    FNNLayer(500, 250),
    Sigmoid(),
    FNNLayer(250, 100),
    Sigmoid(),
    FNNLayer(100, output_dim)
]

model = NN(layers)
```

7.2 Model Compilation

[ ]:
```
optimizer = SGD()
loss = LogLoss()
model.compile(loss=loss, optimizer=optimizer)
```

7.3 Training

[ ]:
```
model.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 2.28404012346916 Test Loss: 2.2881730692377955
Iteration: 400 Train Loss: 2.2871254061109076 Test Loss: 2.2752235510892405
Iteration: 600 Train Loss: 2.273921054489516 Test Loss: 2.2606282347853868
```

```
Iteration: 800 Train Loss: 2.2269116640387248 Test Loss: 2.246660426078933
Epoch: 1 Mean Train Loss: 2.2717962347989644 Train Accuracy: 20.39%
Iteration: 1000 Train Loss: 2.233237676925383 Test Loss: 2.226415003427589
Iteration: 1200 Train Loss: 2.2372321501307884 Test Loss: 2.2033946405378333
Iteration: 1400 Train Loss: 2.1892218536408423 Test Loss: 2.1760851011238542
Iteration: 1600 Train Loss: 2.157169121515797 Test Loss: 2.142309073982135
Iteration: 1800 Train Loss: 2.1170892522196443 Test Loss: 2.0992441334576637
Epoch: 2 Mean Train Loss: 2.1708157127067658 Train Accuracy: 41.31%
Iteration: 2000 Train Loss: 2.060762568881711 Test Loss: 2.0423580138521213
Iteration: 2200 Train Loss: 1.9380380403103494 Test Loss: 1.9722423582668709
Iteration: 2400 Train Loss: 1.9165478156487006 Test Loss: 1.8886983632944314
Iteration: 2600 Train Loss: 1.7860134715862563 Test Loss: 1.7886127027419703
Iteration: 2800 Train Loss: 1.7623253775299375 Test Loss: 1.6818307460995259
Epoch: 3 Mean Train Loss: 1.9047456534684024 Train Accuracy: 51.34%
Iteration: 3000 Train Loss: 1.5932655894413061 Test Loss: 1.578380549454943
Iteration: 3200 Train Loss: 1.5245164752663785 Test Loss: 1.4804370978347827
Iteration: 3400 Train Loss: 1.32752080245697 Test Loss: 1.3919146302064385
Iteration: 3600 Train Loss: 1.316653983882106 Test Loss: 1.3140807863564923
Epoch: 4 Mean Train Loss: 1.4589482532578106 Train Accuracy: 60.19%
Iteration: 3800 Train Loss: 1.2092542831580648 Test Loss: 1.242943228480471
Iteration: 4000 Train Loss: 1.096358143285211 Test Loss: 1.179239083218037
Iteration: 4200 Train Loss: 1.090922557895781 Test Loss: 1.1206933592637163
Iteration: 4400 Train Loss: 1.0715011738443114 Test Loss: 1.0667053438913585
Iteration: 4600 Train Loss: 0.9896281036329937 Test Loss: 1.0157219407597644
Epoch: 5 Mean Train Loss: 1.1241813086430898 Train Accuracy: 70.84%
Iteration: 4800 Train Loss: 0.938165898101296 Test Loss: 0.9690247306758564
Iteration: 5000 Train Loss: 0.8995317195835184 Test Loss: 0.9245470514664017
Iteration: 5200 Train Loss: 0.8699612112337809 Test Loss: 0.8827607479649416
Iteration: 5400 Train Loss: 0.8118284174285131 Test Loss: 0.8452390406424215
Iteration: 5600 Train Loss: 0.6803703487822198 Test Loss: 0.8098093028772811
Epoch: 6 Mean Train Loss: 0.8981919012610583 Train Accuracy: 77.33%
Iteration: 5800 Train Loss: 0.679874725436631 Test Loss: 0.7765800327144426
Iteration: 6000 Train Loss: 0.7279237877918641 Test Loss: 0.746403881344753
Iteration: 6200 Train Loss: 0.8433162714411044 Test Loss: 0.7184342455330311
Iteration: 6400 Train Loss: 0.7658239374491156 Test Loss: 0.6918949386821384
Epoch: 7 Mean Train Loss: 0.7387280651947764 Train Accuracy: 81.48%
Iteration: 6600 Train Loss: 0.6394602193565357 Test Loss: 0.6687736729709524
Iteration: 6800 Train Loss: 0.7910152309915808 Test Loss: 0.6455717864720544
Iteration: 7000 Train Loss: 0.6126157696911012 Test Loss: 0.6245770570013165
Iteration: 7200 Train Loss: 0.5450020138786265 Test Loss: 0.6058778157072467
Iteration: 7400 Train Loss: 0.621565815950639 Test Loss: 0.588341988386481
Epoch: 8 Mean Train Loss: 0.6273713941197117 Train Accuracy: 84.16%
Iteration: 7600 Train Loss: 0.6520502047167622 Test Loss: 0.5718733737021341
Iteration: 7800 Train Loss: 0.6527222272493199 Test Loss: 0.554998559801712
Iteration: 8000 Train Loss: 0.5144428362140812 Test Loss: 0.5405160020578622
Iteration: 8200 Train Loss: 0.46635738436414587 Test Loss: 0.5266756296711285
Iteration: 8400 Train Loss: 0.6144956839410964 Test Loss: 0.5147275383542577
Epoch: 9 Mean Train Loss: 0.5477839359184904 Train Accuracy: 85.98%
```

```
Iteration: 8600 Train Loss: 0.5782137170619571 Test Loss: 0.5015190463406792
Iteration: 8800 Train Loss: 0.4478042495878631 Test Loss: 0.4916331095781057
Iteration: 9000 Train Loss: 0.504249579886346 Test Loss: 0.48012879896999505
Iteration: 9200 Train Loss: 0.5129467724612367 Test Loss: 0.47120738753873387
Epoch: 10 Mean Train Loss: 0.4899696946449459 Train Accuracy: 87.24%
Iteration: 9400 Train Loss: 0.5654260045232984 Test Loss: 0.4623544671640492
Iteration: 9600 Train Loss: 0.5663841887598138 Test Loss: 0.4537431425424106
Iteration: 9800 Train Loss: 0.4088040864015019 Test Loss: 0.44509356548482587
Iteration: 10000 Train Loss: 0.40403547341071 Test Loss: 0.4376180960581893
Iteration: 10200 Train Loss: 0.37587696609326526 Test Loss: 0.4304286947458322
Epoch: 11 Mean Train Loss: 0.44720210331353694 Train Accuracy: 88.10%
Iteration: 10400 Train Loss: 0.7056311804887792 Test Loss: 0.4235561129297255
Iteration: 10600 Train Loss: 0.3780131432543117 Test Loss: 0.41770869095411645
Iteration: 10800 Train Loss: 0.5843399809480122 Test Loss: 0.4118387288766452
Iteration: 11000 Train Loss: 0.41363483912169063 Test Loss: 0.4064487815741835
Iteration: 11200 Train Loss: 0.4321991786408673 Test Loss: 0.40219661314845284
Epoch: 12 Mean Train Loss: 0.41593156387885044 Train Accuracy: 88.71%
Iteration: 11400 Train Loss: 0.27643260522884233 Test Loss: 0.39728385421592527
Iteration: 11600 Train Loss: 0.3797994273795511 Test Loss: 0.39186512529404566
Iteration: 11800 Train Loss: 0.41290724652188737 Test Loss: 0.3879238714549154
Iteration: 12000 Train Loss: 0.5085063202934282 Test Loss: 0.38442718221500216
Epoch: 13 Mean Train Loss: 0.3922256332646561 Train Accuracy: 89.21%
Iteration: 12200 Train Loss: 0.3261393551309578 Test Loss: 0.38070164548530694
Iteration: 12400 Train Loss: 0.24302023948718113 Test Loss: 0.37673109771956925
Iteration: 12600 Train Loss: 0.33919987761042214 Test Loss: 0.3730094726098978
Iteration: 12800 Train Loss: 0.29526959233233774 Test Loss: 0.3693850104882014
Iteration: 13000 Train Loss: 0.31816805599423054 Test Loss: 0.36606076191210246
Epoch: 14 Mean Train Loss: 0.37380278342576756 Train Accuracy: 89.53%
Iteration: 13200 Train Loss: 0.526608835101261 Test Loss: 0.36402524638337064
Iteration: 13400 Train Loss: 0.40675676881217737 Test Loss: 0.36052866450643
Iteration: 13600 Train Loss: 0.3792932682524047 Test Loss: 0.3579500619178222
Iteration: 13800 Train Loss: 0.34791680504823164 Test Loss: 0.3545380568482295
Iteration: 14000 Train Loss: 0.2936769655721373 Test Loss: 0.35227581758745363
Epoch: 15 Mean Train Loss: 0.3589401072714596 Train Accuracy: 89.90%
Model trained!
```

7.4 Evaluation

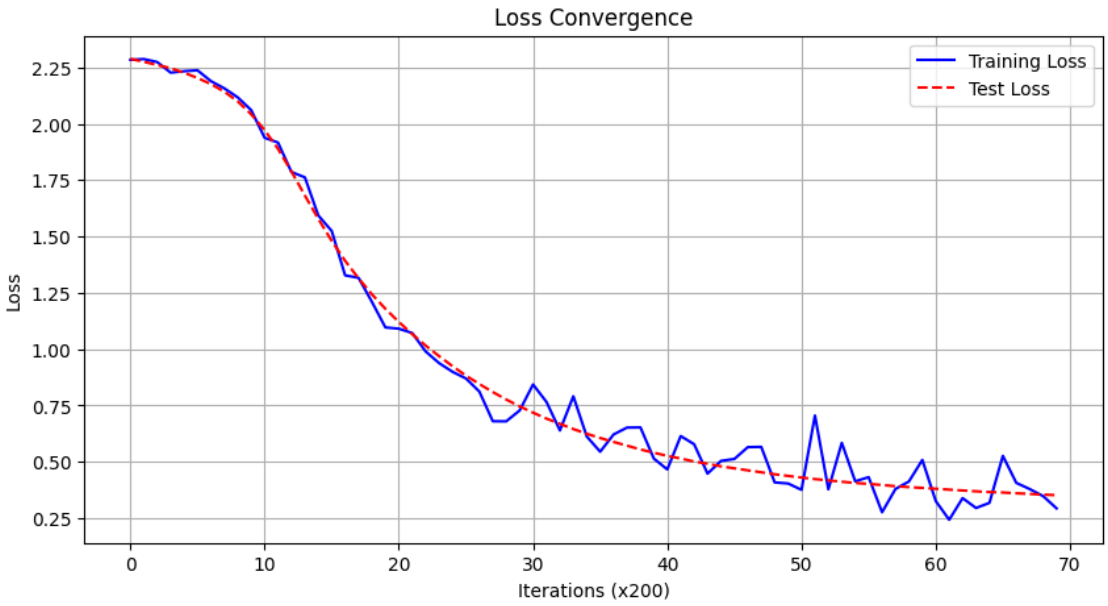
[ ]:
```
model.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.3519457452554122 Train accuracy: 90.09%
Test loss: 0.35226903080223715 Test accuracy: 90.09%

Confusion Matrix:
 [[ 953    0    5    1    1    8    8    1    3    0]
 [   0 1108    3    4    1    0    3    1   14    1]
 [  13    6  926   17   10    1   16    7   34    2]
 [   1    2   31  892    1   36    2   20   19    6]
```

```
[   1    5    5    0  915    1   16    7    5   27]
[  16    2    8   59   16  720   21   11   32    7]
[  15    3   14    0   10   18  894    0    4    0]
[   4   20   23    3    5    0    0  915    3   55]
[   9   13   10   26   19   35   12    9  825   16]
[  15    7    1    9   63    8    0   36    9  861]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       980
           1       0.95      0.98      0.96      1135
           2       0.90      0.90      0.90      1032
           3       0.88      0.88      0.88      1010
           4       0.88      0.93      0.90       982
           5       0.87      0.81      0.84       892
           6       0.92      0.93      0.93       958
           7       0.91      0.89      0.90      1028
           8       0.87      0.85      0.86       974
           9       0.88      0.85      0.87      1009

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000
```

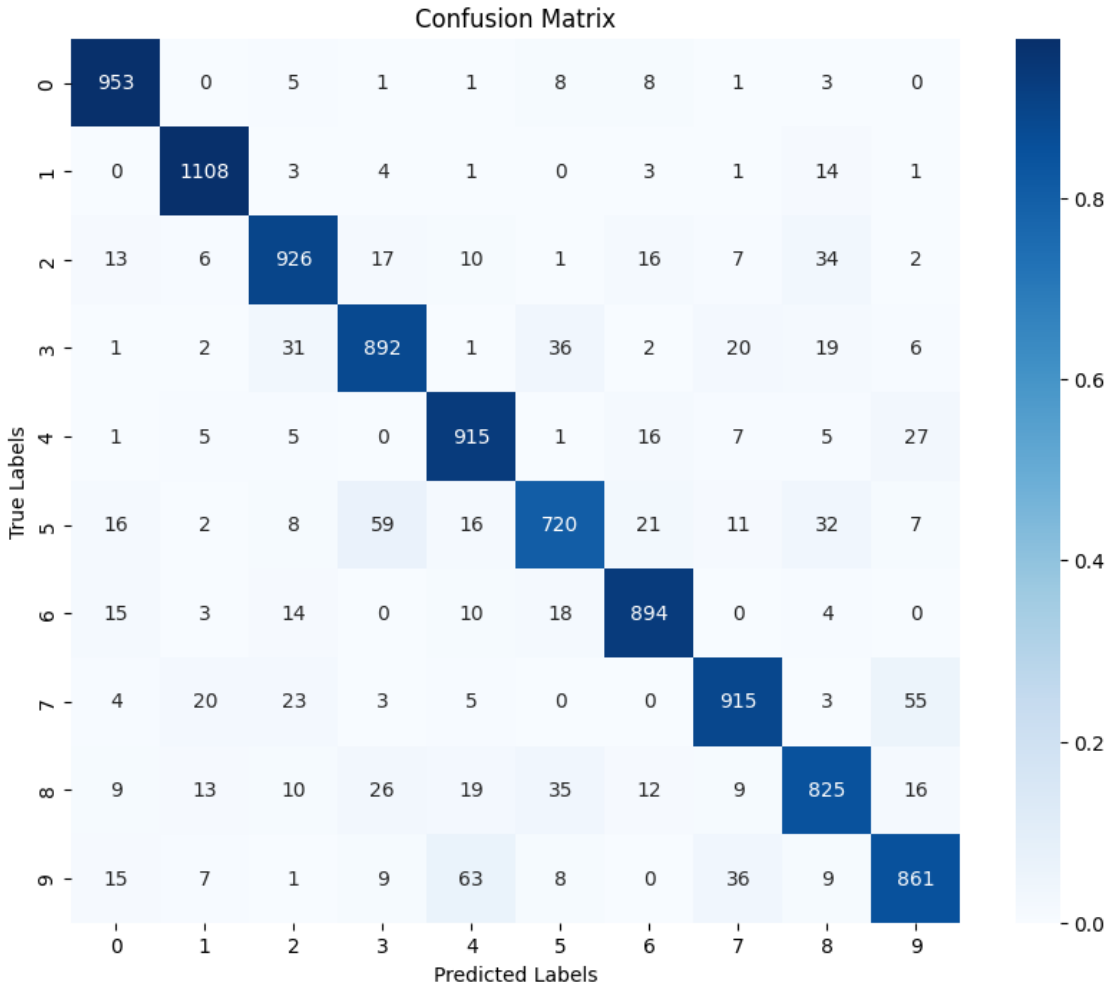
7.5 Loss Plot Visualization

[ ]:
```
model.plot_losses()
```



7.6 Confusion Matrix

[ ]:
```
model.plot_confusion_matrix(X_test, Y_test)
```



7.7 Performance with Tanh Activation Function

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    Tanh(),
    FNNLayer(500, 250),
```

```
    Tanh(),
    FNNLayer(250, 100),
    Tanh(),
    FNNLayer(100, output_dim)
]

tanh_model = NN(layers)
optimizer = SGD()
loss = LogLoss()
tanh_model.compile(loss=loss, optimizer=optimizer)
tanh_model.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.6369368598487277 Test Loss: 0.5204944292061252
Iteration: 400 Train Loss: 0.4731155648871941 Test Loss: 0.3966134160792771
Iteration: 600 Train Loss: 0.3593909323036678 Test Loss: 0.34781203184417436
Iteration: 800 Train Loss: 0.38377500422886823 Test Loss: 0.31953915528730015
Epoch: 1 Mean Train Loss: 0.4968723584018799 Train Accuracy: 86.40%
Iteration: 1000 Train Loss: 0.2581839410137612 Test Loss: 0.3000319568198122
Iteration: 1200 Train Loss: 0.36569183827688745 Test Loss: 0.28483428819911477
Iteration: 1400 Train Loss: 0.3810463944529671 Test Loss: 0.2742792119515289
Iteration: 1600 Train Loss: 0.26286587705052156 Test Loss: 0.2652769409733449
Iteration: 1800 Train Loss: 0.1418390759312137 Test Loss: 0.25637487613564
Epoch: 2 Mean Train Loss: 0.27796630698948344 Train Accuracy: 92.14%
Iteration: 2000 Train Loss: 0.4836013812226341 Test Loss: 0.2508604669724864
Iteration: 2200 Train Loss: 0.34252821276883105 Test Loss: 0.24503222890167545
Iteration: 2400 Train Loss: 0.17305584205810687 Test Loss: 0.23888153605272322
Iteration: 2600 Train Loss: 0.19642864450951097 Test Loss: 0.2333834979337388
Iteration: 2800 Train Loss: 0.17726214840620075 Test Loss: 0.22920658381222267
Epoch: 3 Mean Train Loss: 0.2333778050275643 Train Accuracy: 93.36%
Iteration: 3000 Train Loss: 0.23214888846065262 Test Loss: 0.22567199462524598
Iteration: 3200 Train Loss: 0.125420974617998 Test Loss: 0.22194009566807424
Iteration: 3400 Train Loss: 0.14195810856410718 Test Loss: 0.21862530153457538
Iteration: 3600 Train Loss: 0.14865782966160415 Test Loss: 0.21386258567883404
Epoch: 4 Mean Train Loss: 0.20496048741678383 Train Accuracy: 94.19%
Iteration: 3800 Train Loss: 0.05059963694513671 Test Loss: 0.2118098071835003
Iteration: 4000 Train Loss: 0.11050248907729933 Test Loss: 0.2078400212042656
Iteration: 4200 Train Loss: 0.1544499663427962 Test Loss: 0.20509320259355057
Iteration: 4400 Train Loss: 0.22439086515650514 Test Loss: 0.20326720364440642
Iteration: 4600 Train Loss: 0.24002446431870939 Test Loss: 0.19937372241341597
Epoch: 5 Mean Train Loss: 0.18331855102239386 Train Accuracy: 94.84%
Iteration: 4800 Train Loss: 0.20284791123023432 Test Loss: 0.19639091520135804
Iteration: 5000 Train Loss: 0.19795846730061975 Test Loss: 0.19436346082541436
Iteration: 5200 Train Loss: 0.3324270346916043 Test Loss: 0.19186820055497536
Iteration: 5400 Train Loss: 0.20269733570188211 Test Loss: 0.1907297478177586
Iteration: 5600 Train Loss: 0.19491783454235864 Test Loss: 0.18732245210280732
Epoch: 6 Mean Train Loss: 0.16558931138769706 Train Accuracy: 95.41%
Iteration: 5800 Train Loss: 0.1548026906900199 Test Loss: 0.18599554052271006
```

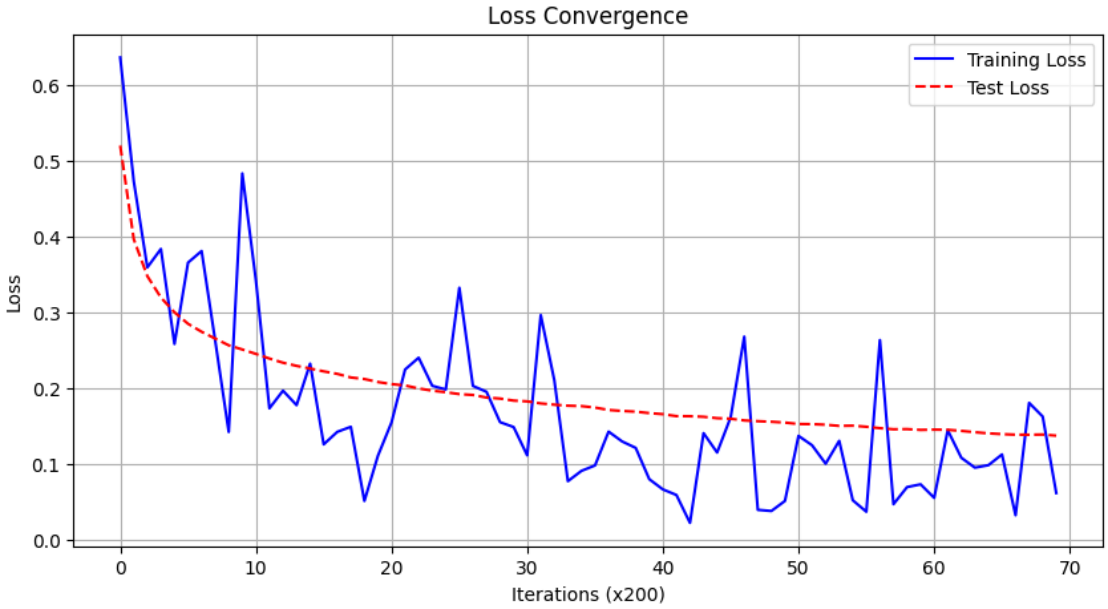
```
Iteration: 6000 Train Loss: 0.1483060298700162 Test Loss: 0.18329318095160652
Iteration: 6200 Train Loss: 0.11089209340264658 Test Loss: 0.18226637922129332
Iteration: 6400 Train Loss: 0.2962995331816804 Test Loss: 0.17955404583949885
Epoch: 7 Mean Train Loss: 0.15054250538780511 Train Accuracy: 95.91%
Iteration: 6600 Train Loss: 0.21021587821146925 Test Loss: 0.17789510509367384
Iteration: 6800 Train Loss: 0.07675170560693848 Test Loss: 0.1765309378065295
Iteration: 7000 Train Loss: 0.09038764384125646 Test Loss: 0.17599125140380473
Iteration: 7200 Train Loss: 0.09757273770423049 Test Loss: 0.17409744333564978
Iteration: 7400 Train Loss: 0.14228981507993446 Test Loss: 0.170887531356118
Epoch: 8 Mean Train Loss: 0.13759647049555468 Train Accuracy: 96.28%
Iteration: 7600 Train Loss: 0.1292600276628043 Test Loss: 0.1695984332194655
Iteration: 7800 Train Loss: 0.12075117838280863 Test Loss: 0.16865392154003658
Iteration: 8000 Train Loss: 0.07940335049231369 Test Loss: 0.16674123137275176
Iteration: 8200 Train Loss: 0.0658996528508571 Test Loss: 0.16534958239684955
Iteration: 8400 Train Loss: 0.05850358698965645 Test Loss: 0.1627368285653674
Epoch: 9 Mean Train Loss: 0.12582340243191747 Train Accuracy: 96.65%
Iteration: 8600 Train Loss: 0.021632642042885022 Test Loss: 0.16263514971200443
Iteration: 8800 Train Loss: 0.14029147220107568 Test Loss: 0.16187405435071103
Iteration: 9000 Train Loss: 0.11464178559273808 Test Loss: 0.16002554426759855
Iteration: 9200 Train Loss: 0.16132096376275112 Test Loss: 0.1593258893605119
Epoch: 10 Mean Train Loss: 0.11549077259656901 Train Accuracy: 96.99%
Iteration: 9400 Train Loss: 0.26783320811128264 Test Loss: 0.15704566309185763
Iteration: 9600 Train Loss: 0.038861823732231605 Test Loss: 0.15608168317117405
Iteration: 9800 Train Loss: 0.03732831791998042 Test Loss: 0.1553069264763415
Iteration: 10000 Train Loss: 0.050632648688527675 Test Loss: 0.15423712517578156
Iteration: 10200 Train Loss: 0.13665408442352187 Test Loss: 0.1523285059104463
Epoch: 11 Mean Train Loss: 0.10603239568465898 Train Accuracy: 97.25%
Iteration: 10400 Train Loss: 0.12422823525220772 Test Loss: 0.15215669253560318
Iteration: 10600 Train Loss: 0.09985503792163924 Test Loss: 0.15140358659947495
Iteration: 10800 Train Loss: 0.13010582111356728 Test Loss: 0.1498482353177954
Iteration: 11000 Train Loss: 0.051534573820569037 Test Loss: 0.1501200034886538
Iteration: 11200 Train Loss: 0.03632552946869619 Test Loss: 0.1487052834653117
Epoch: 12 Mean Train Loss: 0.09751093446588756 Train Accuracy: 97.51%
Iteration: 11400 Train Loss: 0.2633277947251573 Test Loss: 0.1468376547046113
Iteration: 11600 Train Loss: 0.046301044128777155 Test Loss: 0.1454128009164916
Iteration: 11800 Train Loss: 0.06894322564184596 Test Loss: 0.1456284976474635
Iteration: 12000 Train Loss: 0.07260383469411967 Test Loss: 0.14452267382729483
Epoch: 13 Mean Train Loss: 0.09006613381579684 Train Accuracy: 97.72%
Iteration: 12200 Train Loss: 0.054696507130865615 Test Loss: 0.14490421929452704
Iteration: 12400 Train Loss: 0.14413490923980354 Test Loss: 0.14467997525327964
Iteration: 12600 Train Loss: 0.10773929018926065 Test Loss: 0.14327070073052497
Iteration: 12800 Train Loss: 0.09463246725796738 Test Loss: 0.14147075233842132
Iteration: 13000 Train Loss: 0.09792796237723389 Test Loss: 0.13997329366471659
Epoch: 14 Mean Train Loss: 0.08304793749520625 Train Accuracy: 97.93%
Iteration: 13200 Train Loss: 0.11212676866403995 Test Loss: 0.1389453071776684
Iteration: 13400 Train Loss: 0.031633949695761235 Test Loss: 0.13824846524783257
Iteration: 13600 Train Loss: 0.18029146925317224 Test Loss: 0.13819636558045945
Iteration: 13800 Train Loss: 0.16216211853818824 Test Loss: 0.13843973635521586
```

```
Iteration: 14000 Train Loss: 0.061073914907180754 Test Loss: 0.13694147228550096
Epoch: 15 Mean Train Loss: 0.07656701580954922 Train Accuracy: 98.14%
Model trained!
```

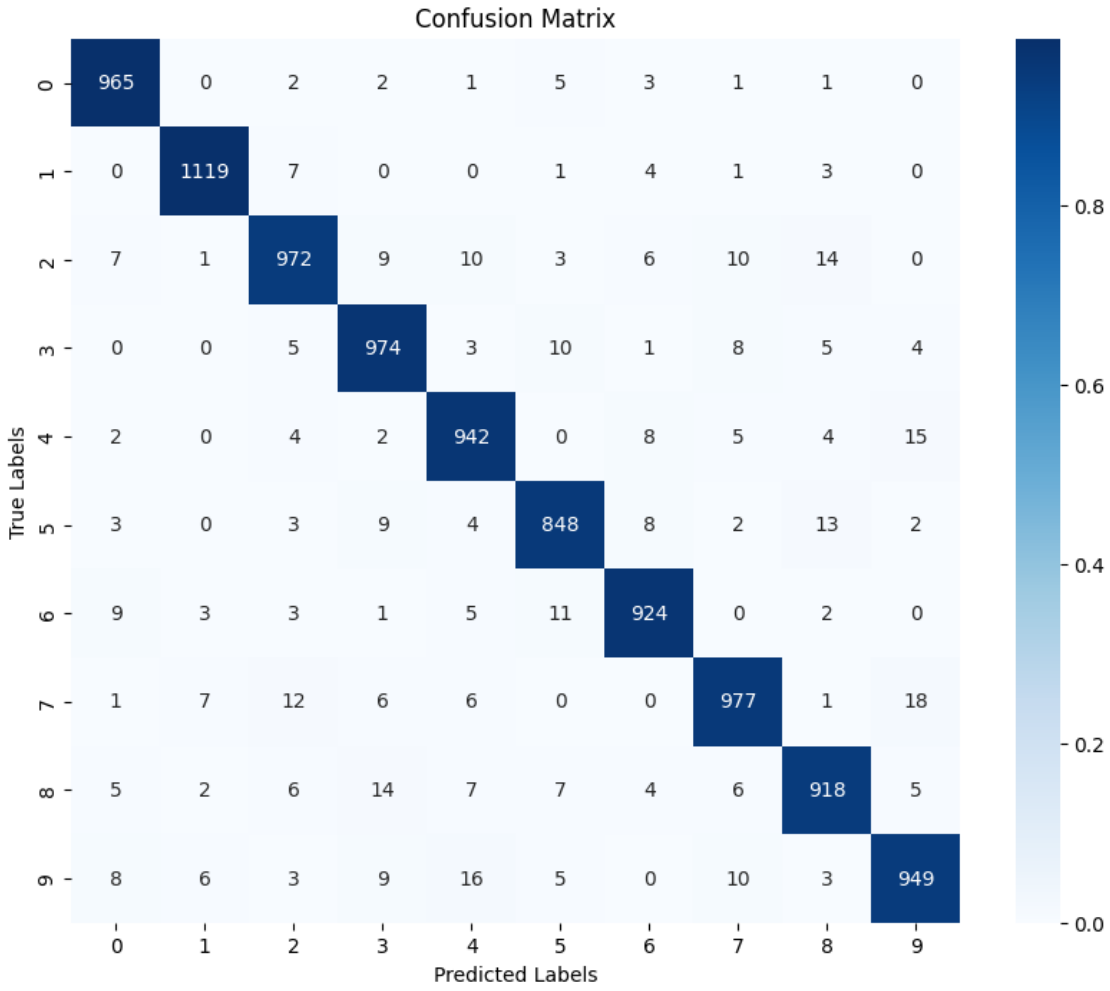
[ ]:
```
tanh_model.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.06980811358605145 Train accuracy: 98.35%
Test loss: 0.13803422907071627 Test accuracy: 95.88%

Confusion Matrix:
 [[ 965    0    2    2    1    5    3    1    1    0]
 [   0 1119    7    0    0    1    4    1    3    0]
 [   7    1  972    9   10    3    6   10   14    0]
 [   0    0    5  974    3   10    1    8    5    4]
 [   2    0    4    2  942    0    8    5    4   15]
 [   3    0    3    9    4  848    8    2   13    2]
 [   9    3    3    1    5   11  924    0    2    0]
 [   1    7   12    6    6    0    0  977    1   18]
 [   5    2    6   14    7    7    4    6  918    5]
 [   8    6    3    9   16    5    0   10    3  949]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       980
           1       0.98      0.99      0.98      1135
           2       0.96      0.94      0.95      1032
           3       0.95      0.96      0.96      1010
           4       0.95      0.96      0.95       982
           5       0.95      0.95      0.95       892
           6       0.96      0.96      0.96       958
           7       0.96      0.95      0.95      1028
           8       0.95      0.94      0.95       974
           9       0.96      0.94      0.95      1009

    accuracy                           0.96     10000
   macro avg       0.96      0.96      0.96     10000
weighted avg       0.96      0.96      0.96     10000
```

[ ]:
```
tanh_model.plot_losses()
```



[ ]:
```
tanh_model.plot_confusion_matrix(X_test, Y_test)
```



7.8 Performance with ReLU

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    ReLU(),
    FNNLayer(500, 250),
    ReLU(),
    FNNLayer(250, 100),
    ReLU(),
    FNNLayer(100, output_dim)
]

relu_model = NN(layers)
```

```
optimizer = SGD()
loss = LogLoss()
relu_model.compile(loss=loss, optimizer=optimizer)
relu_model.fit(X_train, Y_train, X_test, Y_test)
```

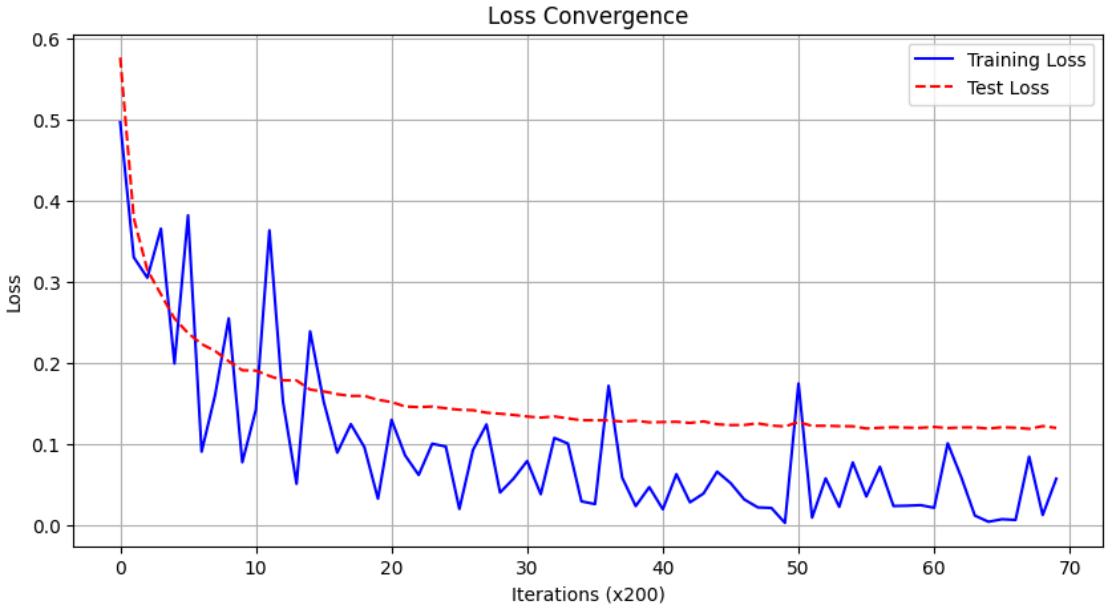
```
Iteration: 200 Train Loss: 0.49656280943967623 Test Loss: 0.5764013924102541
Iteration: 400 Train Loss: 0.33035301264252803 Test Loss: 0.3785817038090857
Iteration: 600 Train Loss: 0.3051923915988052 Test Loss: 0.31481117270753967
Iteration: 800 Train Loss: 0.36569342720154085 Test Loss: 0.28482584297799485
Epoch: 1 Mean Train Loss: 0.534597634309576 Train Accuracy: 85.52%
Iteration: 1000 Train Loss: 0.19961381905857606 Test Loss: 0.25501981022511705
Iteration: 1200 Train Loss: 0.3820573081166786 Test Loss: 0.23698910098709547
Iteration: 1400 Train Loss: 0.09108320797688393 Test Loss: 0.22359471283327687
Iteration: 1600 Train Loss: 0.16115695255330909 Test Loss: 0.21497579903385425
Iteration: 1800 Train Loss: 0.25517579316822603 Test Loss: 0.20243368440502613
Epoch: 2 Mean Train Loss: 0.20831057304886846 Train Accuracy: 93.92%
Iteration: 2000 Train Loss: 0.07802013358843116 Test Loss: 0.1912903684860585
Iteration: 2200 Train Loss: 0.14263492077655393 Test Loss: 0.19087514564516692
Iteration: 2400 Train Loss: 0.3635466750633388 Test Loss: 0.1842317192963076
Iteration: 2600 Train Loss: 0.15255761145562174 Test Loss: 0.17888159436306064
Iteration: 2800 Train Loss: 0.05154466305425118 Test Loss: 0.17874939540446003
Epoch: 3 Mean Train Loss: 0.1560254188132061 Train Accuracy: 95.52%
Iteration: 3000 Train Loss: 0.23917771055161874 Test Loss: 0.16751651227514955
Iteration: 3200 Train Loss: 0.1526223979986282 Test Loss: 0.16520802197122267
Iteration: 3400 Train Loss: 0.08983716207034259 Test Loss: 0.16186886705319814
Iteration: 3600 Train Loss: 0.12495970520119146 Test Loss: 0.1596943377494955
Epoch: 4 Mean Train Loss: 0.125871055886456 Train Accuracy: 96.40%
Iteration: 3800 Train Loss: 0.0966232050340688 Test Loss: 0.1597147497931818
Iteration: 4000 Train Loss: 0.03317708106383954 Test Loss: 0.15492549666396915
Iteration: 4200 Train Loss: 0.13032883467580025 Test Loss: 0.15219161017542063
Iteration: 4400 Train Loss: 0.08651108536679308 Test Loss: 0.14668496862909153
Iteration: 4600 Train Loss: 0.062384339080191774 Test Loss: 0.14595352573817977
Epoch: 5 Mean Train Loss: 0.10476029996102722 Train Accuracy: 97.04%
Iteration: 4800 Train Loss: 0.10085124114837687 Test Loss: 0.14661871356408873
Iteration: 5000 Train Loss: 0.09741480591728757 Test Loss: 0.14444584208586664
Iteration: 5200 Train Loss: 0.020542580658968065 Test Loss: 0.14267817713215816
Iteration: 5400 Train Loss: 0.09300730550826447 Test Loss: 0.1421489210299254
Iteration: 5600 Train Loss: 0.1246154604182413 Test Loss: 0.13903862658966518
Epoch: 6 Mean Train Loss: 0.08894479769337127 Train Accuracy: 97.49%
Iteration: 5800 Train Loss: 0.040784708030983996 Test Loss: 0.13784350799325493
Iteration: 6000 Train Loss: 0.058016441045274474 Test Loss: 0.13626756020292846
Iteration: 6200 Train Loss: 0.07964676228162251 Test Loss: 0.13433043922504884
Iteration: 6400 Train Loss: 0.038685251641619615 Test Loss: 0.13302197204097385
Epoch: 7 Mean Train Loss: 0.07554504843693062 Train Accuracy: 97.96%
Iteration: 6600 Train Loss: 0.1081101821724799 Test Loss: 0.13444130437132404
Iteration: 6800 Train Loss: 0.10105311061842284 Test Loss: 0.1321118837269661
Iteration: 7000 Train Loss: 0.029906680719256902 Test Loss: 0.1297171228650974
```

```
Iteration: 7200 Train Loss: 0.0266021053693517 Test Loss: 0.1297078501724202
Iteration: 7400 Train Loss: 0.17215255748286504 Test Loss: 0.12956730762414023
Epoch: 8 Mean Train Loss: 0.0648303222532081 Train Accuracy: 98.29%
Iteration: 7600 Train Loss: 0.05925443733709494 Test Loss: 0.12816815173527363
Iteration: 7800 Train Loss: 0.02410131419698893 Test Loss: 0.1292826527237837
Iteration: 8000 Train Loss: 0.047446468219187535 Test Loss: 0.12717188209880725
Iteration: 8200 Train Loss: 0.02016253763395732 Test Loss: 0.12757546566432773
Iteration: 8400 Train Loss: 0.06342380262045905 Test Loss: 0.12785209656230462
Epoch: 9 Mean Train Loss: 0.05683921799589621 Train Accuracy: 98.54%
Iteration: 8600 Train Loss: 0.028710250847327 Test Loss: 0.1264913824130348
Iteration: 8800 Train Loss: 0.039638133748515714 Test Loss: 0.1283193674246769
Iteration: 9000 Train Loss: 0.06651077610146718 Test Loss: 0.12478466285922868
Iteration: 9200 Train Loss: 0.052306294145170816 Test Loss: 0.12381555250883297
Epoch: 10 Mean Train Loss: 0.050185065923083506 Train Accuracy: 98.70%
Iteration: 9400 Train Loss: 0.03226583622820712 Test Loss: 0.12393558403631381
Iteration: 9600 Train Loss: 0.022478569290725822 Test Loss: 0.12586419827982886
Iteration: 9800 Train Loss: 0.021805160613761886 Test Loss: 0.12312776604825662
Iteration: 10000 Train Loss: 0.003420786826420169 Test Loss: 0.12221270784743028
Iteration: 10200 Train Loss: 0.17492383162706954 Test Loss: 0.12743374801002963
Epoch: 11 Mean Train Loss: 0.04408377659942238 Train Accuracy: 98.91%
Iteration: 10400 Train Loss: 0.009895204798787888 Test Loss: 0.12283864384340083
Iteration: 10600 Train Loss: 0.0582802869813892 Test Loss: 0.12296337994238421
Iteration: 10800 Train Loss: 0.023142649068454802 Test Loss: 0.12249754706736123
Iteration: 11000 Train Loss: 0.07784327430954086 Test Loss: 0.122343386648228
Iteration: 11200 Train Loss: 0.03590054070866386 Test Loss: 0.11956156515717203
Epoch: 12 Mean Train Loss: 0.038758332401909096 Train Accuracy: 99.06%
Iteration: 11400 Train Loss: 0.07248950923251474 Test Loss: 0.12045219014367714
Iteration: 11600 Train Loss: 0.024167224554964915 Test Loss: 0.12118756340604941
Iteration: 11800 Train Loss: 0.024525446281871737 Test Loss: 0.12064613168982849
Iteration: 12000 Train Loss: 0.02523536178128071 Test Loss: 0.12042793926812814
Epoch: 13 Mean Train Loss: 0.033456719418717196 Train Accuracy: 99.23%
Iteration: 12200 Train Loss: 0.02202231151772089 Test Loss: 0.12154827894621381
Iteration: 12400 Train Loss: 0.1014034196771875 Test Loss: 0.12013656217504347
Iteration: 12600 Train Loss: 0.05955811849729346 Test Loss: 0.12092088185244412
Iteration: 12800 Train Loss: 0.012392927649292356 Test Loss: 0.12097203404721255
Iteration: 13000 Train Loss: 0.004800677089055024 Test Loss: 0.11943524662321141
Epoch: 14 Mean Train Loss: 0.030037177258527288 Train Accuracy: 99.34%
Iteration: 13200 Train Loss: 0.007924112920292388 Test Loss: 0.1211737109456936
Iteration: 13400 Train Loss: 0.0070721189091624975 Test Loss:
0.12054187952430541
Iteration: 13600 Train Loss: 0.08498361152448304 Test Loss: 0.11918969741366935
Iteration: 13800 Train Loss: 0.013106226756359463 Test Loss: 0.1225737075450528
Iteration: 14000 Train Loss: 0.05780829528871902 Test Loss: 0.1203173144242065
Epoch: 15 Mean Train Loss: 0.0271355488016796 Train Accuracy: 99.43%
Model trained!
```

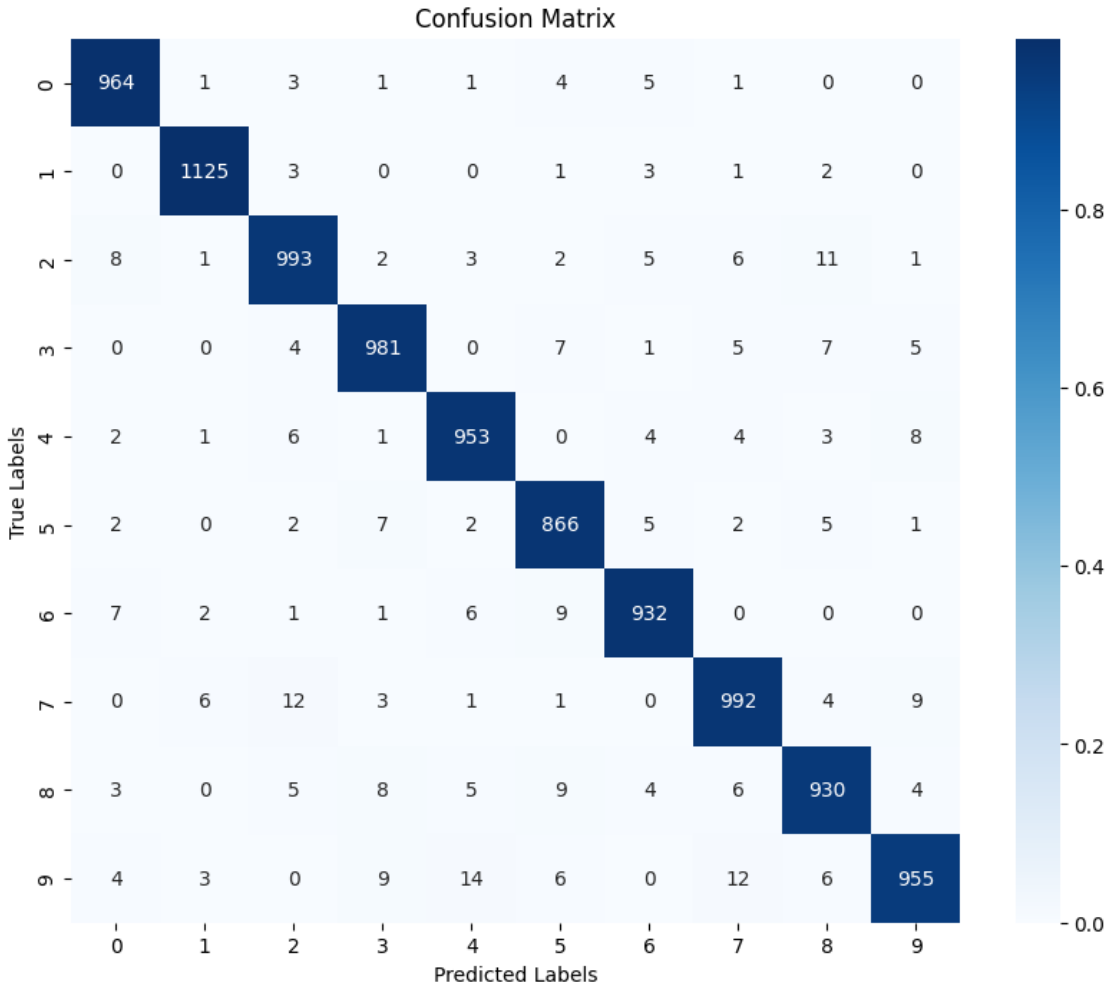
[ ]:
```
relu_model.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.022803324674423416 Train accuracy: 99.58%
Test loss: 0.12120565281551551 Test accuracy: 96.91%

Confusion Matrix:
 [[ 964    1    3    1    1    4    5    1    0    0]
 [   0 1125    3    0    0    1    3    1    2    0]
 [   8    1  993    2    3    2    5    6   11    1]
 [   0    0    4  981    0    7    1    5    7    5]
 [   2    1    6    1  953    0    4    4    3    8]
 [   2    0    2    7    2  866    5    2    5    1]
 [   7    2    1    1    6    9  932    0    0    0]
 [   0    6   12    3    1    1    0  992    4    9]
 [   3    0    5    8    5    9    4    6  930    4]
 [   4    3    0    9   14    6    0   12    6  955]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.96      0.96      1032
           3       0.97      0.97      0.97      1010
           4       0.97      0.97      0.97       982
           5       0.96      0.97      0.96       892
           6       0.97      0.97      0.97       958
           7       0.96      0.96      0.96      1028
           8       0.96      0.95      0.96       974
           9       0.97      0.95      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

[ ]:
```
relu_model.plot_losses()
```



[ ]:
```
relu_model.plot_confusion_matrix(X_test, Y_test)
```



7.9 Performance with Leaky ReLU

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    LeakyReLU(),
    FNNLayer(500, 250),
    LeakyReLU(),
    FNNLayer(250, 100),
    LeakyReLU(),
    FNNLayer(100, output_dim)
]

lrelu_model = NN(layers)
```

```
optimizer = SGD()
loss = LogLoss()
lrelu_model.compile(loss=loss, optimizer=optimizer)
lrelu_model.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.5837200255888777 Test Loss: 0.5714556908020015
Iteration: 400 Train Loss: 0.532756667308845 Test Loss: 0.3791947861155695
Iteration: 600 Train Loss: 0.48849123043183723 Test Loss: 0.3213814851909864
Iteration: 800 Train Loss: 0.15478070904859037 Test Loss: 0.27657555910327714
Epoch: 1 Mean Train Loss: 0.5275257893492852 Train Accuracy: 85.74%
Iteration: 1000 Train Loss: 0.28693668118054955 Test Loss: 0.26280551556033505
Iteration: 1200 Train Loss: 0.17584558065311043 Test Loss: 0.2409190163417514
Iteration: 1400 Train Loss: 0.2798806351774464 Test Loss: 0.22510041261240019
Iteration: 1600 Train Loss: 0.5242409291215124 Test Loss: 0.21618383171032865
Iteration: 1800 Train Loss: 0.10196507305350727 Test Loss: 0.20567736847263202
Epoch: 2 Mean Train Loss: 0.21577161565623157 Train Accuracy: 93.71%
Iteration: 2000 Train Loss: 0.11032247228589825 Test Loss: 0.19925980787120282
Iteration: 2200 Train Loss: 0.09947097742695399 Test Loss: 0.19154027127945855
Iteration: 2400 Train Loss: 0.2840196792884248 Test Loss: 0.18133162226933314
Iteration: 2600 Train Loss: 0.15248953705742274 Test Loss: 0.17549765305657425
Iteration: 2800 Train Loss: 0.23634086099951188 Test Loss: 0.17255185698147885
Epoch: 3 Mean Train Loss: 0.16427902766263425 Train Accuracy: 95.21%
Iteration: 3000 Train Loss: 0.14884011016163728 Test Loss: 0.168731343963206
Iteration: 3200 Train Loss: 0.20923238414824918 Test Loss: 0.1654660524290165
Iteration: 3400 Train Loss: 0.18553147869645786 Test Loss: 0.16424243912652373
Iteration: 3600 Train Loss: 0.03257458501401755 Test Loss: 0.15915136315201836
Epoch: 4 Mean Train Loss: 0.13382871347322065 Train Accuracy: 96.09%
Iteration: 3800 Train Loss: 0.07295811791988592 Test Loss: 0.1554253778857847
Iteration: 4000 Train Loss: 0.15164406836487954 Test Loss: 0.15235194671766128
Iteration: 4200 Train Loss: 0.11921936507763892 Test Loss: 0.15302808024617268
Iteration: 4400 Train Loss: 0.07461390288095818 Test Loss: 0.14833258190218934
Iteration: 4600 Train Loss: 0.12451090467632792 Test Loss: 0.14661318098610346
Epoch: 5 Mean Train Loss: 0.1126943381223989 Train Accuracy: 96.76%
Iteration: 4800 Train Loss: 0.08065647828487157 Test Loss: 0.14667916503445563
Iteration: 5000 Train Loss: 0.13043055343896354 Test Loss: 0.14039858035047018
Iteration: 5200 Train Loss: 0.08448317634249401 Test Loss: 0.13848295274568703
Iteration: 5400 Train Loss: 0.08050436185345711 Test Loss: 0.1339865227126377
Iteration: 5600 Train Loss: 0.19123999596652325 Test Loss: 0.13480762912171176
Epoch: 6 Mean Train Loss: 0.09632640291610814 Train Accuracy: 97.25%
Iteration: 5800 Train Loss: 0.11062901835252889 Test Loss: 0.13304593114230787
Iteration: 6000 Train Loss: 0.05734142538886256 Test Loss: 0.13311792795785038
Iteration: 6200 Train Loss: 0.115134904608799 Test Loss: 0.13370402212231347
Iteration: 6400 Train Loss: 0.07119629176787953 Test Loss: 0.12962657771793157
Epoch: 7 Mean Train Loss: 0.084123560726736 Train Accuracy: 97.63%
Iteration: 6600 Train Loss: 0.13092424219865434 Test Loss: 0.1295054164186612
Iteration: 6800 Train Loss: 0.04914603177840091 Test Loss: 0.1284306989020743
Iteration: 7000 Train Loss: 0.0767147292814836 Test Loss: 0.12679696419170788
```

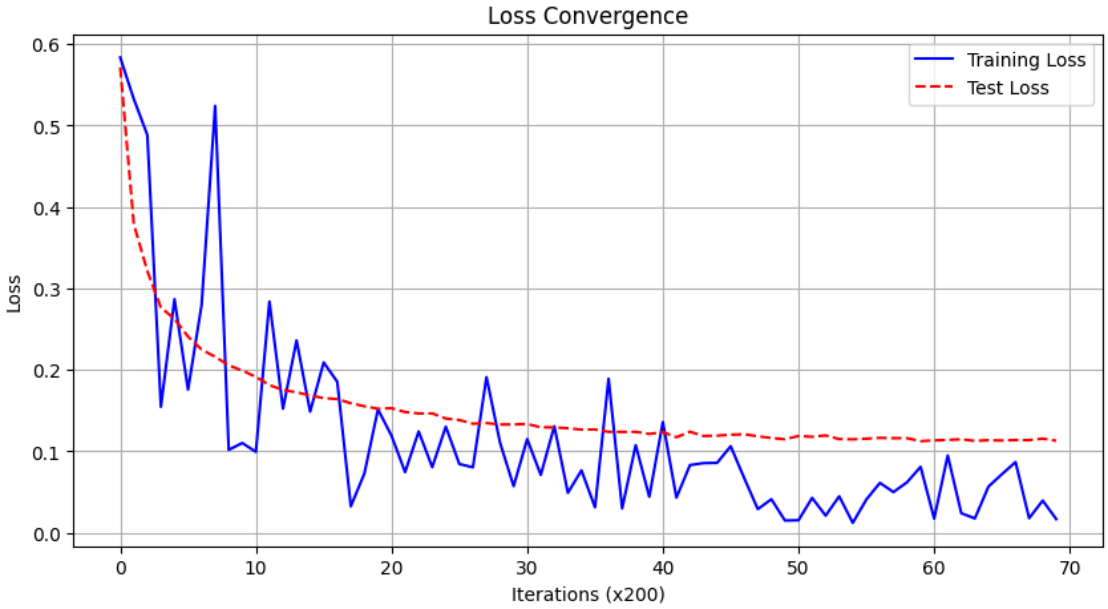
```
Iteration: 7200 Train Loss: 0.031367146767350224 Test Loss: 0.1269331465193947
Iteration: 7400 Train Loss: 0.18926645466725914 Test Loss: 0.12401742605687467
Epoch: 8 Mean Train Loss: 0.07243556683394242 Train Accuracy: 97.99%
Iteration: 7600 Train Loss: 0.030006289123596278 Test Loss: 0.12393839261800142
Iteration: 7800 Train Loss: 0.10774494138618512 Test Loss: 0.12397666420282275
Iteration: 8000 Train Loss: 0.04438613950172233 Test Loss: 0.12146116292832189
Iteration: 8200 Train Loss: 0.1360932657865764 Test Loss: 0.12373324601228618
Iteration: 8400 Train Loss: 0.04324190681855274 Test Loss: 0.11740046160129265
Epoch: 9 Mean Train Loss: 0.06416368885931042 Train Accuracy: 98.28%
Iteration: 8600 Train Loss: 0.08321628762362791 Test Loss: 0.12414694656404485
Iteration: 8800 Train Loss: 0.08565465402710051 Test Loss: 0.11889125017827143
Iteration: 9000 Train Loss: 0.08595746680778135 Test Loss: 0.11936436414229153
Iteration: 9200 Train Loss: 0.10641246645914278 Test Loss: 0.12053646614794031
Epoch: 10 Mean Train Loss: 0.056348547764796884 Train Accuracy: 98.52%
Iteration: 9400 Train Loss: 0.06715016914571816 Test Loss: 0.12108477005059798
Iteration: 9600 Train Loss: 0.029164168200154484 Test Loss: 0.11857321540054618
Iteration: 9800 Train Loss: 0.04130953495916201 Test Loss: 0.11652914284230191
Iteration: 10000 Train Loss: 0.015153672840121701 Test Loss: 0.11490761347590081
Iteration: 10200 Train Loss: 0.01550565151841421 Test Loss: 0.11892120430539772
Epoch: 11 Mean Train Loss: 0.05027502948579009 Train Accuracy: 98.69%
Iteration: 10400 Train Loss: 0.04295839726332554 Test Loss: 0.1179397317363899
Iteration: 10600 Train Loss: 0.02114449530148314 Test Loss: 0.11960777294915674
Iteration: 10800 Train Loss: 0.04490771600112968 Test Loss: 0.11515973865554313
Iteration: 11000 Train Loss: 0.01226628863738936 Test Loss: 0.11485201001280305
Iteration: 11200 Train Loss: 0.04095489204676321 Test Loss: 0.11565653538698385
Epoch: 12 Mean Train Loss: 0.0446003504546984 Train Accuracy: 98.87%
Iteration: 11400 Train Loss: 0.06137316116549676 Test Loss: 0.11665578947827943
Iteration: 11600 Train Loss: 0.05009154617996708 Test Loss: 0.11631356088309282
Iteration: 11800 Train Loss: 0.06214928394287729 Test Loss: 0.11625437183215412
Iteration: 12000 Train Loss: 0.08107825170363052 Test Loss: 0.11258925286031102
Epoch: 13 Mean Train Loss: 0.04046094336744198 Train Accuracy: 99.03%
Iteration: 12200 Train Loss: 0.0175606349797471 Test Loss: 0.11353181023861773
Iteration: 12400 Train Loss: 0.0949881497776195 Test Loss: 0.11412183373357804
Iteration: 12600 Train Loss: 0.024127601022943028 Test Loss: 0.11489526436753507
Iteration: 12800 Train Loss: 0.01770995855574384 Test Loss: 0.11291383043215303
Iteration: 13000 Train Loss: 0.05676931472078629 Test Loss: 0.11377451082554023
Epoch: 14 Mean Train Loss: 0.035026858950087635 Train Accuracy: 99.18%
Iteration: 13200 Train Loss: 0.07214130905463824 Test Loss: 0.1133565229622948
Iteration: 13400 Train Loss: 0.08692691314165159 Test Loss: 0.11394319964323056
Iteration: 13600 Train Loss: 0.017942879634995888 Test Loss: 0.11385961264574702
Iteration: 13800 Train Loss: 0.03966658168694953 Test Loss: 0.11569785717168922
Iteration: 14000 Train Loss: 0.016842336860546647 Test Loss: 0.1130868761051794
Epoch: 15 Mean Train Loss: 0.03179548701708007 Train Accuracy: 99.26%
Model trained!
```

[ ]:
```
lrelu_model.evaluate(X_train, Y_train, X_test, Y_test)
```

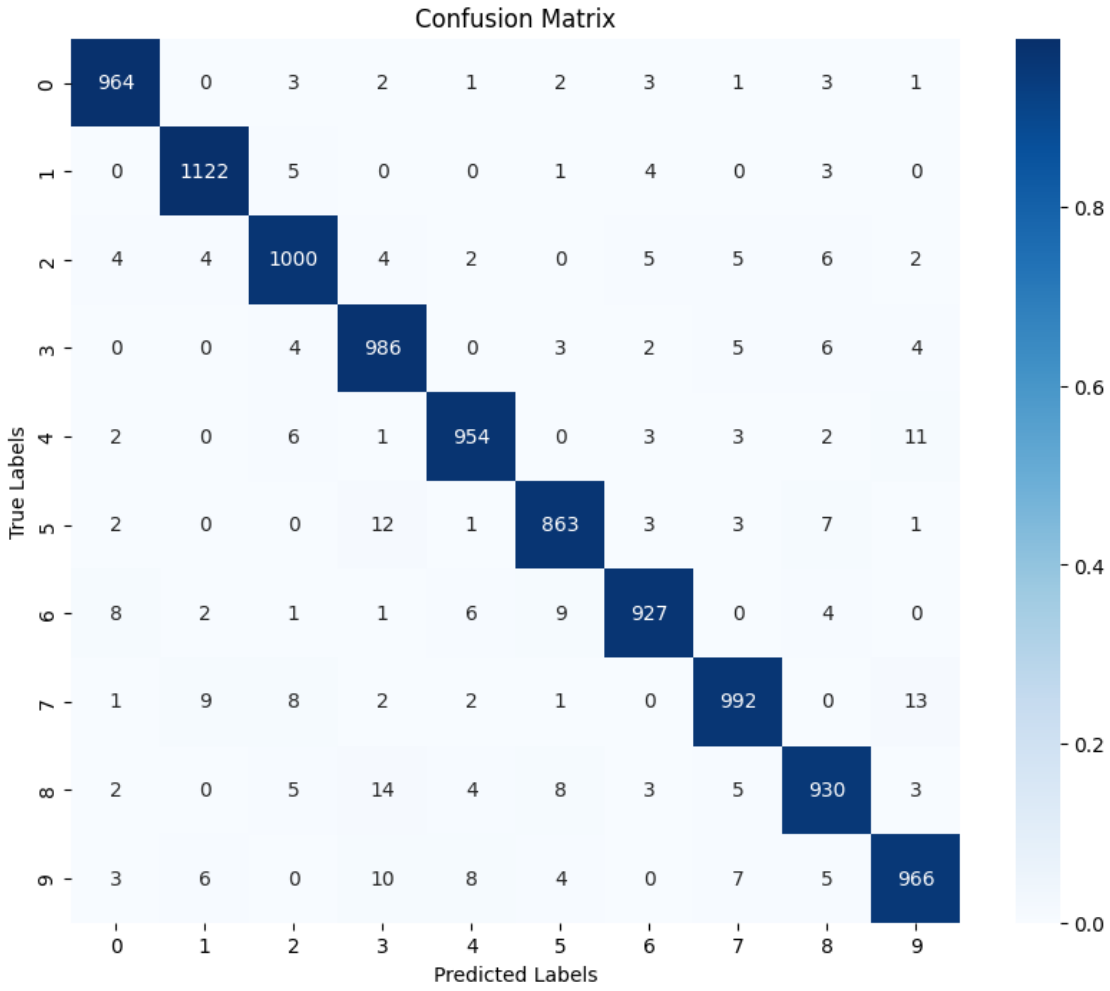
```
Train loss: 0.026928010522560643 Train accuracy: 99.46%
```

```
Test loss: 0.11234078461906374 Test accuracy: 97.04%

Confusion Matrix:
 [[ 964    0    3    2    1    2    3    1    3    1]
 [   0 1122    5    0    0    1    4    0    3    0]
 [   4    4 1000    4    2    0    5    5    6    2]
 [   0    0    4  986    0    3    2    5    6    4]
 [   2    0    6    1  954    0    3    3    2   11]
 [   2    0    0   12    1  863    3    3    7    1]
 [   8    2    1    1    6    9  927    0    4    0]
 [   1    9    8    2    2    1    0  992    0   13]
 [   2    0    5   14    4    8    3    5  930    3]
 [   3    6    0   10    8    4    0    7    5  966]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       980
           1       0.98      0.99      0.99      1135
           2       0.97      0.97      0.97      1032
           3       0.96      0.98      0.97      1010
           4       0.98      0.97      0.97       982
           5       0.97      0.97      0.97       892
           6       0.98      0.97      0.97       958
           7       0.97      0.96      0.97      1028
           8       0.96      0.95      0.96       974
           9       0.97      0.96      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

[ ]:
```
lrelu_model.plot_losses()
```

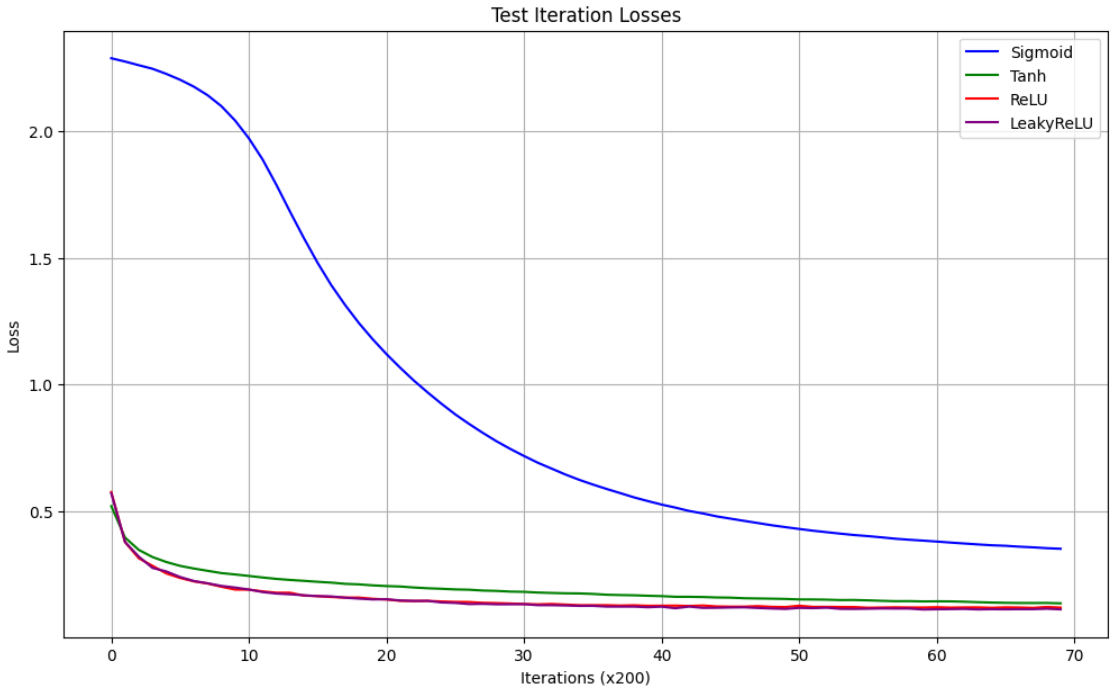


[ ]:
```
lrelu_model.plot_confusion_matrix(X_test, Y_test)
```



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

def plot_models_losses(models, labels, title, attribute):
    plt.figure(figsize=(12, 7))

    colors = ['blue', 'green', 'red', 'purple']

    for i, model in enumerate(models):
        losses = getattr(model, attribute)
        plt.plot(losses, label=labels[i], color=colors[i])

    plt.xlabel('Iterations (x200)')
    plt.ylabel('Loss')
    plt.title(title)
    plt.legend()
    plt.grid(True)
```

```
    plt.show()

model1 = model
model2 = tanh_model
model3 = relu_model
model4 = lrelu_model

models = [model1, model2, model3, model4]
labels = ['Sigmoid', 'Tanh', 'ReLU', 'LeakyReLU']

# Plots training iteration losses
plot_models_losses(models, labels, 'Training Iteration Losses',␣
 ↪'iteration_losses')

# Plots test iteration losses
plot_models_losses(models, labels, 'Test Iteration Losses',␣
 ↪'iteration_test_losses')
```



7.10 Observations

Activation	Test Accuracy	Train Accuracy	Train Loss	Test Loss
Sigmoid	90.09%	90.09%	0.3519	0.3523
Tanh	95.88%	98.35%	0.0698	0.1380
ReLU	96.91%	99.58%	0.0228	0.1212
Leaky ReLU	97.04%	99.46%	0.0269	0.1123

The values in this table might change based on the run. The observations might be affected slightly as a result.

ReLU and Leaky ReLU outperform all the other activation functions.

- ReLU is expected to perform better than Sigmoid and Tanh functions because it doesn't squash output values to a narrow range, so gradients aren't as susceptible to vanishing.

- ReLU doesn't saturate for positive values. This means it doesn't suffer as much from the vanishing gradient problem. Gradients flow easily during backpropagation, which can result in faster convergence.
- Tanh squashes its outputs to be in the range [-1, 1], which can be more desirable than [0, 1] in some scenarios as it centers the output around 0.
- For inputs close to 0, tanh has a steeper gradient compared to the sigmoid function, which can result in faster learning.
- I wanted to experiment with Leaky ReLU as sometimes it is possible that the dying ReLU problem comes up, in which neurons get stuck during training and stop updating their weights.

his happens when a large gradient flows through a ReLU neuron, updating the weights in such a way that the neuron will always output 0.

- We don't alywas see Leaky ReLU outperform ReLU though. This may be because the dying ReLU issue isn't dominant here for such a simple dataset. My choice of the hyperparameter alpha in Leaky ReLU might just be luckily ideal for this setting as well. Plus, I am not sure if I am using the term correctly, but leaky ReLU might be offering a regularization effect: if some neurons become inactive, it could force other parts of the network to adapt and generalize better.
- Weight initialization can significantly influence the early stages of training by determining the initial magnitude of neuron activations and gradients. If the activations are too large or too small, it can either slow down learning or cause it to diverge altogether.

- Even though we are comparing the performance of these activation functions on the same architecture and hyperparameters (learning rate, batch size), the performance is definitely subject to the weight initialization which is different for all the 4 models as weights are initialized separately for each model. Our hope is that with enough epochs, each model gets a chance to overcome the shortcomings of their initializations, but still the performance does depend on it.
- For example, if weights are initialized too small, especially for sigmoid or tanh, activations can get squashed into the saturated regions, leading to the vanishing gradient problem. As a result, learning will be very slow.
- If we set 95% test accuracy as a benchmark for acceotable performance, both ReLU and leaky ReLU achieve that by the 3rd epoch of training. TanH is also commendable as it gets there by its 6th epoch. However, even 15 epochs aren't enough for the sigmoid activation function. Either it leads to slow convergence or its weight in this run were initialized very poorly.
- Among the 3 options given in the assignment, ReLU performs the best, so we'll consider that to be our activation function in the baseline going ahead.

8 Performance of the best baseline model with other optimizers

8.1 Momentum

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    ReLU(),
    FNNLayer(500, 250),
    ReLU(),
    FNNLayer(250, 100),
    ReLU(),
    FNNLayer(100, output_dim)
]

best_model_momentum = NN(layers)
optimizer = Momentum()
```

```
loss = LogLoss()
best_model_momentum.compile(loss=loss, optimizer=optimizer)
best_model_momentum.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.2831846500824217 Test Loss: 0.2739371801716733
Iteration: 400 Train Loss: 0.3509966008712886 Test Loss: 0.23237908831807264
Iteration: 600 Train Loss: 0.1848757538311216 Test Loss: 0.1921458697739969
Iteration: 800 Train Loss: 0.21470706763327163 Test Loss: 0.16444030258385292
Epoch: 1 Mean Train Loss: 0.27697266153109423 Train Accuracy: 92.21%
Iteration: 1000 Train Loss: 0.043229211169753555 Test Loss: 0.1368287007421211
Iteration: 1200 Train Loss: 0.09366322273931255 Test Loss: 0.14043229370694368
Iteration: 1400 Train Loss: 0.04675467104091051 Test Loss: 0.13651016261821336
Iteration: 1600 Train Loss: 0.06923949332294382 Test Loss: 0.1171638699692277
Iteration: 1800 Train Loss: 0.017326517277805507 Test Loss: 0.11756689050733475
Epoch: 2 Mean Train Loss: 0.10386960222653042 Train Accuracy: 97.07%
Iteration: 2000 Train Loss: 0.01064660658408851 Test Loss: 0.11263574886885167
Iteration: 2200 Train Loss: 0.03295209130354141 Test Loss: 0.11511347565685265
Iteration: 2400 Train Loss: 0.15560778821391505 Test Loss: 0.11152189463772884
Iteration: 2600 Train Loss: 0.10348763474882947 Test Loss: 0.11474208273913644
Iteration: 2800 Train Loss: 0.05184727339357478 Test Loss: 0.12181106965046606
Epoch: 3 Mean Train Loss: 0.056098694746021145 Train Accuracy: 98.28%
Iteration: 3000 Train Loss: 0.005947833936711324 Test Loss: 0.1122231122879263
Iteration: 3200 Train Loss: 0.045919718584183915 Test Loss: 0.11027161058393142
Iteration: 3400 Train Loss: 0.028262573944200266 Test Loss: 0.11192580617310512
Iteration: 3600 Train Loss: 0.009727932033991235 Test Loss: 0.11012448581155813
Epoch: 4 Mean Train Loss: 0.03366935478380062 Train Accuracy: 99.02%
Iteration: 3800 Train Loss: 0.006723901995543362 Test Loss: 0.10549393891394221
Iteration: 4000 Train Loss: 0.010753762358672377 Test Loss: 0.11700875170275363
Iteration: 4200 Train Loss: 0.0023939059809860493 Test Loss: 0.11732980448754024
Iteration: 4400 Train Loss: 0.009823463086880356 Test Loss: 0.1135027135637596
Iteration: 4600 Train Loss: 0.013611056506457568 Test Loss: 0.10163310774294038
Epoch: 5 Mean Train Loss: 0.02196195995903989 Train Accuracy: 99.43%
Iteration: 4800 Train Loss: 0.02080609413999867 Test Loss: 0.10761284160724711
Iteration: 5000 Train Loss: 0.015886846185504254 Test Loss: 0.10406157540129113
Iteration: 5200 Train Loss: 0.006245822556339889 Test Loss: 0.10843144252866575
Iteration: 5400 Train Loss: 0.08871690982650524 Test Loss: 0.12193552677866831
Iteration: 5600 Train Loss: 0.019855856227121695 Test Loss: 0.10838855122100473
Epoch: 6 Mean Train Loss: 0.013167409059730111 Train Accuracy: 99.69%
Iteration: 5800 Train Loss: 0.010857057302766746 Test Loss: 0.10714027291660443
Iteration: 6000 Train Loss: 0.0029026248812316696 Test Loss: 0.1070076570653262
Iteration: 6200 Train Loss: 0.010851347195237886 Test Loss: 0.1162653229209535
Iteration: 6400 Train Loss: 0.0020556980135874527 Test Loss: 0.11633857113356853
Epoch: 7 Mean Train Loss: 0.007537798161754629 Train Accuracy: 99.84%
Iteration: 6600 Train Loss: 0.0027724358675314618 Test Loss: 0.11376335554564744
Iteration: 6800 Train Loss: 0.0015900171302593222 Test Loss: 0.11353104470791947
Iteration: 7000 Train Loss: 0.009183483465667393 Test Loss: 0.12043536688692873
Iteration: 7200 Train Loss: 0.004482088206333248 Test Loss: 0.11620226078766255
```

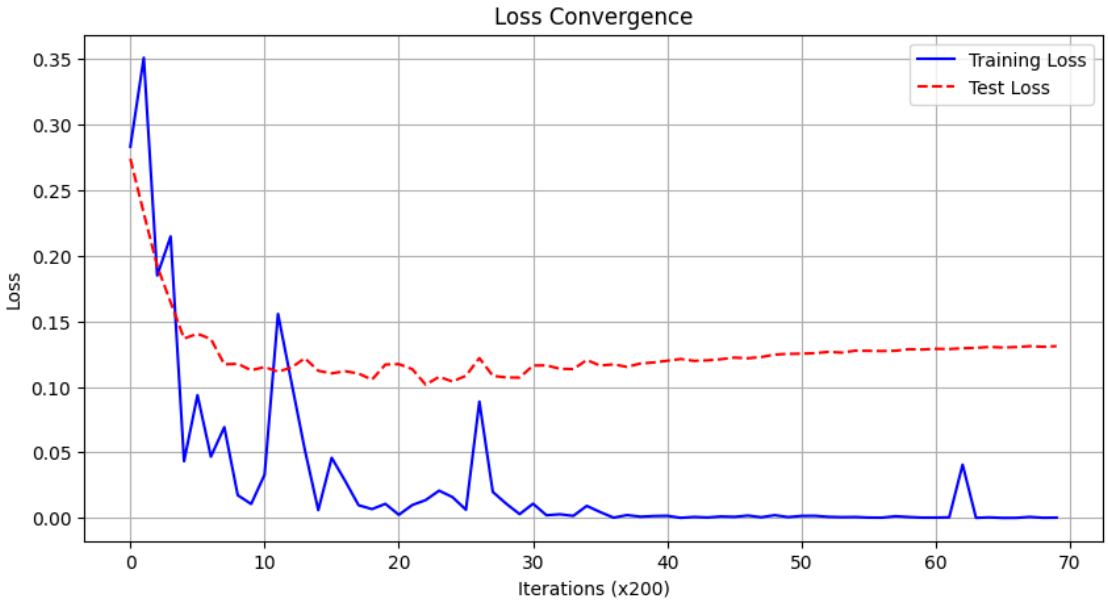
Iteration: 7400 Train Loss: 0.00022764946472536977 Test Loss:
0.11713659105901485
Epoch: 8 Mean Train Loss: 0.005204865248852933 Train Accuracy: 99.92%
Iteration: 7600 Train Loss: 0.00217085744024046 Test Loss: 0.11520108573367414
Iteration: 7800 Train Loss: 0.0009740614097759922 Test Loss: 0.117783593679081
Iteration: 8000 Train Loss: 0.0014676913567336458 Test Loss: 0.11859342759201263
Iteration: 8200 Train Loss: 0.0016320256399619374 Test Loss: 0.1198658327188951
Iteration: 8400 Train Loss: 8.504661667495418e-05 Test Loss: 0.12124191438953032
Epoch: 9 Mean Train Loss: 0.002545963236524152 Train Accuracy: 99.97%
Iteration: 8600 Train Loss: 0.0007625937692346189 Test Loss: 0.11979015080488989
Iteration: 8800 Train Loss: 0.0003532664045775895 Test Loss: 0.120192744624782
Iteration: 9000 Train Loss: 0.0011306484729433255 Test Loss: 0.12105830806757685
Iteration: 9200 Train Loss: 0.0008399289373668776 Test Loss: 0.12229104717589255
Epoch: 10 Mean Train Loss: 0.0012455947523710908 Train Accuracy: 99.99%
Iteration: 9400 Train Loss: 0.0018012937563965806 Test Loss: 0.12176969602164918
Iteration: 9600 Train Loss: 0.0005306194225950672 Test Loss: 0.12270182836877605
Iteration: 9800 Train Loss: 0.002089420364355117 Test Loss: 0.12462265699065905
Iteration: 10000 Train Loss: 0.000657320653290865 Test Loss: 0.12510853479826248
Iteration: 10200 Train Loss: 0.0015491773329133404 Test Loss:
0.12527679256933272
Epoch: 11 Mean Train Loss: 0.0009258835155305051 Train Accuracy: 100.00%
Iteration: 10400 Train Loss: 0.0016350489946288988 Test Loss:
0.12565601513189045
Iteration: 10600 Train Loss: 0.0008542434164809265 Test Loss: 0.1266818226290943
Iteration: 10800 Train Loss: 0.0005921383651625576 Test Loss:
0.12609149903383907
Iteration: 11000 Train Loss: 0.0007152668671020851 Test Loss: 0.1275799712084096
Iteration: 11200 Train Loss: 0.00028957289148976323 Test Loss:
0.12742393565765373
Epoch: 12 Mean Train Loss: 0.000776569070520096 Train Accuracy: 100.00%
Iteration: 11400 Train Loss: 0.00021926983080805608 Test Loss:
0.1272682462211778
Iteration: 11600 Train Loss: 0.0013218618437845 99 Test Loss: 0.1274991684237617
Iteration: 11800 Train Loss: 0.000690557718136234 Test Loss: 0.12869574303603076
Iteration: 12000 Train Loss: 0.00024000816994821424 Test Loss:
0.1284377856347252
Epoch: 13 Mean Train Loss: 0.0006187292423175501 Train Accuracy: 100.00%
Iteration: 12200 Train Loss: 0.0002672475722655464 Test Loss:
0.12898186730317973
Iteration: 12400 Train Loss: 0.00044208368088372415 Test Loss:
0.12881025090242945
Iteration: 12600 Train Loss: 0.04064922352275301 Test Loss: 0.12941282171691545
Iteration: 12800 Train Loss: 5.440584339268951e-05 Test Loss:
0.12969185378662984
Iteration: 13000 Train Loss: 0.00043619282308691266 Test Loss:
0.13043461951634078
Epoch: 14 Mean Train Loss: 0.000538246478577447 Train Accuracy: 100.00%
Iteration: 13200 Train Loss: 3.836634254027277e-06 Test Loss: 0.1299639794267162

```
Iteration: 13400 Train Loss: 5.394686904212013e-05 Test Loss:
0.13034670689400107
Iteration: 13600 Train Loss: 0.0007892235610350549 Test Loss:
0.13100945170288375
Iteration: 13800 Train Loss: 0.00011810122597159337 Test Loss:
0.1305442407148455
Iteration: 14000 Train Loss: 0.0002453446014000969 Test Loss:
0.13102144146127856
Epoch: 15 Mean Train Loss: 0.00048667145739244487 Train Accuracy: 100.00%
Model trained!
```

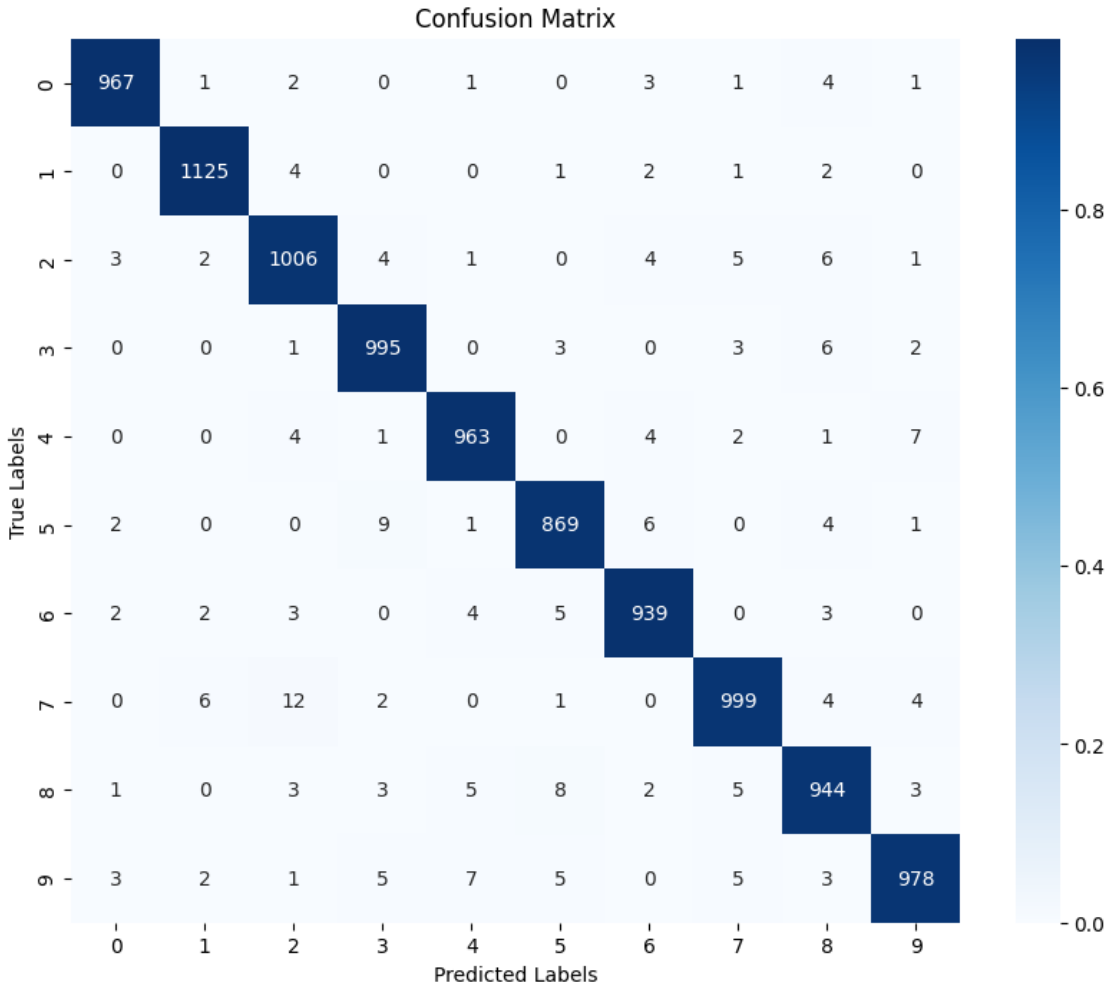
[ ]:
```
best_model_momentum.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.00037430192749715054 Train accuracy: 100.00%
Test loss: 0.13207718167723873 Test accuracy: 97.85%

Confusion Matrix:
 [[ 967    1    2    0    1    0    3    1    4    1]
 [   0 1125    4    0    0    1    2    1    2    0]
 [   3    2 1006    4    1    0    4    5    6    1]
 [   0    0    1  995    0    3    0    3    6    2]
 [   0    0    4    1  963    0    4    2    1    7]
 [   2    0    0    9    1  869    6    0    4    1]
 [   2    2    3    0    4    5  939    0    3    0]
 [   0    6   12    2    0    1    0  999    4    4]
 [   1    0    3    3    5    8    2    5  944    3]
 [   3    2    1    5    7    5    0    5    3  978]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.97      0.97      1032
           3       0.98      0.99      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.97      0.97      0.97       892
           6       0.98      0.98      0.98       958
           7       0.98      0.97      0.98      1028
           8       0.97      0.97      0.97       974
           9       0.98      0.97      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

[ ]:
```
best_model_momentum.plot_losses()
```



[ ]:
```
best_model_momentum.plot_confusion_matrix(X_test, Y_test)
```



8.2 RMSProp

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    ReLU(),
    FNNLayer(500, 250),
    ReLU(),
    FNNLayer(250, 100),
    ReLU(),
    FNNLayer(100, output_dim)
]

best_model_rmsprop = NN(layers)
```

```
optimizer = RMSProp()
loss = LogLoss()
best_model_rmsprop.compile(loss=loss, optimizer=optimizer)
best_model_rmsprop.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.8611892463555653 Test Loss: 0.7039133869595239
Iteration: 400 Train Loss: 0.35707097285046063 Test Loss: 0.7108105935671143
Iteration: 600 Train Loss: 0.4380136647892557 Test Loss: 0.486649431615153
Iteration: 800 Train Loss: 0.7963917784223166 Test Loss: 0.5992141536867432
Epoch: 1 Mean Train Loss: 0.6623115042604161 Train Accuracy: 86.23%
Iteration: 1000 Train Loss: 0.22313052213516094 Test Loss: 0.8721529644762427
Iteration: 1200 Train Loss: 0.4857553856685233 Test Loss: 0.43115069396131156
Iteration: 1400 Train Loss: 0.39971691488116234 Test Loss: 0.4749583881452514
Iteration: 1600 Train Loss: 0.4414229664598167 Test Loss: 0.7655896421174432
Iteration: 1800 Train Loss: 0.8711488959623745 Test Loss: 0.5689467981365716
Epoch: 2 Mean Train Loss: 0.572174427857848 Train Accuracy: 86.96%
Iteration: 2000 Train Loss: 0.39883172775303977 Test Loss: 0.6084832773224301
Iteration: 2200 Train Loss: 0.4392633930447103 Test Loss: 0.5958015614513833
Iteration: 2400 Train Loss: 0.8897832977748494 Test Loss: 0.6695856647992021
Iteration: 2600 Train Loss: 1.913475515837698 Test Loss: 0.6029034462448679
Iteration: 2800 Train Loss: 0.45825831737107403 Test Loss: 0.6462468918690627
Epoch: 3 Mean Train Loss: 0.6732437760694563 Train Accuracy: 83.37%
Iteration: 3000 Train Loss: 1.2965023769745878 Test Loss: 0.6195842733889121
Iteration: 3200 Train Loss: 1.0006762050480935 Test Loss: 0.9258330636814636
Iteration: 3400 Train Loss: 0.7259264707697258 Test Loss: 0.6671539252989899
Iteration: 3600 Train Loss: 0.6970979202882857 Test Loss: 0.6835057174135428
Epoch: 4 Mean Train Loss: 0.7297604737498485 Train Accuracy: 80.71%
Iteration: 3800 Train Loss: 0.8224515818772209 Test Loss: 0.7684239285237802
Iteration: 4000 Train Loss: 0.7092668675649894 Test Loss: 0.7058044677892925
Iteration: 4200 Train Loss: 0.7478971313438474 Test Loss: 0.7898906705076285
Iteration: 4400 Train Loss: 1.0779496686582095 Test Loss: 1.3273379437625799
Iteration: 4600 Train Loss: 0.7542640456121408 Test Loss: 0.7409690006872467
Epoch: 5 Mean Train Loss: 0.8506625429587532 Train Accuracy: 75.49%
Iteration: 4800 Train Loss: 0.5772293039663141 Test Loss: 0.8754192604355209
Iteration: 5000 Train Loss: 0.46629982887113086 Test Loss: 0.9119080327514741
Iteration: 5200 Train Loss: 0.9247667461701483 Test Loss: 1.409989314880598
Iteration: 5400 Train Loss: 0.9956703907358364 Test Loss: 0.9382074302615898
Iteration: 5600 Train Loss: 1.300627647934534 Test Loss: 0.9880029097803537
Epoch: 6 Mean Train Loss: 0.9427874202609406 Train Accuracy: 71.48%
Iteration: 5800 Train Loss: 0.943760963659874 Test Loss: 0.8721619731516177
Iteration: 6000 Train Loss: 1.5565271862661967 Test Loss: 2.3640052587359346
Iteration: 6200 Train Loss: 1.640370464828468 Test Loss: 0.9267813334759313
Iteration: 6400 Train Loss: 0.8072071453272058 Test Loss: 1.026942928787609
Epoch: 7 Mean Train Loss: 0.9666084855609441 Train Accuracy: 69.90%
Iteration: 6600 Train Loss: 1.0057265987269925 Test Loss: 1.047887617812218
Iteration: 6800 Train Loss: 1.882743883341938 Test Loss: 0.9666124204960774
Iteration: 7000 Train Loss: 1.3362011621654104 Test Loss: 1.3027209957133785
```

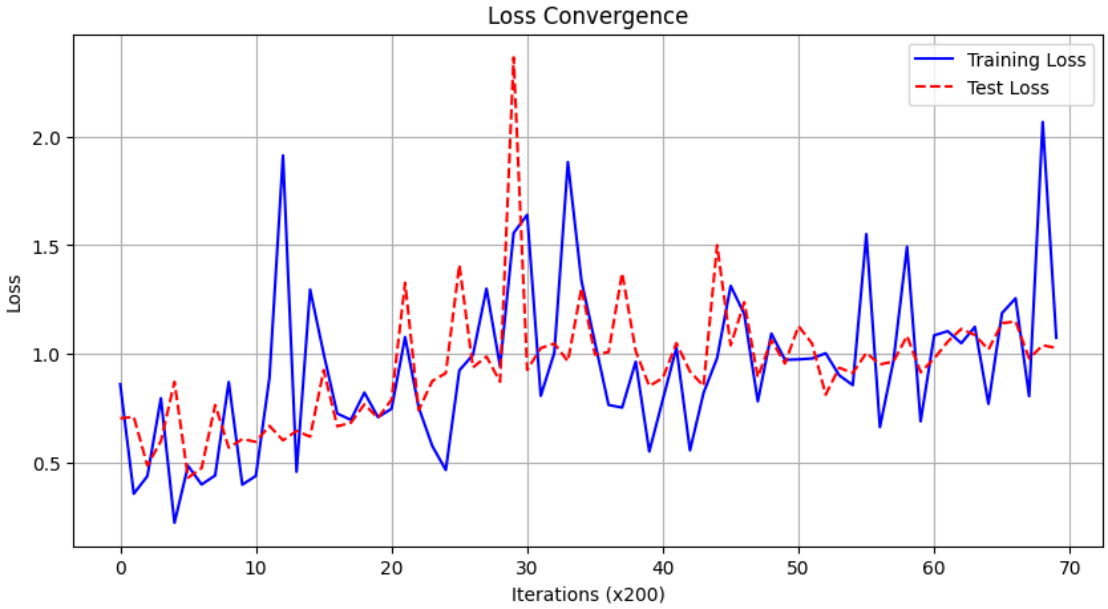
```
Iteration: 7200 Train Loss: 1.0513942538814638 Test Loss: 0.9973713848570974
Iteration: 7400 Train Loss: 0.7655073541863792 Test Loss: 1.0087141860846183
Epoch: 8 Mean Train Loss: 0.9699002259507638 Train Accuracy: 68.97%
Iteration: 7600 Train Loss: 0.753470292211167 Test Loss: 1.3728610436335373
Iteration: 7800 Train Loss: 0.9645144609051839 Test Loss: 1.0128929728238392
Iteration: 8000 Train Loss: 0.5514662168762763 Test Loss: 0.8525351376045068
Iteration: 8200 Train Loss: 0.7908010833509823 Test Loss: 0.8894912037832319
Iteration: 8400 Train Loss: 1.0286032453946983 Test Loss: 1.0500208584414434
Epoch: 9 Mean Train Loss: 0.9772672042925444 Train Accuracy: 68.53%
Iteration: 8600 Train Loss: 0.5564287405559589 Test Loss: 0.9194471788412909
Iteration: 8800 Train Loss: 0.8206366074240425 Test Loss: 0.8529526444196239
Iteration: 9000 Train Loss: 0.9813729885554836 Test Loss: 1.5003484314580213
Iteration: 9200 Train Loss: 1.3133291855537181 Test Loss: 1.039647970430479
Epoch: 10 Mean Train Loss: 1.0328064501308545 Train Accuracy: 65.62%
Iteration: 9400 Train Loss: 1.183858049720097 Test Loss: 1.2382111174244363
Iteration: 9600 Train Loss: 0.7818389501529228 Test Loss: 0.8930212438133966
Iteration: 9800 Train Loss: 1.093723365897941 Test Loss: 1.063967177873172
Iteration: 10000 Train Loss: 0.9731280208730386 Test Loss: 0.9561408183685912
Iteration: 10200 Train Loss: 0.9750303432917676 Test Loss: 1.1281255212490826
Epoch: 11 Mean Train Loss: 1.0133182820636237 Train Accuracy: 66.17%
Iteration: 10400 Train Loss: 0.9794198486158336 Test Loss: 1.0480024584576155
Iteration: 10600 Train Loss: 1.0034258861144931 Test Loss: 0.8131191416311242
Iteration: 10800 Train Loss: 0.9019736483705898 Test Loss: 0.9367543409172432
Iteration: 11000 Train Loss: 0.8567169036405026 Test Loss: 0.91226894537347
Iteration: 11200 Train Loss: 1.5513301035680591 Test Loss: 1.0049044151026458
Epoch: 12 Mean Train Loss: 1.000893442404341 Train Accuracy: 66.42%
Iteration: 11400 Train Loss: 0.663915387648535 Test Loss: 0.952005411718073
Iteration: 11600 Train Loss: 0.9690277001889079 Test Loss: 0.9651282518295147
Iteration: 11800 Train Loss: 1.4932848813149104 Test Loss: 1.0833856157092399
Iteration: 12000 Train Loss: 0.6898969926654881 Test Loss: 0.9156255432876522
Epoch: 13 Mean Train Loss: 1.0020561202912914 Train Accuracy: 66.84%
Iteration: 12200 Train Loss: 1.0857017909818896 Test Loss: 0.9789491713767375
Iteration: 12400 Train Loss: 1.104961313924397 Test Loss: 1.056777093006338
Iteration: 12600 Train Loss: 1.049535014943951 Test Loss: 1.115385574580768
Iteration: 12800 Train Loss: 1.125001215028015 Test Loss: 1.0872451715846914
Iteration: 13000 Train Loss: 0.7706531856122625 Test Loss: 1.0203926510557293
Epoch: 14 Mean Train Loss: 1.085053832938938 Train Accuracy: 63.81%
Iteration: 13200 Train Loss: 1.1879322393458842 Test Loss: 1.1412629325492165
Iteration: 13400 Train Loss: 1.256133796544501 Test Loss: 1.1498014522790754
Iteration: 13600 Train Loss: 0.8056995024127995 Test Loss: 0.9791361574106883
Iteration: 13800 Train Loss: 2.067259117060872 Test Loss: 1.0399406910346247
Iteration: 14000 Train Loss: 1.075116884031449 Test Loss: 1.0276082894168732
Epoch: 15 Mean Train Loss: 1.0719968598021274 Train Accuracy: 63.84%
Model trained!
```

[ ]:
```
best_model_rmsprop.evaluate(X_train, Y_train, X_test, Y_test)
```

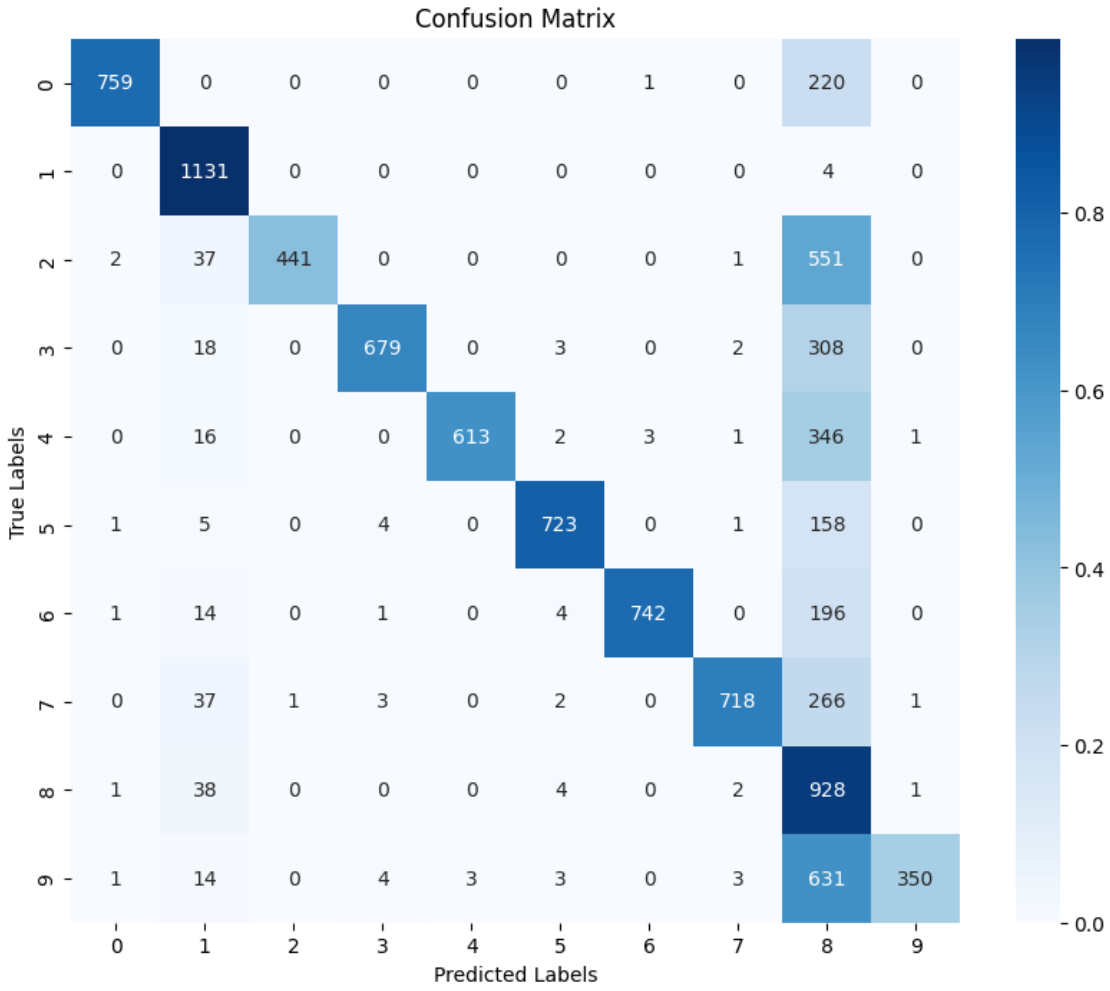
```
Train loss: 1.4270552521192934 Train accuracy: 70.71%
```

```
Test loss: 1.4923830521884554 Test accuracy: 70.84%

Confusion Matrix:
 [[ 759    0    0    0    0    0    1    0  220    0]
 [   0 1131    0    0    0    0    0    0    4    0]
 [   2   37  441    0    0    0    0    1  551    0]
 [   0   18    0  679    0    3    0    2  308    0]
 [   0   16    0    0  613    2    3    1  346    1]
 [   1    5    0    4    0  723    0    1  158    0]
 [   1   14    0    1    0    4  742    0  196    0]
 [   0   37    1    3    0    2    0  718  266    1]
 [   1   38    0    0    0    4    0    2  928    1]
 [   1   14    0    4    3    3    0    3  631  350]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.77      0.87       980
           1       0.86      1.00      0.93      1135
           2       1.00      0.43      0.60      1032
           3       0.98      0.67      0.80      1010
           4       1.00      0.62      0.77       982
           5       0.98      0.81      0.89       892
           6       0.99      0.77      0.87       958
           7       0.99      0.70      0.82      1028
           8       0.26      0.95      0.41       974
           9       0.99      0.35      0.51      1009

    accuracy                           0.71     10000
   macro avg       0.90      0.71      0.75     10000
weighted avg       0.90      0.71      0.75     10000
```

[ ]:
```
best_model_rmsprop.plot_losses()
```



[ ]:
```
best_model_rmsprop.plot_confusion_matrix(X_test, Y_test)
```



8.3 Adam

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    ReLU(),
    FNNLayer(500, 250),
    ReLU(),
    FNNLayer(250, 100),
    ReLU(),
    FNNLayer(100, output_dim)
]

best_model_adam = NN(layers)
```

```
optimizer = Adam()
loss = LogLoss()
best_model_adam.compile(loss=loss, optimizer=optimizer)
best_model_adam.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.2626578659015413 Test Loss: 0.3849439589897805
Iteration: 400 Train Loss: 0.2356629693344481 Test Loss: 0.4549075785197895
Iteration: 600 Train Loss: 0.31019765411731076 Test Loss: 0.3430033483875648
Iteration: 800 Train Loss: 0.24733567676926344 Test Loss: 0.2757730784214253
Epoch: 1 Mean Train Loss: 0.4339105918944163 Train Accuracy: 89.18%
Iteration: 1000 Train Loss: 0.30689332211144105 Test Loss: 0.2731464007045547
Iteration: 1200 Train Loss: 0.20397046367776933 Test Loss: 0.3769309562724609
Iteration: 1400 Train Loss: 0.2926237877439811 Test Loss: 0.3519468038054768
Iteration: 1600 Train Loss: 0.32132891775642125 Test Loss: 0.36277139073396836
Iteration: 1800 Train Loss: 0.19108151246371324 Test Loss: 0.37126781559592564
Epoch: 2 Mean Train Loss: 0.34407863935596317 Train Accuracy: 91.85%
Iteration: 2000 Train Loss: 0.18625849158469626 Test Loss: 0.3034613864325888
Iteration: 2200 Train Loss: 0.2077125864000273 Test Loss: 0.2562673641192234
Iteration: 2400 Train Loss: 0.24967935684255763 Test Loss: 0.2817903709617708
Iteration: 2600 Train Loss: 0.27114698366677725 Test Loss: 0.25796518477503866
Iteration: 2800 Train Loss: 0.1905257292162674 Test Loss: 0.24870348804405484
Epoch: 3 Mean Train Loss: 0.25680051545545385 Train Accuracy: 93.68%
Iteration: 3000 Train Loss: 0.25278994051136255 Test Loss: 0.26779208614235234
Iteration: 3200 Train Loss: 0.14313394386906672 Test Loss: 0.2554887791230677
Iteration: 3400 Train Loss: 0.12037428737233749 Test Loss: 0.23795346339669926
Iteration: 3600 Train Loss: 0.22403003912183447 Test Loss: 0.2622124607699704
Epoch: 4 Mean Train Loss: 0.22277164319472417 Train Accuracy: 94.75%
Iteration: 3800 Train Loss: 0.2774051643196963 Test Loss: 0.24241948169291885
Iteration: 4000 Train Loss: 0.11371677219951547 Test Loss: 0.2476849085725598
Iteration: 4200 Train Loss: 0.07844043990549412 Test Loss: 0.28339775725397537
Iteration: 4400 Train Loss: 0.18628275379997355 Test Loss: 0.2247177708386428
Iteration: 4600 Train Loss: 0.013548687379626077 Test Loss: 0.22664118046974727
Epoch: 5 Mean Train Loss: 0.19964308333670028 Train Accuracy: 95.24%
Iteration: 4800 Train Loss: 0.10277031738586628 Test Loss: 0.24609341087204645
Iteration: 5000 Train Loss: 0.1704016946852441 Test Loss: 0.2544927843618474
Iteration: 5200 Train Loss: 0.19037328243046792 Test Loss: 0.28013878771966444
Iteration: 5400 Train Loss: 0.0809467097753698 Test Loss: 0.23839812795511528
Iteration: 5600 Train Loss: 0.37050944284753806 Test Loss: 0.24785718169340432
Epoch: 6 Mean Train Loss: 0.21681198311280486 Train Accuracy: 94.85%
Iteration: 5800 Train Loss: 0.25295916949720487 Test Loss: 0.2587479837797115
Iteration: 6000 Train Loss: 0.6565094497574797 Test Loss: 0.28887939591885864
Iteration: 6200 Train Loss: 0.437349459191386 Test Loss: 0.3394061052038449
Iteration: 6400 Train Loss: 0.36795757171409177 Test Loss: 0.2410804737866429
Epoch: 7 Mean Train Loss: 0.19133799617329736 Train Accuracy: 95.06%
Iteration: 6600 Train Loss: 0.06347981426003459 Test Loss: 0.2818861083469183
Iteration: 6800 Train Loss: 0.33750740741581525 Test Loss: 0.42097241004816355
Iteration: 7000 Train Loss: 0.2243472051462386 Test Loss: 0.2503169342995386
```

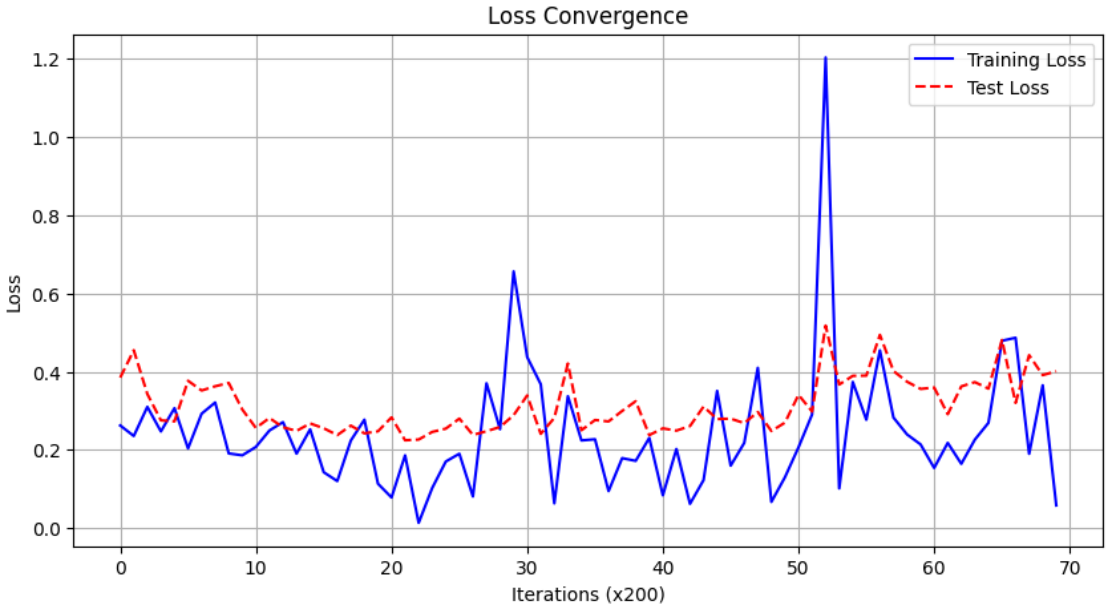
```
Iteration: 7200 Train Loss: 0.22731401738373472 Test Loss: 0.2765469330154619
Iteration: 7400 Train Loss: 0.09478409849923614 Test Loss: 0.27289794357295427
Epoch: 8 Mean Train Loss: 0.22391586797894736 Train Accuracy: 94.63%
Iteration: 7600 Train Loss: 0.17907206611417675 Test Loss: 0.2996674943260333
Iteration: 7800 Train Loss: 0.1721385412957175 Test Loss: 0.324665751328771
Iteration: 8000 Train Loss: 0.23079097425838863 Test Loss: 0.23797466419952326
Iteration: 8200 Train Loss: 0.08394911946719855 Test Loss: 0.2551646670189457
Iteration: 8400 Train Loss: 0.20257448466867156 Test Loss: 0.24945174565874478
Epoch: 9 Mean Train Loss: 0.19896713737193267 Train Accuracy: 95.25%
Iteration: 8600 Train Loss: 0.06221428066492719 Test Loss: 0.2609876964130304
Iteration: 8800 Train Loss: 0.12292838694254485 Test Loss: 0.3108616440641256
Iteration: 9000 Train Loss: 0.35095075109839313 Test Loss: 0.2789629847321414
Iteration: 9200 Train Loss: 0.1597560184645811 Test Loss: 0.28001219704384256
Epoch: 10 Mean Train Loss: 0.21759450213421572 Train Accuracy: 94.64%
Iteration: 9400 Train Loss: 0.21807180457022596 Test Loss: 0.26859768300976006
Iteration: 9600 Train Loss: 0.40994387231043883 Test Loss: 0.2969664832202843
Iteration: 9800 Train Loss: 0.06715535516695606 Test Loss: 0.24837618632637384
Iteration: 10000 Train Loss: 0.13023435714672327 Test Loss: 0.269830512669518
Iteration: 10200 Train Loss: 0.20676412888682297 Test Loss: 0.34093671960142086
Epoch: 11 Mean Train Loss: 0.20070384724195756 Train Accuracy: 94.95%
Iteration: 10400 Train Loss: 0.2907359266503591 Test Loss: 0.2982581559466734
Iteration: 10600 Train Loss: 1.203504031436232 Test Loss: 0.5181164749399098
Iteration: 10800 Train Loss: 0.10165072912190838 Test Loss: 0.36721532752379166
Iteration: 11000 Train Loss: 0.37400417242215245 Test Loss: 0.3894682777935805
Iteration: 11200 Train Loss: 0.27720971387264587 Test Loss: 0.3900544784020601
Epoch: 12 Mean Train Loss: 0.27872326459883423 Train Accuracy: 93.34%
Iteration: 11400 Train Loss: 0.45468726795139786 Test Loss: 0.49425682894997164
Iteration: 11600 Train Loss: 0.28278791637578804 Test Loss: 0.4018022794408151
Iteration: 11800 Train Loss: 0.23968255623157358 Test Loss: 0.3744407041124194
Iteration: 12000 Train Loss: 0.2140563232120147 Test Loss: 0.3560558779693011
Epoch: 13 Mean Train Loss: 0.2912823932577764 Train Accuracy: 92.97%
Iteration: 12200 Train Loss: 0.15399199499834826 Test Loss: 0.3595547367524739
Iteration: 12400 Train Loss: 0.21811455042804154 Test Loss: 0.29133793913606953
Iteration: 12600 Train Loss: 0.16454194747882422 Test Loss: 0.36249011507659973
Iteration: 12800 Train Loss: 0.2264283416197995 Test Loss: 0.37343067547023573
Iteration: 13000 Train Loss: 0.2693146117037535 Test Loss: 0.35663374017165844
Epoch: 14 Mean Train Loss: 0.27252331172866257 Train Accuracy: 93.09%
Iteration: 13200 Train Loss: 0.47923726160244096 Test Loss: 0.48123807373875666
Iteration: 13400 Train Loss: 0.4868774659530085 Test Loss: 0.32023836761030966
Iteration: 13600 Train Loss: 0.19016253614257617 Test Loss: 0.4424125540595789
Iteration: 13800 Train Loss: 0.36542552869061695 Test Loss: 0.3913098119562008
Iteration: 14000 Train Loss: 0.05819649262449997 Test Loss: 0.40114375388408946
Epoch: 15 Mean Train Loss: 0.29239831290200624 Train Accuracy: 92.59%
Model trained!
```

[ ]:
```
best_model_adam.evaluate(X_train, Y_train, X_test, Y_test)
```

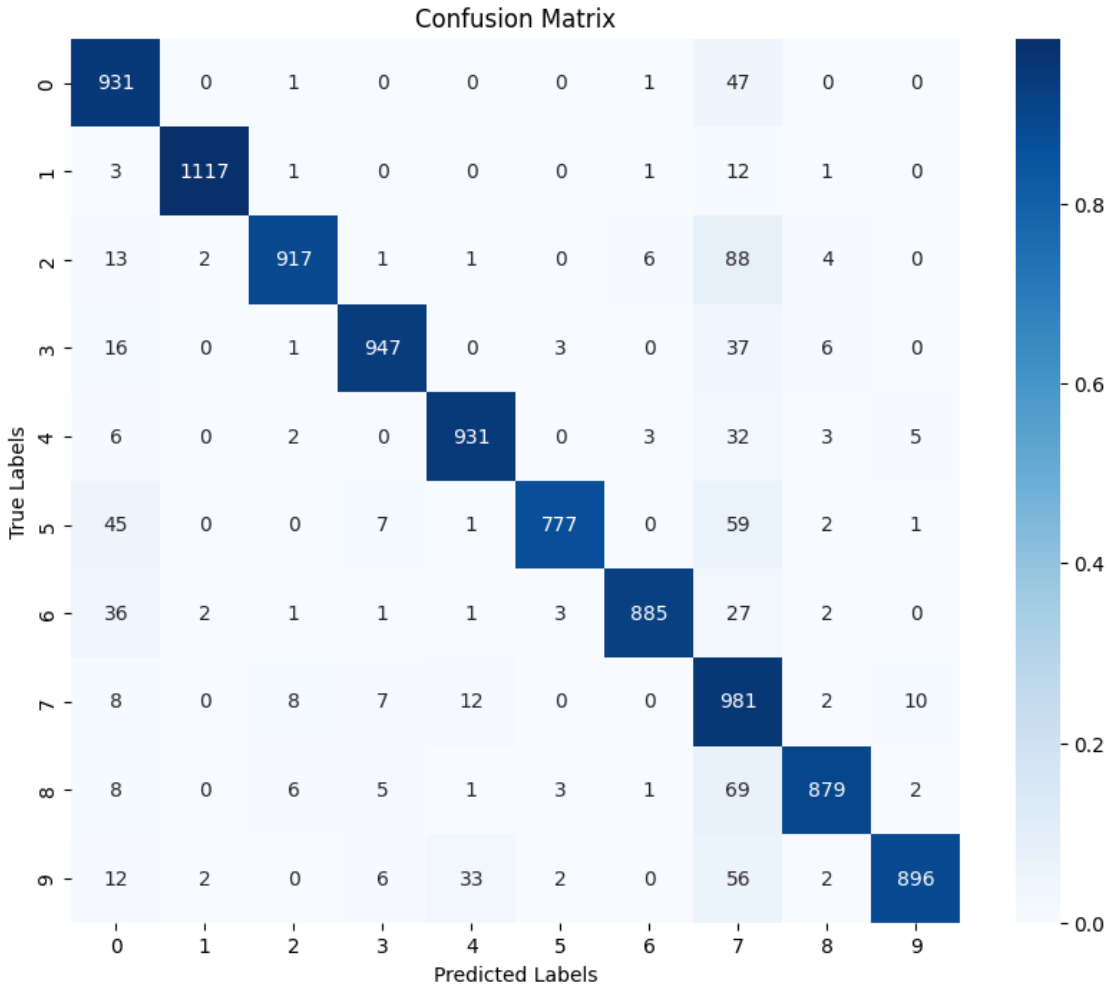
```
Train loss: 0.28485010562113333 Train accuracy: 93.32%
```

```
Test loss: 0.35776347232648775 Test accuracy: 92.61%

Confusion Matrix:
 [[ 931    0    1    0    0    0    1   47    0    0]
 [   3 1117    1    0    0    0    1   12    1    0]
 [  13    2  917    1    1    0    6   88    4    0]
 [  16    0    1  947    0    3    0   37    6    0]
 [   6    0    2    0  931    0    3   32    3    5]
 [  45    0    0    7    1  777    0   59    2    1]
 [  36    2    1    1    1    3  885   27    2    0]
 [   8    0    8    7   12    0    0  981    2   10]
 [   8    0    6    5    1    3    1   69  879    2]
 [  12    2    0    6   33    2    0   56    2  896]]

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.95      0.90       980
           1       0.99      0.98      0.99      1135
           2       0.98      0.89      0.93      1032
           3       0.97      0.94      0.95      1010
           4       0.95      0.95      0.95       982
           5       0.99      0.87      0.93       892
           6       0.99      0.92      0.95       958
           7       0.70      0.95      0.81      1028
           8       0.98      0.90      0.94       974
           9       0.98      0.89      0.93      1009

    accuracy                           0.93     10000
   macro avg       0.94      0.92      0.93     10000
weighted avg       0.94      0.93      0.93     10000
```

[ ]:
```
best_model_adam.plot_losses()
```



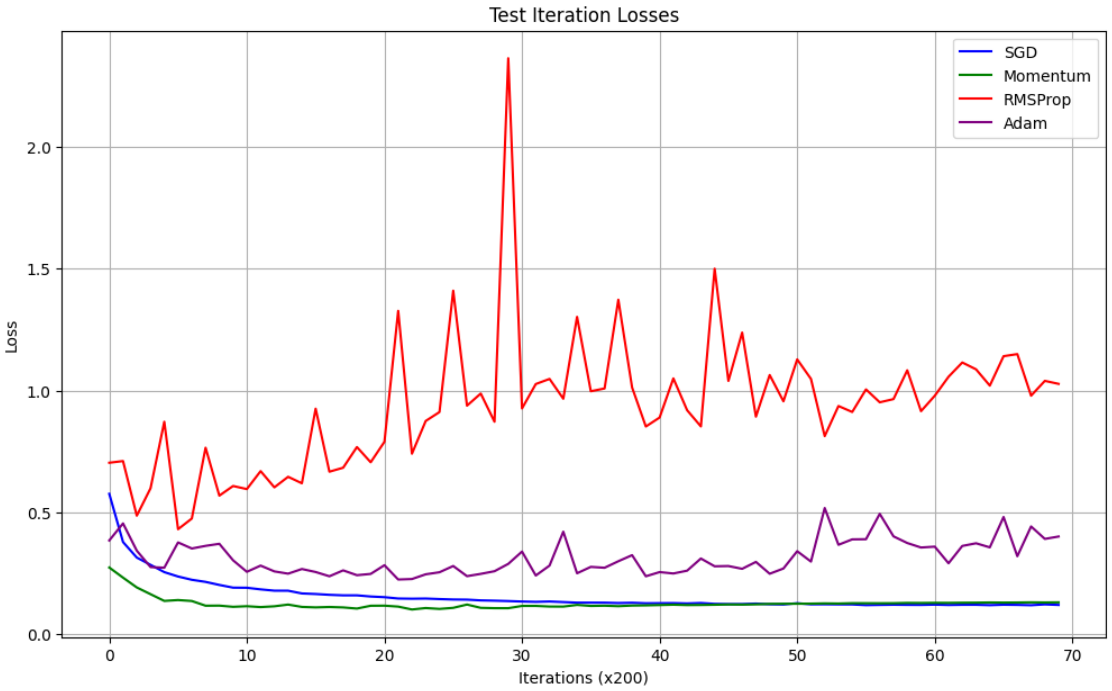
[ ]:
```
best_model_adam.plot_confusion_matrix(X_test, Y_test)
```



8.4 Comparing the 4 optimizers

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

def plot_models_losses(models, labels, title, attribute):
    plt.figure(figsize=(12, 7))

    colors = ['blue', 'green', 'red', 'purple']

    for i, model in enumerate(models):
        losses = getattr(model, attribute)
        plt.plot(losses, label=labels[i], color=colors[i])

    plt.xlabel('Iterations (x200)')
    plt.ylabel('Loss')
    plt.title(title)
```

```
    plt.legend()
    plt.grid(True)
    plt.show()

model1 = relu_model
model2 = best_model_momentum
model3 = best_model_rmsprop
model4 = best_model_adam

models = [model1, model2, model3, model4]
labels = ['SGD', 'Momentum', 'RMSProp', 'Adam']

# Plots training iteration losses
plot_models_losses(models, labels, 'Training Iteration Losses',␣
 ↪'iteration_losses')

# Plots test iteration losses
plot_models_losses(models, labels, 'Test Iteration Losses',␣
 ↪'iteration_test_losses')
```



Optimizer	Test Accuracy	Train Accuracy	Train Loss	Test Loss
SGD	96.91%	99.58%	0.0228	0.1212
Momentum	97.85%	100.00%	0.0004	0.1321
RMSProp	70.84%	70.71%	1.4270	1,4924
Adam	92.61%	93.32%	0.2848	0.3577

- I am surprised by the results, because I expected RMSProp and Adam to perform better but I think the initializations were unfortunate for these two models in this run. Momentum and GD seem to be more stable optimizers also as we see less variance in the loss plots.
- RMSProp and Adam are both hyperparameter sensitive as they have other beta, epsilon, etc. terms in their formulae. The values I chose might not have been suitable for this setup.
- Momentum performs better than GD and leads to faster convergence as well. This is also expected because the moving average of past gradients acts as a kind of inertia. This inertia can help the optimizer to navigate through the optimization landscape and overcome small barriers.

9 Performance with MSE Loss as criterion

We want to see if using Mean-squared error as the loss criterion for classification problems has a huge impact on the performance. We'll look at the base model architecture (ReLU) with the GD and Momentum optimizers.

9.1 Best Model with SGD

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    ReLU(),
    FNNLayer(500, 250),
    ReLU(),
    FNNLayer(250, 100),
    ReLU(),
    FNNLayer(100, output_dim)
]

best_model_mse = NN(layers)
optimizer = SGD()
loss = MSE()
best_model_mse.compile(loss=loss, optimizer=optimizer)
best_model_mse.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.7639431459233281 Test Loss: 0.786340076958693
Iteration: 400 Train Loss: 0.6173905654223258 Test Loss: 0.6315594198813431
Iteration: 600 Train Loss: 0.4696360210640014 Test Loss: 0.4818534525108481
Iteration: 800 Train Loss: 0.4556449154616644 Test Loss: 0.3647332202124966
Epoch: 1 Mean Train Loss: 0.597284572181989 Train Accuracy: 57.05%
Iteration: 1000 Train Loss: 0.2833996243166443 Test Loss: 0.2795598274564075
Iteration: 1200 Train Loss: 0.22323868131738006 Test Loss: 0.2287074202432093
Iteration: 1400 Train Loss: 0.21617243257778918 Test Loss: 0.19907003069065093
Iteration: 1600 Train Loss: 0.16603812966092046 Test Loss: 0.17950475417732326
Iteration: 1800 Train Loss: 0.18764176308506986 Test Loss: 0.16516704648111896
Epoch: 2 Mean Train Loss: 0.21713282376809714 Train Accuracy: 87.57%
Iteration: 2000 Train Loss: 0.13492070611817913 Test Loss: 0.1550719964994174
Iteration: 2200 Train Loss: 0.11147184044004207 Test Loss: 0.14666179367279797
Iteration: 2400 Train Loss: 0.1155742024395939 Test Loss: 0.13995260358368466
Iteration: 2600 Train Loss: 0.17391641865629723 Test Loss: 0.1343091802416463
Iteration: 2800 Train Loss: 0.13047690002809548 Test Loss: 0.12975459407857054
Epoch: 3 Mean Train Loss: 0.1472880469098416 Train Accuracy: 90.94%
Iteration: 3000 Train Loss: 0.08928457746151913 Test Loss: 0.12598153979806825
Iteration: 3200 Train Loss: 0.059944456457365826 Test Loss: 0.12242128203591265
Iteration: 3400 Train Loss: 0.05269903015226152 Test Loss: 0.11832420262431484
Iteration: 3600 Train Loss: 0.185144057789712 Test Loss: 0.11546343066164941
Epoch: 4 Mean Train Loss: 0.12187669145023035 Train Accuracy: 92.36%
Iteration: 3800 Train Loss: 0.10636175737323686 Test Loss: 0.11342077660472777
Iteration: 4000 Train Loss: 0.14309498222571498 Test Loss: 0.11083834679998217
Iteration: 4200 Train Loss: 0.1425896586055136 Test Loss: 0.10874706270862415
Iteration: 4400 Train Loss: 0.0720274045309563 Test Loss: 0.10634564729240743
Iteration: 4600 Train Loss: 0.06824578898430887 Test Loss: 0.1043854017838591
```

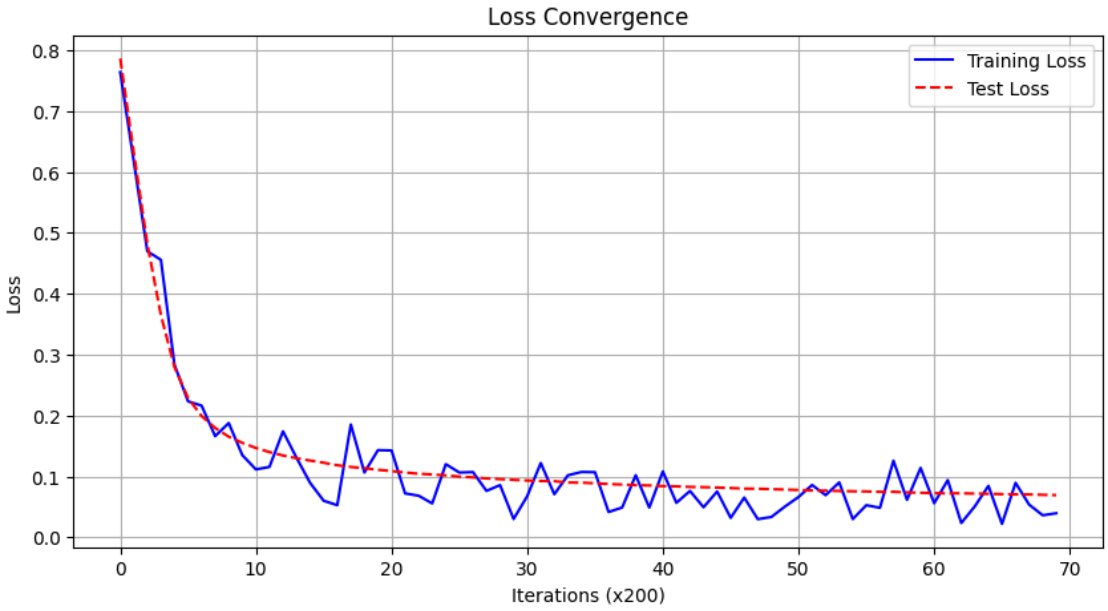
```
Epoch: 5 Mean Train Loss: 0.10693370850883217 Train Accuracy: 93.32%
Iteration: 4800 Train Loss: 0.05560203237864242 Test Loss: 0.10310101585391118
Iteration: 5000 Train Loss: 0.12008721549267246 Test Loss: 0.10164782530348863
Iteration: 5200 Train Loss: 0.10629087771015971 Test Loss: 0.09966892561106895
Iteration: 5400 Train Loss: 0.1071007688580228 Test Loss: 0.09837984402868968
Iteration: 5600 Train Loss: 0.076169469498836 Test Loss: 0.09663472792908667
Epoch: 6 Mean Train Loss: 0.09673957345318006 Train Accuracy: 93.99%
Iteration: 5800 Train Loss: 0.08571597310840609 Test Loss: 0.09548179497102922
Iteration: 6000 Train Loss: 0.030028258691386246 Test Loss: 0.09433122927965541
Iteration: 6200 Train Loss: 0.0674004170922172 Test Loss: 0.0932160838438881
Iteration: 6400 Train Loss: 0.12187917483484265 Test Loss: 0.09239922528754367
Epoch: 7 Mean Train Loss: 0.08898028055781079 Train Accuracy: 94.47%
Iteration: 6600 Train Loss: 0.07070092933457556 Test Loss: 0.09210199001929438
Iteration: 6800 Train Loss: 0.10178200934237253 Test Loss: 0.08989777133746658
Iteration: 7000 Train Loss: 0.1072523018346708 Test Loss: 0.0897348110011966
Iteration: 7200 Train Loss: 0.10691075234367212 Test Loss: 0.0886426064942368
Iteration: 7400 Train Loss: 0.04152298506877869 Test Loss: 0.08748583860283196
Epoch: 8 Mean Train Loss: 0.08258952301422777 Train Accuracy: 94.91%
Iteration: 7600 Train Loss: 0.04886169902399484 Test Loss: 0.08660670655245493
Iteration: 7800 Train Loss: 0.1018881465027024 Test Loss: 0.08561451879735578
Iteration: 8000 Train Loss: 0.04879379288265889 Test Loss: 0.08510567999583057
Iteration: 8200 Train Loss: 0.10805893850848872 Test Loss: 0.08402897779874652
Iteration: 8400 Train Loss: 0.05658445475503397 Test Loss: 0.08358134937038551
Epoch: 9 Mean Train Loss: 0.07732868580079372 Train Accuracy: 95.26%
Iteration: 8600 Train Loss: 0.07606378964932813 Test Loss: 0.08269421805888921
Iteration: 8800 Train Loss: 0.04925244083953967 Test Loss: 0.08230361073860697
Iteration: 9000 Train Loss: 0.07517640210188765 Test Loss: 0.08166087044540252
Iteration: 9200 Train Loss: 0.03190199373512347 Test Loss: 0.08094372385377474
Epoch: 10 Mean Train Loss: 0.07285201246845635 Train Accuracy: 95.57%
Iteration: 9400 Train Loss: 0.06518609142683214 Test Loss: 0.07986012398532547
Iteration: 9600 Train Loss: 0.029729673694888655 Test Loss: 0.07968944452716636
Iteration: 9800 Train Loss: 0.03336583491820955 Test Loss: 0.07905035106664272
Iteration: 10000 Train Loss: 0.050449594155700106 Test Loss: 0.07835387071232701
Iteration: 10200 Train Loss: 0.0661313048087308 Test Loss: 0.07765035357892595
Epoch: 11 Mean Train Loss: 0.06899098173317385 Train Accuracy: 95.80%
Iteration: 10400 Train Loss: 0.08585385212620875 Test Loss: 0.07686347902008628
Iteration: 10600 Train Loss: 0.06913472884821452 Test Loss: 0.07654287123093806
Iteration: 10800 Train Loss: 0.09011457189679062 Test Loss: 0.07600645214625401
Iteration: 11000 Train Loss: 0.029870091272042754 Test Loss: 0.07552858966510323
Iteration: 11200 Train Loss: 0.052715758945733215 Test Loss: 0.0753139220213199
Epoch: 12 Mean Train Loss: 0.0654082253989492 Train Accuracy: 96.05%
Iteration: 11400 Train Loss: 0.04834402863835882 Test Loss: 0.07461795189699634
Iteration: 11600 Train Loss: 0.12567233111447262 Test Loss: 0.07474552259327923
Iteration: 11800 Train Loss: 0.061513314187539164 Test Loss: 0.07368150494875379
Iteration: 12000 Train Loss: 0.11390198443669067 Test Loss: 0.07312518534654598
Epoch: 13 Mean Train Loss: 0.06218656496451477 Train Accuracy: 96.26%
Iteration: 12200 Train Loss: 0.05591549431112677 Test Loss: 0.0724416906970608
Iteration: 12400 Train Loss: 0.09380714554561559 Test Loss: 0.07217054323588255
```

```
Iteration: 12600 Train Loss: 0.023417959155154492 Test Loss: 0.07219261146051349
Iteration: 12800 Train Loss: 0.05094074636449108 Test Loss: 0.07176021595778805
Iteration: 13000 Train Loss: 0.08443745772415365 Test Loss: 0.07133933791932741
Epoch: 14 Mean Train Loss: 0.05937031790528617 Train Accuracy: 96.45%
Iteration: 13200 Train Loss: 0.021997539992406695 Test Loss: 0.07084402033351739
Iteration: 13400 Train Loss: 0.08943534773585553 Test Loss: 0.0704513597684367
Iteration: 13600 Train Loss: 0.05359865676443906 Test Loss: 0.07031547948865828
Iteration: 13800 Train Loss: 0.03613566172974491 Test Loss: 0.06970072144426512
Iteration: 14000 Train Loss: 0.03946408970562236 Test Loss: 0.0692457900100879
Epoch: 15 Mean Train Loss: 0.05672609550308393 Train Accuracy: 96.65%
Model trained!
```

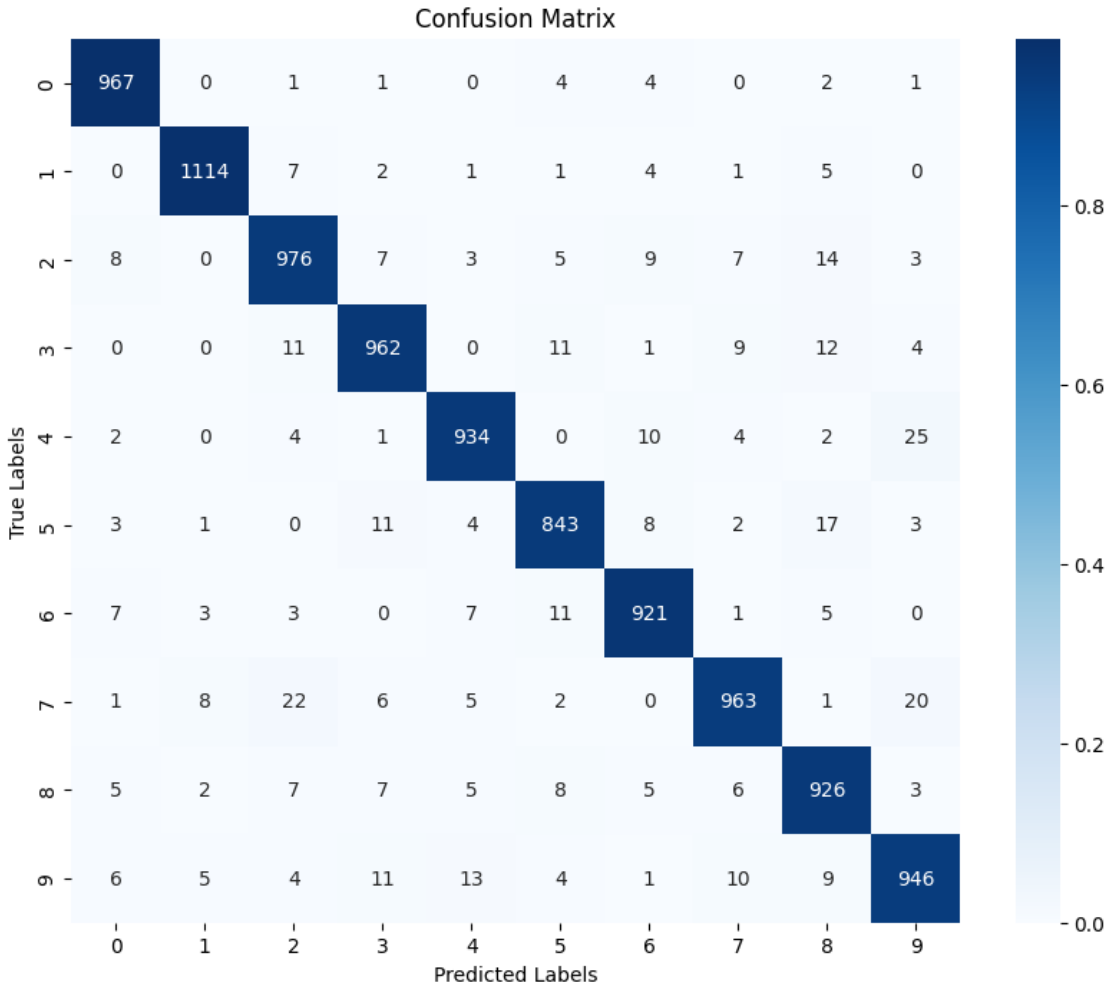
[ ]:
```
best_model_mse.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.05435408199843766 Train accuracy: 96.85%
Test loss: 0.06934993816712287 Test accuracy: 95.52%

Confusion Matrix:
 [[ 967    0    1    1    0    4    4    0    2    1]
 [   0 1114    7    2    1    1    4    1    5    0]
 [   8    0  976    7    3    5    9    7   14    3]
 [   0    0   11  962    0   11    1    9   12    4]
 [   2    0    4    1  934    0   10    4    2   25]
 [   3    1    0   11    4  843    8    2   17    3]
 [   7    3    3    0    7   11  921    1    5    0]
 [   1    8   22    6    5    2    0  963    1   20]
 [   5    2    7    7    5    8    5    6  926    3]
 [   6    5    4   11   13    4    1   10    9  946]]

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       980
           1       0.98      0.98      0.98      1135
           2       0.94      0.95      0.94      1032
           3       0.95      0.95      0.95      1010
           4       0.96      0.95      0.96       982
           5       0.95      0.95      0.95       892
           6       0.96      0.96      0.96       958
           7       0.96      0.94      0.95      1028
           8       0.93      0.95      0.94       974
           9       0.94      0.94      0.94      1009

    accuracy                           0.96     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.96      0.96      0.96     10000
```

[ ]:
```
best_model_mse.plot_losses()
```



[ ]:
```
best_model_mse.plot_confusion_matrix(X_test, Y_test)
```



9.2 Best Model with Momentum

[ ]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    ReLU(),
    FNNLayer(500, 250),
    ReLU(),
    FNNLayer(250, 100),
    ReLU(),
    FNNLayer(100, output_dim)
]

best_model_mse_momentum = NN(layers)
```

```
optimizer = Momentum()
loss = MSE()
best_model_mse_momentum.compile(loss=loss, optimizer=optimizer)
best_model_mse_momentum.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.1970906314136762 Test Loss: 0.1805879480748743
Iteration: 400 Train Loss: 0.17197580855856368 Test Loss: 0.12912247218147627
Iteration: 600 Train Loss: 0.07667621654166826 Test Loss: 0.10589176116273526
Iteration: 800 Train Loss: 0.20201804926609157 Test Loss: 0.09189451963171143
Epoch: 1 Mean Train Loss: 0.19116738321512441 Train Accuracy: 86.98%
Iteration: 1000 Train Loss: 0.11846449367061995 Test Loss: 0.08682049771885827
Iteration: 1200 Train Loss: 0.08658912464911242 Test Loss: 0.08134716743986997
Iteration: 1400 Train Loss: 0.07981202378648175 Test Loss: 0.07847022954196056
Iteration: 1600 Train Loss: 0.13479868614625845 Test Loss: 0.07130267924571303
Iteration: 1800 Train Loss: 0.05162109883333352 Test Loss: 0.07370463489524325
Epoch: 2 Mean Train Loss: 0.0723230574914483 Train Accuracy: 95.46%
Iteration: 2000 Train Loss: 0.038583662274235764 Test Loss: 0.06747866685555669
Iteration: 2200 Train Loss: 0.05013087989267662 Test Loss: 0.0642704736385103
Iteration: 2400 Train Loss: 0.022534868103959865 Test Loss: 0.06327812510719653
Iteration: 2600 Train Loss: 0.027830584160751858 Test Loss: 0.06368344672709586
Iteration: 2800 Train Loss: 0.046428133428981215 Test Loss: 0.06383099886761731
Epoch: 3 Mean Train Loss: 0.05277907316348113 Train Accuracy: 96.81%
Iteration: 3000 Train Loss: 0.03224140345252674 Test Loss: 0.057476442416864885
Iteration: 3200 Train Loss: 0.0399471732878066 Test Loss: 0.05742139663415011
Iteration: 3400 Train Loss: 0.13856692120312644 Test Loss: 0.055003207828667924
Iteration: 3600 Train Loss: 0.10615227384639969 Test Loss: 0.05592716567951586
Epoch: 4 Mean Train Loss: 0.041287132426810036 Train Accuracy: 97.52%
Iteration: 3800 Train Loss: 0.0854888353035457 Test Loss: 0.05522072914327266
Iteration: 4000 Train Loss: 0.03266194929109867 Test Loss: 0.055574120642917876
Iteration: 4200 Train Loss: 0.06781348403715717 Test Loss: 0.05411973912333867
Iteration: 4400 Train Loss: 0.1365357166590022 Test Loss: 0.050739730282715355
Iteration: 4600 Train Loss: 0.07706192685743045 Test Loss: 0.051806048713713436
Epoch: 5 Mean Train Loss: 0.033764713045780895 Train Accuracy: 98.05%
Iteration: 4800 Train Loss: 0.03954390611153155 Test Loss: 0.050304554238037875
Iteration: 5000 Train Loss: 0.04799084620243607 Test Loss: 0.05018510070403136
Iteration: 5200 Train Loss: 0.0018084978210678431 Test Loss:
0.052504271214224595
Iteration: 5400 Train Loss: 0.05111471382395269 Test Loss: 0.0506403465431699
Iteration: 5600 Train Loss: 0.003033212889893 Test Loss: 0.051052889486848996
Epoch: 6 Mean Train Loss: 0.027625003902990438 Train Accuracy: 98.45%
Iteration: 5800 Train Loss: 0.01969120487146415 Test Loss: 0.04882618192933534
Iteration: 6000 Train Loss: 0.03640926428603445 Test Loss: 0.049381467852933514
Iteration: 6200 Train Loss: 0.05362784283931517 Test Loss: 0.050901698901937215
Iteration: 6400 Train Loss: 0.04624667939149073 Test Loss: 0.047948724628327354
Epoch: 7 Mean Train Loss: 0.022791588952280756 Train Accuracy: 98.80%
Iteration: 6600 Train Loss: 0.041993591243251915 Test Loss: 0.04948511018059221
Iteration: 6800 Train Loss: 0.032257863144419585 Test Loss: 0.04909422675535733
```

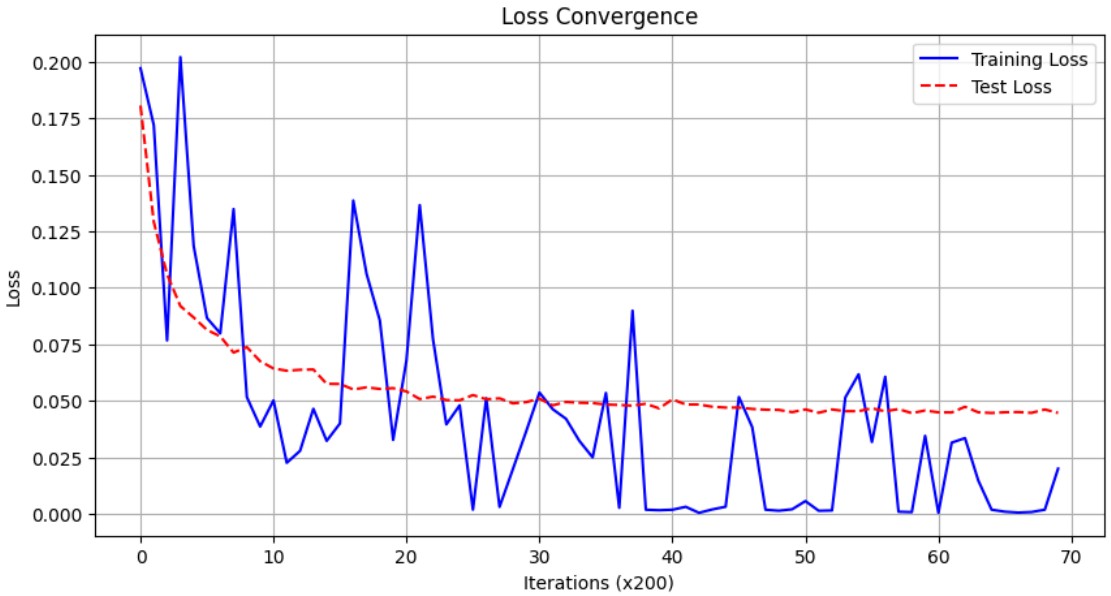
```
Iteration: 7000 Train Loss: 0.024944747752649262 Test Loss: 0.04895996096644378
Iteration: 7200 Train Loss: 0.053439949673676906 Test Loss: 0.048303684407267636
Iteration: 7400 Train Loss: 0.002624051350678323 Test Loss: 0.04812586296227786
Epoch: 8 Mean Train Loss: 0.019575287203726033 Train Accuracy: 98.96%
Iteration: 7600 Train Loss: 0.08983391179761235 Test Loss: 0.04782534694976398
Iteration: 7800 Train Loss: 0.0017805192439540993 Test Loss:
0.048674660710060824
Iteration: 8000 Train Loss: 0.0015677020205807908 Test Loss: 0.04661512587178291
Iteration: 8200 Train Loss: 0.001776553823964323 Test Loss: 0.05063365784521381
Iteration: 8400 Train Loss: 0.0030786164725278963 Test Loss: 0.04832175740204139
Epoch: 9 Mean Train Loss: 0.01799233723239177 Train Accuracy: 99.06%
Iteration: 8600 Train Loss: 0.0004082271823611512 Test Loss:
0.048333382070440044
Iteration: 8800 Train Loss: 0.0019195581745124041 Test Loss: 0.04735216444248095
Iteration: 9000 Train Loss: 0.003102138526230075 Test Loss: 0.04697577840398052
Iteration: 9200 Train Loss: 0.0516563774155374 Test Loss: 0.04697523875810805
Epoch: 10 Mean Train Loss: 0.015708917432344444 Train Accuracy: 99.18%
Iteration: 9400 Train Loss: 0.03830874979827771 Test Loss: 0.04636547761474864
Iteration: 9600 Train Loss: 0.0017828709918086732 Test Loss:
0.046042723670484724
Iteration: 9800 Train Loss: 0.0013513709229307337 Test Loss: 0.04594534093614382
Iteration: 10000 Train Loss: 0.001979297690821758 Test Loss: 0.04494378032298385
Iteration: 10200 Train Loss: 0.0056029495582733315 Test Loss:
0.04616363317513936
Epoch: 11 Mean Train Loss: 0.01390834279577289 Train Accuracy: 99.27%
Iteration: 10400 Train Loss: 0.0013222294176251272 Test Loss:
0.044670301600777076
Iteration: 10600 Train Loss: 0.0014678502090252096 Test Loss:
0.04614569630343927
Iteration: 10800 Train Loss: 0.05145090611202465 Test Loss: 0.04537894847064648
Iteration: 11000 Train Loss: 0.061712134229466295 Test Loss:
0.045385875565114545
Iteration: 11200 Train Loss: 0.03173156081339136 Test Loss: 0.04665241019240059
Epoch: 12 Mean Train Loss: 0.013079484110559177 Train Accuracy: 99.32%
Iteration: 11400 Train Loss: 0.06060056469231619 Test Loss: 0.0453349599058866
Iteration: 11600 Train Loss: 0.000905516461129438 Test Loss: 0.0463148142949495
Iteration: 11800 Train Loss: 0.0007240195436050568 Test Loss:
0.04450212136354454
Iteration: 12000 Train Loss: 0.034501425244259844 Test Loss: 0.04569477064206384
Epoch: 13 Mean Train Loss: 0.012238949698139627 Train Accuracy: 99.36%
Iteration: 12200 Train Loss: 0.0004142310142220414 Test Loss:
0.04486287684754749
Iteration: 12400 Train Loss: 0.03146009622004301 Test Loss: 0.044818674192978705
Iteration: 12600 Train Loss: 0.03347036108221617 Test Loss: 0.04733055253716873
Iteration: 12800 Train Loss: 0.014690610807754963 Test Loss: 0.04487274968137074
Iteration: 13000 Train Loss: 0.0017982261806102266 Test Loss:
0.04457748470219147
Epoch: 14 Mean Train Loss: 0.01083512827291987 Train Accuracy: 99.44%
```

```
Iteration: 13200 Train Loss: 0.0009061920499299286 Test Loss:
0.04489683403413712
Iteration: 13400 Train Loss: 0.0005269156351594572 Test Loss:
0.044979619287331005
Iteration: 13600 Train Loss: 0.0007636593809900603 Test Loss:
0.04462478216316208
Iteration: 13800 Train Loss: 0.0017930969702823125 Test Loss:
0.04613263679752347
Iteration: 14000 Train Loss: 0.01995772081227858 Test Loss: 0.04460141538948777
Epoch: 15 Mean Train Loss: 0.010092435763894904 Train Accuracy: 99.48%
Model trained!
```

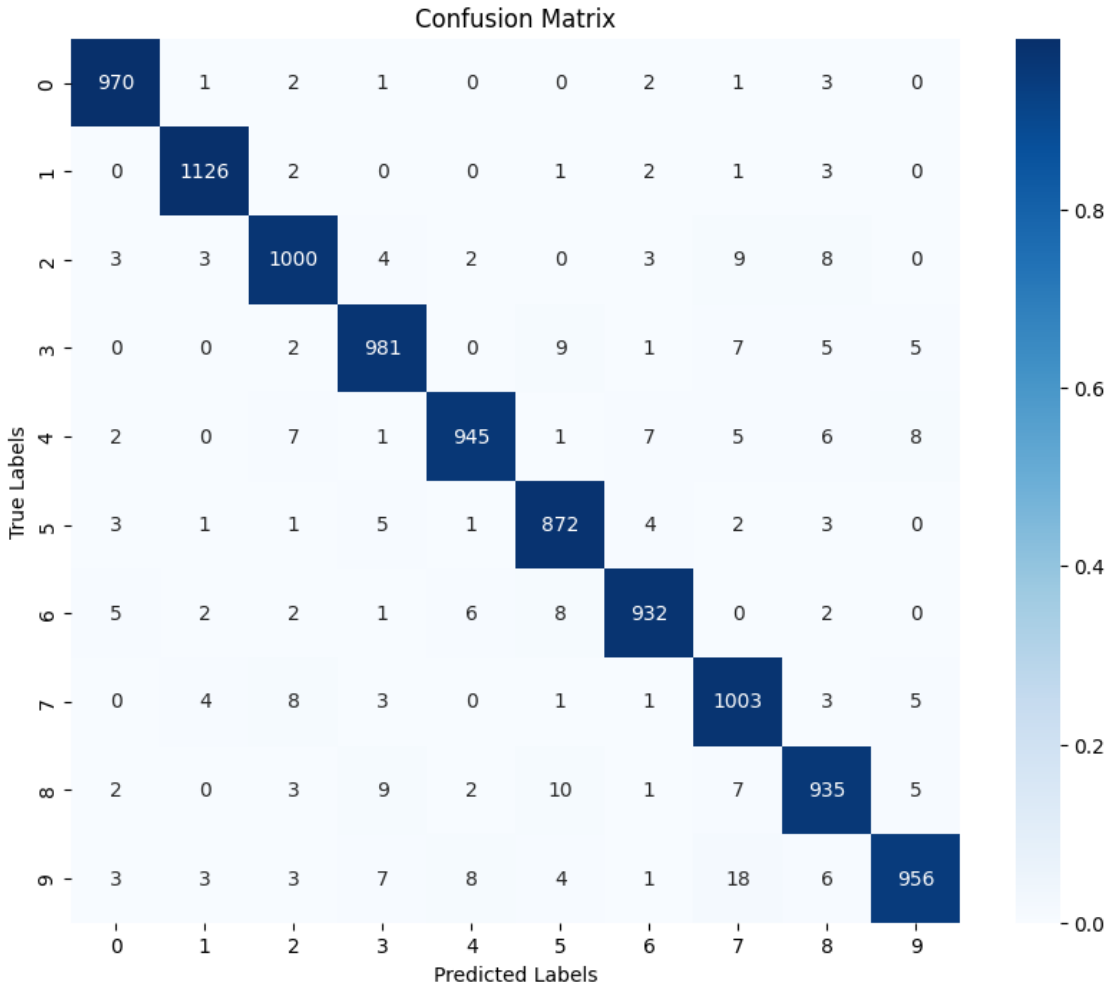
[ ]:
```
best_model_mse_momentum.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.009374661613132869 Train accuracy: 99.52%
Test loss: 0.0449472018655512 Test accuracy: 97.20%

Confusion Matrix:
 [[ 970    1    2    1    0    0    2    1    3    0]
 [   0 1126    2    0    0    1    2    1    3    0]
 [   3    3 1000    4    2    0    3    9    8    0]
 [   0    0    2  981    0    9    1    7    5    5]
 [   2    0    7    1  945    1    7    5    6    8]
 [   3    1    1    5    1  872    4    2    3    0]
 [   5    2    2    1    6    8  932    0    2    0]
 [   0    4    8    3    0    1    1 1003    3    5]
 [   2    0    3    9    2   10    1    7  935    5]
 [   3    3    3    7    8    4    1   18    6  956]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.97      0.97      0.97      1032
           3       0.97      0.97      0.97      1010
           4       0.98      0.96      0.97       982
           5       0.96      0.98      0.97       892
           6       0.98      0.97      0.97       958
           7       0.95      0.98      0.96      1028
           8       0.96      0.96      0.96       974
           9       0.98      0.95      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

[ ]:
```
best_model_mse_momentum.plot_losses()
```



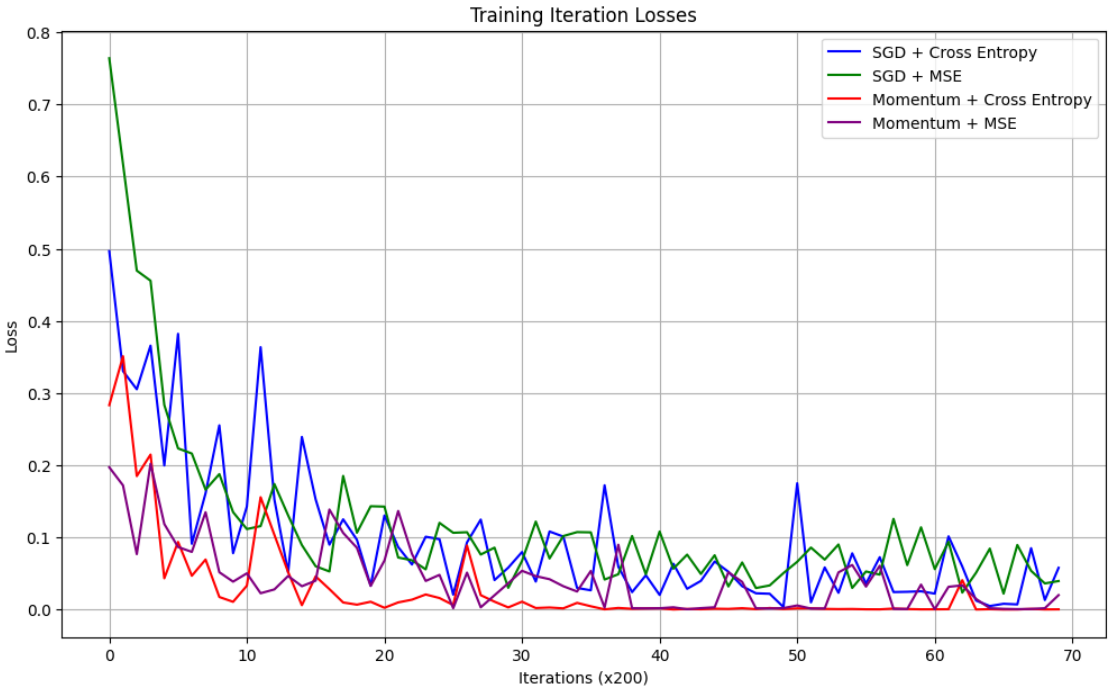
[ ]:
```
best_model_mse_momentum.plot_confusion_matrix(X_test, Y_test)
```

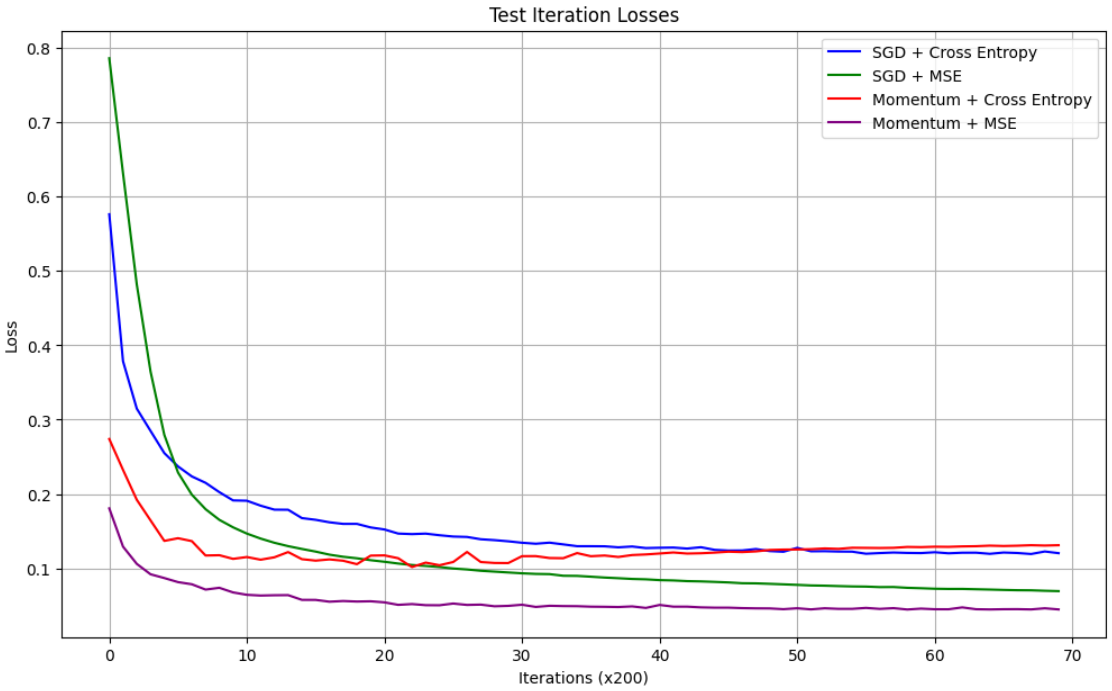


9.3 Comparison

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

def plot_models_losses(models, labels, title, attribute):
    plt.figure(figsize=(12, 7))

    colors = ['blue', 'green', 'red', 'purple']

    for i, model in enumerate(models):
        losses = getattr(model, attribute)
        plt.plot(losses, label=labels[i], color=colors[i])

    plt.xlabel('Iterations (x200)')
    plt.ylabel('Loss')
    plt.title(title)
```

```
    plt.legend()
    plt.grid(True)
    plt.show()

model1 = relu_model
model2 = best_model_mse
model3 = best_model_momentum
model4 = best_model_mse_momentum

models = [model1, model2, model3, model4]
labels = ['SGD + Cross Entropy', 'SGD + MSE', 'Momentum + Cross Entropy',␣
 ↪'Momentum + MSE']

# Plots training iteration losses
plot_models_losses(models, labels, 'Training Iteration Losses',␣
 ↪'iteration_losses')

# Plots test iteration losses
plot_models_losses(models, labels, 'Test Iteration Losses',␣
 ↪'iteration_test_losses')
```





9.4 Observations

Configuration	Test Accuracy	Train Accuracy	Train Loss	Test Loss
SGD + Log Loss	96.91%	99.58%	0.0288	0.1212
SGD + MSE	95.52%	96.85%	0.05435	0.0693
Momentum + Log Loss	97.85%	100.00%	0.00037	0.1320
Momentum + MSE	97.20%	99.52%	0.0094	0.0449

We make two important observations: * The models trained with Cross Entropy as the loss criterion outperform their counterparts trained with mean-sqaured error. * The models trained with MSE as the loss criterion saturate to lower loss values.

We can infer that the Cross Entropy loss is more critical of misclassifications, which is one of the important factors for it leading to better performance. This can be concluded from the fact that the Cross entropy models have higher accuracy even though the loss value observed is more.

Also, the MSE criterion leads to more stable loss plots which can probably linked with the relative smoothness of the loss function itself.

10 PyTorch vs From Scratch Implementation Analysis

[16]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Define the NN architecture
class PyTorchNN(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(PyTorchNN, self).__init__()
        self.fc1 = nn.Linear(input_dim, 500)
        self.fc2 = nn.Linear(500, 250)
        self.fc3 = nn.Linear(250, 100)
        self.fc4 = nn.Linear(100, output_dim)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = torch.relu(self.fc3(x))
        return self.fc4(x)  # No activation here, CrossEntropyLoss adds softmax

model = PyTorchNN(input_dim=28*28, output_dim=10)
optimizer = optim.Adam(model.parameters())
criterion = nn.CrossEntropyLoss()

# Training function
def train_model(model, train_loader, test_loader, epochs=15, log_interval=200):
    iteration_losses = []
    iteration_test_losses = []
    history = []

    model.train()
    iteration_counter = 0

    for epoch in range(epochs):
        tr_loss_epoch, tr_acc = 0, 0
        for batch_idx, (X_t, Y_t) in enumerate(train_loader):
            optimizer.zero_grad()
            preds = model(X_t.view(X_t.size(0), -1))
            loss = criterion(preds, Y_t)
            loss.backward()
            optimizer.step()
```

```
            tr_loss_epoch += loss.item()
            tr_acc += (preds.argmax(1) == Y_t).float().mean().item()

            iteration_counter += 1
            if iteration_counter % log_interval == 0:
                with torch.no_grad():
                    te_loss = sum(criterion(model(X_te.view(X_te.size(0), -1)),␣
↪Y_te) for X_te, Y_te in test_loader)
                    iteration_test_losses.append(te_loss/len(test_loader))
                iteration_losses.append(loss.item())
                print(f"Iteration: {iteration_counter} Train Loss: {loss.
↪item()} Test Loss: {te_loss/len(test_loader)}")

        mean_tr_loss_epoch = tr_loss_epoch / len(train_loader)
        mean_tr_acc = tr_acc / len(train_loader)
        history.append({
            "Epoch": epoch,
            "Train Loss": mean_tr_loss_epoch,
            "Train Accuracy": mean_tr_acc
        })
        print(f"Epoch: {epoch+1} Mean Train Loss: {mean_tr_loss_epoch} Train␣
↪Accuracy: {mean_tr_acc * 100:.2f}%")

    return iteration_losses, iteration_test_losses, history


def evaluate_model(model, train_loader, test_loader, criterion):
    model.eval()


    train_loss, train_correct, train_total = 0.0, 0, 0
    for X, Y in train_loader:
        outputs = model(X.view(X.size(0), -1))
        loss = criterion(outputs, Y)
        train_loss += loss.item() * X.size(0)
        _, predicted = outputs.max(1)
        train_correct += (predicted == Y).sum().item()
        train_total += Y.size(0)

    train_loss /= train_total
    train_accuracy = train_correct / train_total


    test_loss, test_correct, test_total = 0.0, 0, 0
    for X, Y in test_loader:
        outputs = model(X.view(X.size(0), -1))
        loss = criterion(outputs, Y)
```

```
        test_loss += loss.item() * X.size(0)
        _, predicted = outputs.max(1)
        test_correct += (predicted == Y).sum().item()
        test_total += Y.size(0)

    test_loss /= test_total
    test_accuracy = test_correct / test_total

    print(f"Train loss: {train_loss:.4f} Train accuracy: {train_accuracy*100:.
↪2f}%")
    print(f"Test loss: {test_loss:.4f} Test accuracy: {test_accuracy*100:.2f}%")

    test_preds = torch.cat([model(X.view(X.size(0), -1)) for X, _ in␣
↪test_loader])
    test_targets = torch.cat([Y for _, Y in test_loader])
    cm = confusion_matrix(test_targets.numpy(), test_preds.argmax(1).numpy())
    cr = classification_report(test_targets.numpy(), test_preds.argmax(1).
↪numpy())
    print("\nConfusion Matrix:\n", cm)
    print("\nClassification Report:\n", cr)

    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='g', cmap='Blues')
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title('Confusion Matrix')
    plt.show()

    plt.figure(figsize=(10, 5))
    plt.plot(iteration_losses, label='Training Loss', color='blue')
    plt.plot(iteration_test_losses, label='Test Loss', color='red',␣
↪linestyle='--')
    plt.xlabel('Iterations')
    plt.ylabel('Loss')
    plt.title('Loss Convergence')
    plt.legend()
    plt.grid(True)
    plt.show()
```

[ ]:
```
train_dataset = TensorDataset(torch.tensor(X_train, dtype=torch.float), torch.
↪tensor(Y_train, dtype=torch.long))
test_dataset = TensorDataset(torch.tensor(X_test, dtype=torch.float), torch.
↪tensor(Y_test, dtype=torch.long))

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)
```

```
iteration_losses, iteration_test_losses, history = train_model(model,␣
↪train_loader, test_loader)
```

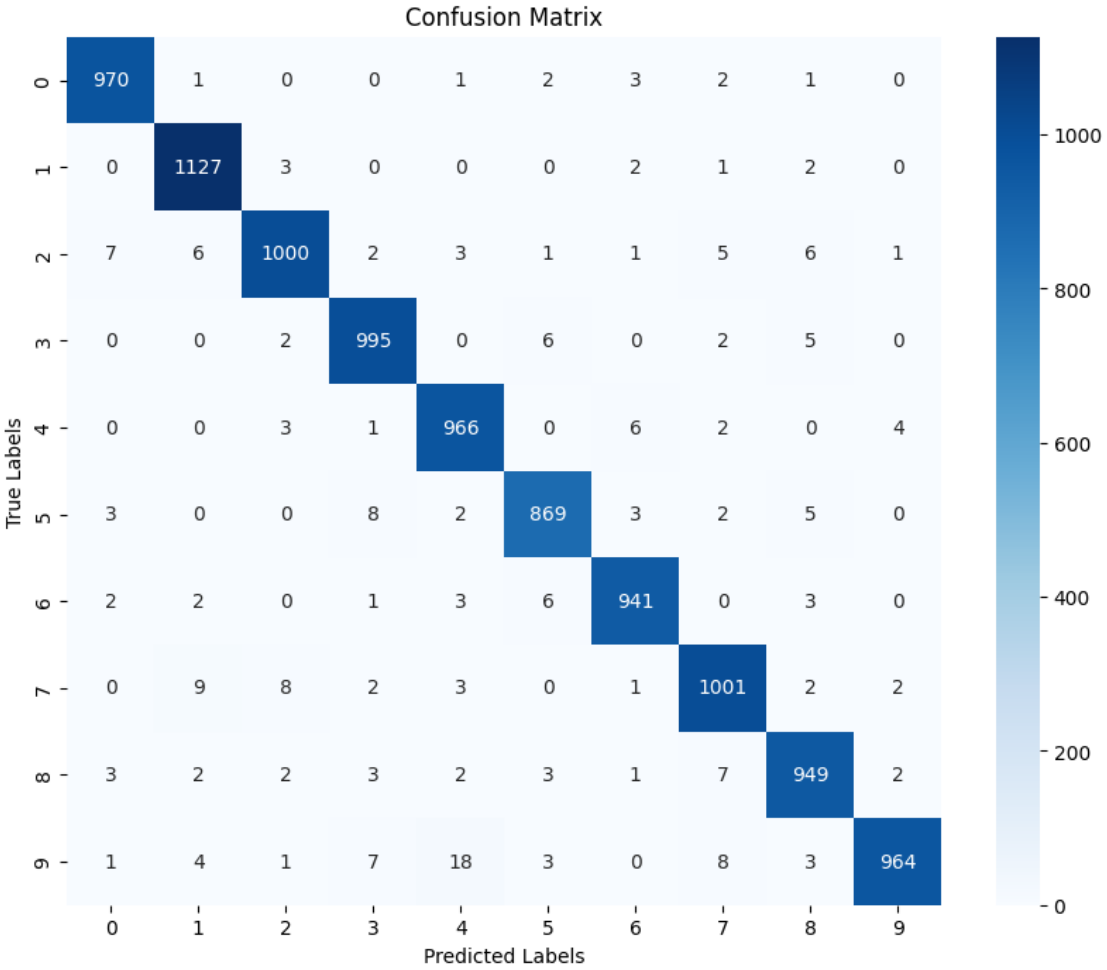
```
Iteration: 200 Train Loss: 0.06082472577691078 Test Loss: 0.22181178629398346
Iteration: 400 Train Loss: 0.04833768680691719 Test Loss: 0.17600177228450775
Iteration: 600 Train Loss: 0.103426992893219 Test Loss: 0.14798253774642944
Iteration: 800 Train Loss: 0.16304656863212585 Test Loss: 0.13436774909496307
Epoch: 1 Mean Train Loss: 0.22737133383615885 Train Accuracy: 93.20%
Iteration: 1000 Train Loss: 0.03831842541694641 Test Loss: 0.14795252680778503
Iteration: 1200 Train Loss: 0.15561886131763458 Test Loss: 0.13509203493595123
Iteration: 1400 Train Loss: 0.17523612082004547 Test Loss: 0.17890894412994385
Iteration: 1600 Train Loss: 0.11719272285699844 Test Loss: 0.17084895074367523
Iteration: 1800 Train Loss: 0.10489985346794128 Test Loss: 0.12421727925539017
Epoch: 2 Mean Train Loss: 0.09823897467064324 Train Accuracy: 97.03%
Iteration: 2000 Train Loss: 0.04964468628168106 Test Loss: 0.12013204395771027
Iteration: 2200 Train Loss: 0.0065956139005720615 Test Loss: 0.13450106978416443
Iteration: 2400 Train Loss: 0.0033921569120138884 Test Loss: 0.12118520587682724
Iteration: 2600 Train Loss: 0.018474765121936798 Test Loss: 0.11425206810235977
Iteration: 2800 Train Loss: 0.032377712428569794 Test Loss: 0.13774296641349792
Epoch: 3 Mean Train Loss: 0.06836337528959997 Train Accuracy: 97.83%
Iteration: 3000 Train Loss: 0.04918832331895828 Test Loss: 0.11908082664012909
Iteration: 3200 Train Loss: 0.049682214856147766 Test Loss: 0.13234791159629822
Iteration: 3400 Train Loss: 0.09597451239824295 Test Loss: 0.1474922001361847
Iteration: 3600 Train Loss: 0.16862526535987854 Test Loss: 0.12312252819538116
Epoch: 4 Mean Train Loss: 0.052449682496176826 Train Accuracy: 98.36%
Iteration: 3800 Train Loss: 0.019973279908299446 Test Loss: 0.12972666323184967
Iteration: 4000 Train Loss: 0.027004629373550415 Test Loss: 0.1291181892156601
Iteration: 4200 Train Loss: 0.10947994142770767 Test Loss: 0.13832023739814758
Iteration: 4400 Train Loss: 0.009926235303282738 Test Loss: 0.12320786714553833
Iteration: 4600 Train Loss: 0.02319205552339554 Test Loss: 0.11712861806154251
Epoch: 5 Mean Train Loss: 0.040905315182970414 Train Accuracy: 98.70%
Iteration: 4800 Train Loss: 0.014959310181438923 Test Loss: 0.11155783385038376
Iteration: 5000 Train Loss: 0.09384650737047195 Test Loss: 0.1346023827791214
Iteration: 5200 Train Loss: 0.0027280733920633793 Test Loss: 0.12169599533081055
Iteration: 5400 Train Loss: 0.006174034439027309 Test Loss: 0.12868931889533997
Iteration: 5600 Train Loss: 0.004179316572844982 Test Loss: 0.1025124341249466
Epoch: 6 Mean Train Loss: 0.03511757170809418 Train Accuracy: 98.92%
Iteration: 5800 Train Loss: 0.006064356304705143 Test Loss: 0.10396204143762589
Iteration: 6000 Train Loss: 0.002017013728618622 Test Loss: 0.11338946223258972
Iteration: 6200 Train Loss: 0.016996918246150017 Test Loss: 0.13352186977863312
Iteration: 6400 Train Loss: 0.055988065898418427 Test Loss: 0.1307571977376938
Epoch: 7 Mean Train Loss: 0.030434907727156776 Train Accuracy: 99.05%
Iteration: 6600 Train Loss: 0.01748274639248848 Test Loss: 0.13396087288856506
Iteration: 6800 Train Loss: 0.0006360544357448816 Test Loss: 0.14049048721790314
Iteration: 7000 Train Loss: 0.0008533213986083865 Test Loss: 0.15218427777290344
Iteration: 7200 Train Loss: 0.023763306438922882 Test Loss: 0.12125767022371292
```

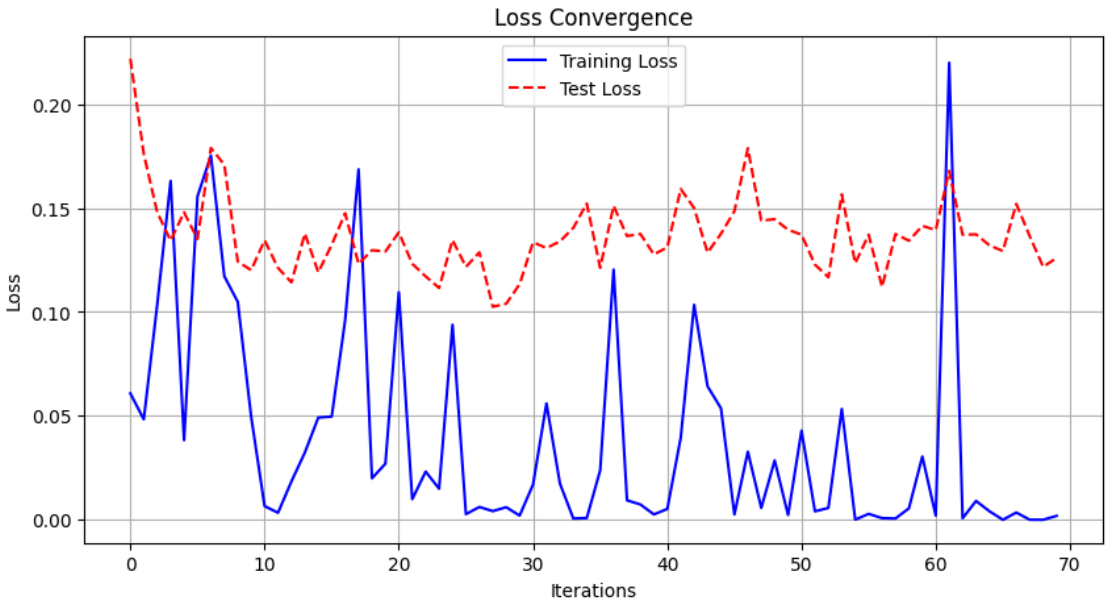
Iteration: 7400 Train Loss: 0.12040121853351593 Test Loss: 0.1510743349790573
Epoch: 8 Mean Train Loss: 0.028526398262259355 Train Accuracy: 99.11%
Iteration: 7600 Train Loss: 0.009370646439492702 Test Loss: 0.13655321300029755
Iteration: 7800 Train Loss: 0.0073816850781440735 Test Loss: 0.13752202689647675
Iteration: 8000 Train Loss: 0.002596559002995491 Test Loss: 0.12768301367759705
Iteration: 8200 Train Loss: 0.005228433758020401 Test Loss: 0.1311102956533432
Iteration: 8400 Train Loss: 0.03948444500565529 Test Loss: 0.15920792520046234
Epoch: 9 Mean Train Loss: 0.025116434766296967 Train Accuracy: 99.26%
Iteration: 8600 Train Loss: 0.10343998670578003 Test Loss: 0.14997223019599915
Iteration: 8800 Train Loss: 0.06423290073871613 Test Loss: 0.12857979536056519
Iteration: 9000 Train Loss: 0.053659308701753616 Test Loss: 0.13760852813720703
Iteration: 9200 Train Loss: 0.002624084008857608 Test Loss: 0.1485091596841812
Epoch: 10 Mean Train Loss: 0.02595645790356946 Train Accuracy: 99.24%
Iteration: 9400 Train Loss: 0.03277468681335449 Test Loss: 0.17878390848636627
Iteration: 9600 Train Loss: 0.005710309371352196 Test Loss: 0.14394491910934448
Iteration: 9800 Train Loss: 0.02856796234846115 Test Loss: 0.1446511298418045
Iteration: 10000 Train Loss: 0.0023106655571609735 Test Loss: 0.1395248919725418
Iteration: 10200 Train Loss: 0.042873531579971313 Test Loss: 0.1372189223766327
Epoch: 11 Mean Train Loss: 0.02297187876496176 Train Accuracy: 99.41%
Iteration: 10400 Train Loss: 0.004089265130460262 Test Loss: 0.12279897183179855
Iteration: 10600 Train Loss: 0.005712785292416811 Test Loss: 0.11671970039606094
Iteration: 10800 Train Loss: 0.05333881825208664 Test Loss: 0.156594380736351
Iteration: 11000 Train Loss: 9.743036207510158e-05 Test Loss:
0.12345176935195923
Iteration: 11200 Train Loss: 0.0028537181206047535 Test Loss:
0.13718561828136444
Epoch: 12 Mean Train Loss: 0.019409418563684378 Train Accuracy: 99.44%
Iteration: 11400 Train Loss: 0.0008151645888574421 Test Loss:
0.11207246780395508
Iteration: 11600 Train Loss: 0.0006391549832187593 Test Loss:
0.13752269744873047
Iteration: 11800 Train Loss: 0.005514831282198429 Test Loss: 0.13429614901542664
Iteration: 12000 Train Loss: 0.030449112877249718 Test Loss: 0.14141802489757538
Epoch: 13 Mean Train Loss: 0.01601693076966644 Train Accuracy: 99.52%
Iteration: 12200 Train Loss: 0.0020337491296231747 Test Loss:
0.13942229747772217
Iteration: 12400 Train Loss: 0.21994638442993164 Test Loss: 0.16794392466545105
Iteration: 12600 Train Loss: 0.0007552322931587696 Test Loss: 0.1371762901544571
Iteration: 12800 Train Loss: 0.009089373983442783 Test Loss: 0.13736434280872345
Iteration: 13000 Train Loss: 0.004108937457203865 Test Loss: 0.1320672631263733
Epoch: 14 Mean Train Loss: 0.018754798646481066 Train Accuracy: 99.47%
Iteration: 13200 Train Loss: 1.9494997104629874e-05 Test Loss:
0.12932385504245758
Iteration: 13400 Train Loss: 0.0034909851383417845 Test Loss: 0.1520252525806427
Iteration: 13600 Train Loss: 4.041140346089378e-05 Test Loss:
0.13649344444274902
Iteration: 13800 Train Loss: 1.2949802112416364e-05 Test Loss:
0.12171926349401474

```
Iteration: 14000 Train Loss: 0.0018992230761796236 Test Loss:
0.12600120902061462
Epoch: 15 Mean Train Loss: 0.015705808024624426 Train Accuracy: 99.55%
```

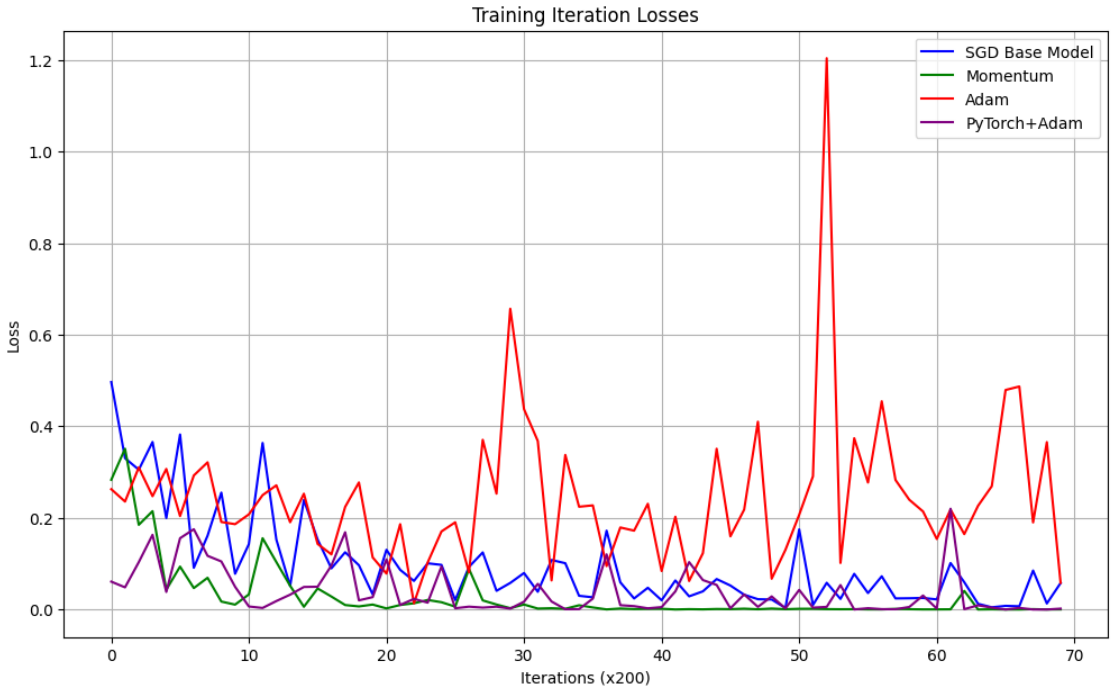
[ ]:
```
evaluate_model(model, train_loader, test_loader, criterion)
```

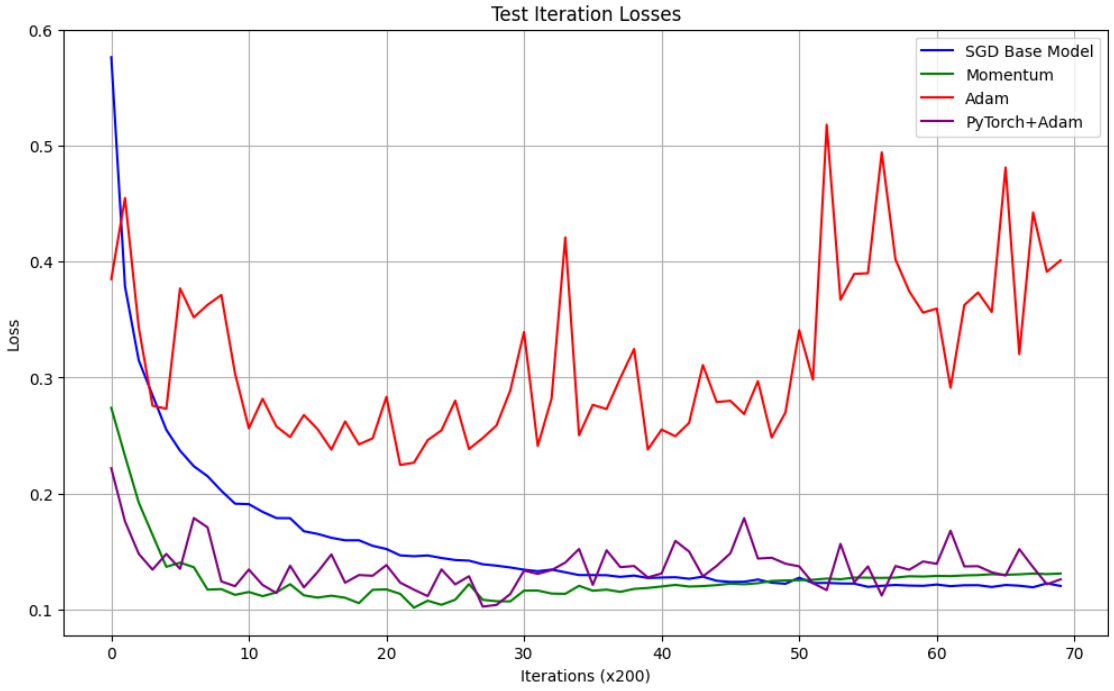
```
Train loss: 0.0122 Train accuracy: 99.63%
Test loss: 0.1311 Test accuracy: 97.82%

Confusion Matrix:
 [[ 970    1    0    0    1    2    3    2    1    0]
 [   0 1127    3    0    0    0    2    1    2    0]
 [   7    6 1000    2    3    1    1    5    6    1]
 [   0    0    2  995    0    6    0    2    5    0]
 [   0    0    3    1  966    0    6    2    0    4]
 [   3    0    0    8    2  869    3    2    5    0]
 [   2    2    0    1    3    6  941    0    3    0]
 [   0    9    8    2    3    0    1 1001    2    2]
 [   3    2    2    3    2    3    1    7  949    2]
 [   1    4    1    7   18    3    0    8    3  964]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.98      0.99      0.99      1135
           2       0.98      0.97      0.98      1032
           3       0.98      0.99      0.98      1010
           4       0.97      0.98      0.98       982
           5       0.98      0.97      0.98       892
           6       0.98      0.98      0.98       958
           7       0.97      0.97      0.97      1028
           8       0.97      0.97      0.97       974
           9       0.99      0.96      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```


Confusion Matrix
True Labels
Predicted Labels
0 970 1 0 0 1 2 3 2 1 0
1 0 1127 3 0 0 0 2 1 2 0
2 7 6 1000 2 3 1 1 5 6 1
3 0 0 2 995 0 6 0 2 5 0
4 0 0 3 1 966 0 6 2 0 4
5 3 0 0 8 2 869 3 2 5 0
6 2 2 0 1 3 6 941 0 3 0
7 0 9 8 2 3 0 1 1001 2 2
8 3 2 2 3 2 3 1 7 949 2
9 1 4 1 7 18 3 0 8 3 964
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0


Loss Convergence
Training Loss
Test Loss
Loss
Iterations
0.20
0.15
0.10
0.05
0.00
0
10
20
30
40
50
60
70

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

def plot_models_losses(losses_list, labels, title):
    plt.figure(figsize=(12, 7))

    colors = ['blue', 'green', 'red', 'purple']

    for i, losses in enumerate(losses_list):
        plt.plot(losses, label=labels[i], color=colors[i])

    plt.xlabel('Iterations (x200)')
    plt.ylabel('Loss')
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.show()

model1 = relu_model
model2 = best_model_momentum
model3 = best_model_adam


training_losses = [
    model1.iteration_losses,
    model2.iteration_losses,
    model3.iteration_losses,
    iteration_losses
]

test_losses = [
    model1.iteration_test_losses,
    model2.iteration_test_losses,
    model3.iteration_test_losses,
    iteration_test_losses
]

labels = ['SGD Base Model', 'Momentum', 'Adam', 'PyTorch+Adam']

# Plots training iteration losses
plot_models_losses(training_losses, labels, 'Training Iteration Losses')

# Plots test iteration losses
plot_models_losses(test_losses, labels, 'Test Iteration Losses')
```


Training Iteration Losses
SGD Base Model
Momentum
Adam
PyTorch+Adam
Loss
Iterations (x200)


Test Iteration Losses
SGD Base Model
Momentum
Adam
PyTorch+Adam
Loss
Iterations (x200)

10.1 Observations

The training is much faster when we use the PyTorch implementation. This version also gives the best performance among all the models we have tried so far. PyTorch has optimized low-level implementations for many tensor operations. These optimizations might include more efficient memory access patterns, fused operations, and other tricks that a manual numpy implementation might not take advantage of. PyTorch provides automatic differentiation, which ensures that the gradients are computed correctly and efficiently. Implementing backpropagation manually in numpy can be error-prone and might not be as optimized as PyTorch's autograd. Here is a summary to compare results:

Configuration	Test Accuracy	Train Accuracy	Train Loss	Test Loss
SGD	96.91%	99.58%	0.0228	0.1212
Momentum	97.85%	100.00%	0.00037	0.1320
NumPy+Adam	92.61%	93.32%	0.2848	0.3577
PyTorch+Adam	97.82%	99.63%	0.0122	0.1311

Suprisingly, the momentum implementation outperforms the PyTorch implementation of Adam as well. Some research shows that momentum based gradient descent introduces a certain level of implicit noise into the weight updates. This noise can act as a form of regularization, which can sometimes help in generalization.[1][2] Adam, with its adaptive learning rates for each parameter, can sometimes overfit to the training data, especially if the model capacity is high and dropout or other regularization methods are not used.

11 Weight Decay Effects

11.1 Weight Decay 0.5

11.1.1 SGD

[18]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.5),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.5),
    ReLU(),
    FNNLayer(250, 100, weight_decay = 0.5),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.5)
]

sgd_model_halfalpha = NN(layers)
optimizer = SGD()
loss = LogLoss()
sgd_model_halfalpha.compile(loss=loss, optimizer=optimizer)
```

```
sgd_model_halfalpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 2.3017919715727944 Test Loss: 2.3017244501048695
Iteration: 400 Train Loss: 2.3024163265998636 Test Loss: 2.302314815295173
Iteration: 600 Train Loss: 2.3031309135419322 Test Loss: 2.3022845333623443
Iteration: 800 Train Loss: 2.3018953289929494 Test Loss: 2.302310788233524
Epoch: 1 Mean Train Loss: 2.2929790709290208 Train Accuracy: 16.45%
Iteration: 1000 Train Loss: 2.3017447394304726 Test Loss: 2.3023281484590545
Iteration: 1200 Train Loss: 2.3028537809799667 Test Loss: 2.302278164931032
Iteration: 1400 Train Loss: 2.3029433094736302 Test Loss: 2.3022947300001064
Iteration: 1600 Train Loss: 2.3023761139010404 Test Loss: 2.302350756875167
Iteration: 1800 Train Loss: 2.302936908270609 Test Loss: 2.3024108006116917
Epoch: 2 Mean Train Loss: 2.3023601550211237 Train Accuracy: 11.18%
Iteration: 2000 Train Loss: 2.302970889131903 Test Loss: 2.30235481137353
Iteration: 2200 Train Loss: 2.303656013843194 Test Loss: 2.302281349213606
Iteration: 2400 Train Loss: 2.301983699322089 Test Loss: 2.30234303048243
Iteration: 2600 Train Loss: 2.301599574912692 Test Loss: 2.3023815664682687
Iteration: 2800 Train Loss: 2.302418499703153 Test Loss: 2.302282453305738
Epoch: 3 Mean Train Loss: 2.3023619365598313 Train Accuracy: 11.18%
Iteration: 3000 Train Loss: 2.3027204744921255 Test Loss: 2.3022884672003068
Iteration: 3200 Train Loss: 2.302083850796489 Test Loss: 2.302362316358651
Iteration: 3400 Train Loss: 2.30218220728723 Test Loss: 2.302276787688777
Iteration: 3600 Train Loss: 2.3022139039264626 Test Loss: 2.302398676505665
Epoch: 4 Mean Train Loss: 2.3023409801864725 Train Accuracy: 11.23%
Iteration: 3800 Train Loss: 2.3017023758892803 Test Loss: 2.3023141203301387
Iteration: 4000 Train Loss: 2.3029332276898753 Test Loss: 2.3023858074947654
Iteration: 4200 Train Loss: 2.3013522509804805 Test Loss: 2.302312474316256
Iteration: 4400 Train Loss: 2.302752024546586 Test Loss: 2.3023126603714985
Iteration: 4600 Train Loss: 2.3022050307640667 Test Loss: 2.30234588453997
Epoch: 5 Mean Train Loss: 2.302351161938208 Train Accuracy: 11.24%
Iteration: 4800 Train Loss: 2.302743705993243 Test Loss: 2.302290650555317
Iteration: 5000 Train Loss: 2.3026011109349636 Test Loss: 2.3023435157090493
Iteration: 5200 Train Loss: 2.3036056143639874 Test Loss: 2.302348410995908
Iteration: 5400 Train Loss: 2.301384398157329 Test Loss: 2.3023957409663596
Iteration: 5600 Train Loss: 2.3014471278069735 Test Loss: 2.302308480241021
Epoch: 6 Mean Train Loss: 2.3023348861618738 Train Accuracy: 11.20%
Iteration: 5800 Train Loss: 2.3036048798315267 Test Loss: 2.3022984732968688
Iteration: 6000 Train Loss: 2.301728163832392 Test Loss: 2.3022894710902273
Iteration: 6200 Train Loss: 2.301814355136652 Test Loss: 2.3023257031506854
Iteration: 6400 Train Loss: 2.3031877578902673 Test Loss: 2.302308574552227
Epoch: 7 Mean Train Loss: 2.3023458306274556 Train Accuracy: 11.22%
Iteration: 6600 Train Loss: 2.301249582057335 Test Loss: 2.3022616489138543
Iteration: 6800 Train Loss: 2.302765538597965 Test Loss: 2.3023415167016736
Iteration: 7000 Train Loss: 2.3024286883293024 Test Loss: 2.3023588026206747
Iteration: 7200 Train Loss: 2.3024717298238535 Test Loss: 2.302301832746408
Iteration: 7400 Train Loss: 2.3030304990981385 Test Loss: 2.302298681822644
Epoch: 8 Mean Train Loss: 2.302350301396244 Train Accuracy: 11.16%
```

```
Iteration: 7600 Train Loss: 2.3029912946633004 Test Loss: 2.3023534127285163
Iteration: 7800 Train Loss: 2.302235818780205 Test Loss: 2.30235063277561
Iteration: 8000 Train Loss: 2.302805894818955 Test Loss: 2.3023542732912787
Iteration: 8200 Train Loss: 2.302895295929452 Test Loss: 2.302356256121973
Iteration: 8400 Train Loss: 2.3028401415289808 Test Loss: 2.3023010898301695
Epoch: 9 Mean Train Loss: 2.3023601337527753 Train Accuracy: 11.22%
Iteration: 8600 Train Loss: 2.3028640595963408 Test Loss: 2.3023099582797166
Iteration: 8800 Train Loss: 2.30171104662307 Test Loss: 2.3022730856873137
Iteration: 9000 Train Loss: 2.3034433175462588 Test Loss: 2.30231253872902
Iteration: 9200 Train Loss: 2.301853152997509 Test Loss: 2.3023627336644585
Epoch: 10 Mean Train Loss: 2.302357965385579 Train Accuracy: 11.22%
Iteration: 9400 Train Loss: 2.301556958862193 Test Loss: 2.302310513457736
Iteration: 9600 Train Loss: 2.3018730642456298 Test Loss: 2.3022733880131194
Iteration: 9800 Train Loss: 2.3024496633862936 Test Loss: 2.302373735136005
Iteration: 10000 Train Loss: 2.302295303769162 Test Loss: 2.3023027114251975
Iteration: 10200 Train Loss: 2.303138023382799 Test Loss: 2.302397221627009
Epoch: 11 Mean Train Loss: 2.302359761150262 Train Accuracy: 11.24%
Iteration: 10400 Train Loss: 2.3023412019678386 Test Loss: 2.3023124543078985
Iteration: 10600 Train Loss: 2.301489717950786 Test Loss: 2.3022902126855986
Iteration: 10800 Train Loss: 2.3020893450105264 Test Loss: 2.302330930000897
Iteration: 11000 Train Loss: 2.302304301645366 Test Loss: 2.302327664504594
Iteration: 11200 Train Loss: 2.3024854842405884 Test Loss: 2.3023985960276736
Epoch: 12 Mean Train Loss: 2.3023575316656966 Train Accuracy: 11.23%
Iteration: 11400 Train Loss: 2.301829563325045 Test Loss: 2.3022897941999814
Iteration: 11600 Train Loss: 2.302771435666576 Test Loss: 2.3024074778432757
Iteration: 11800 Train Loss: 2.301985848298574 Test Loss: 2.3022683374866433
Iteration: 12000 Train Loss: 2.301907652887854 Test Loss: 2.3023681449219358
Epoch: 13 Mean Train Loss: 2.3023596667336195 Train Accuracy: 11.21%
Iteration: 12200 Train Loss: 2.3013528804510175 Test Loss: 2.3023186356810323
Iteration: 12400 Train Loss: 2.302520969478443 Test Loss: 2.302276536734473
Iteration: 12600 Train Loss: 2.3006615364668344 Test Loss: 2.3022541466187647
Iteration: 12800 Train Loss: 2.302257502438624 Test Loss: 2.3023731269442114
Iteration: 13000 Train Loss: 2.301110794951851 Test Loss: 2.302340483732805
Epoch: 14 Mean Train Loss: 2.302347156917246 Train Accuracy: 11.24%
Iteration: 13200 Train Loss: 2.3026351786652515 Test Loss: 2.302260063493672
Iteration: 13400 Train Loss: 2.3025354720416678 Test Loss: 2.3022405664540195
Iteration: 13600 Train Loss: 2.302743007749469 Test Loss: 2.3023534460496737
Iteration: 13800 Train Loss: 2.3012320423916672 Test Loss: 2.302317201718603
Iteration: 14000 Train Loss: 2.301963029393092 Test Loss: 2.3023069574970028
Epoch: 15 Mean Train Loss: 2.3023532429548714 Train Accuracy: 11.22%
Model trained!
```

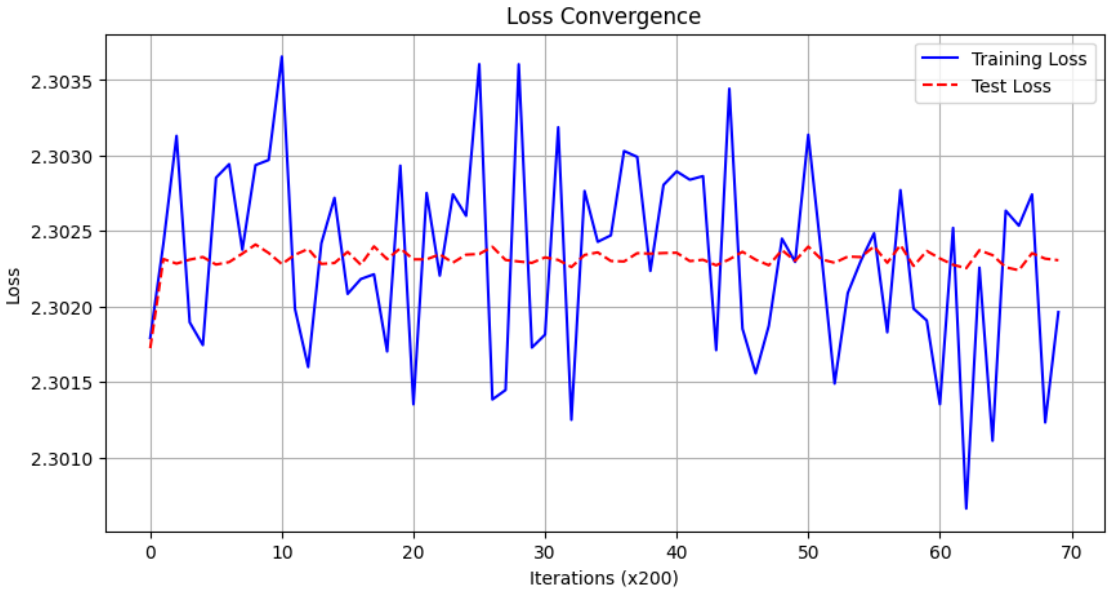
[19]:
```
sgd_model_halfalpha.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 2.3023430208775486 Train accuracy: 10.44%
Test loss: 2.3023533700948295 Test accuracy: 10.28%

Confusion Matrix:
```

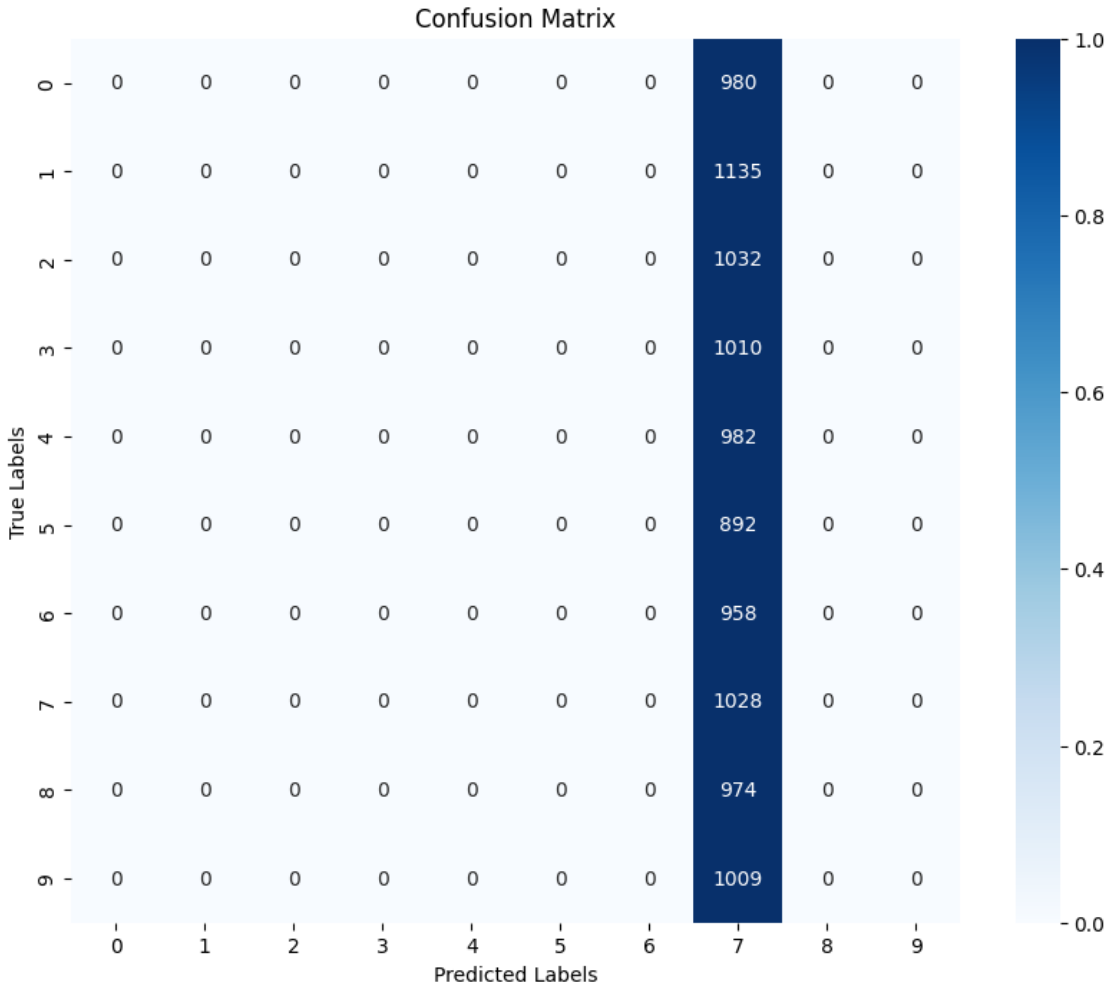
```
[[   0    0    0    0    0    0    0  980    0    0]
 [   0    0    0    0    0    0    0 1135    0    0]
 [   0    0    0    0    0    0    0 1032    0    0]
 [   0    0    0    0    0    0    0 1010    0    0]
 [   0    0    0    0    0    0    0  982    0    0]
 [   0    0    0    0    0    0    0  892    0    0]
 [   0    0    0    0    0    0    0  958    0    0]
 [   0    0    0    0    0    0    0 1028    0    0]
 [   0    0    0    0    0    0    0  974    0    0]
 [   0    0    0    0    0    0    0 1009    0    0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       980
           1       0.00      0.00      0.00      1135
           2       0.00      0.00      0.00      1032
           3       0.00      0.00      0.00      1010
           4       0.00      0.00      0.00       982
           5       0.00      0.00      0.00       892
           6       0.00      0.00      0.00       958
           7       0.10      1.00      0.19      1028
           8       0.00      0.00      0.00       974
           9       0.00      0.00      0.00      1009

    accuracy                           0.10     10000
   macro avg       0.01      0.10      0.02     10000
weighted avg       0.01      0.10      0.02     10000

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[20]:

```
sgd_model_halfalpha.plot_losses()
```



[21]:
```
sgd_model_halfalpha.plot_confusion_matrix(X_test, Y_test)
```



11.1.2 Momentum

[22]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.5),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.5),
    ReLU(),
    FNNLayer(250, 100, weight_decay = 0.5),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.5)
]

momentum_model_halfalpha = NN(layers)
```

```
optimizer = Momentum()
loss = LogLoss()
momentum_model_halfalpha.compile(loss=loss, optimizer=optimizer)
momentum_model_halfalpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 2.3028357288744203 Test Loss: 2.302156660144446
Iteration: 400 Train Loss: 2.3026408000847898 Test Loss: 2.3024215819876974
Iteration: 600 Train Loss: 2.302070425932069 Test Loss: 2.302229776765459
Iteration: 800 Train Loss: 2.3011183873611447 Test Loss: 2.3024022742762194
Epoch: 1 Mean Train Loss: 2.3019840449955633 Train Accuracy: 11.06%
Iteration: 1000 Train Loss: 2.3031027044479275 Test Loss: 2.302523207132461
Iteration: 1200 Train Loss: 2.301694761205412 Test Loss: 2.302426776972655
Iteration: 1400 Train Loss: 2.3028822432020304 Test Loss: 2.30213394825567
Iteration: 1600 Train Loss: 2.303119232528476 Test Loss: 2.3023716967009307
Iteration: 1800 Train Loss: 2.3011491881546022 Test Loss: 2.3021800933645067
Epoch: 2 Mean Train Loss: 2.3024167216964835 Train Accuracy: 10.48%
Iteration: 2000 Train Loss: 2.301398027581473 Test Loss: 2.302325313190977
Iteration: 2200 Train Loss: 2.3027913291734046 Test Loss: 2.3024811653075585
Iteration: 2400 Train Loss: 2.302519352744127 Test Loss: 2.3022882680936645
Iteration: 2600 Train Loss: 2.302214281263187 Test Loss: 2.302395725858685
Iteration: 2800 Train Loss: 2.3021542849315426 Test Loss: 2.302192266338214
Epoch: 3 Mean Train Loss: 2.302421974350497 Train Accuracy: 10.45%
Iteration: 3000 Train Loss: 2.3030793719338982 Test Loss: 2.302517873740374
Iteration: 3200 Train Loss: 2.3009791683699428 Test Loss: 2.302304627147719
Iteration: 3400 Train Loss: 2.302203140610879 Test Loss: 2.3025490076144033
Iteration: 3600 Train Loss: 2.302397252496835 Test Loss: 2.302349801287517
Epoch: 4 Mean Train Loss: 2.3024306021591077 Train Accuracy: 10.42%
Iteration: 3800 Train Loss: 2.303063055742045 Test Loss: 2.3024172667115317
Iteration: 4000 Train Loss: 2.3035460440770175 Test Loss: 2.3021712828971572
Iteration: 4200 Train Loss: 2.299970672789192 Test Loss: 2.3020953655806338
Iteration: 4400 Train Loss: 2.301961429013009 Test Loss: 2.3022784260726747
Iteration: 4600 Train Loss: 2.303137508913005 Test Loss: 2.3023317072325424
Epoch: 5 Mean Train Loss: 2.30240347925941 Train Accuracy: 10.49%
Iteration: 4800 Train Loss: 2.303695201913608 Test Loss: 2.302077013391415
Iteration: 5000 Train Loss: 2.3016655561418844 Test Loss: 2.3022674558289444
Iteration: 5200 Train Loss: 2.3041517428659124 Test Loss: 2.302212570551276
Iteration: 5400 Train Loss: 2.300547393937057 Test Loss: 2.3022806121340262
Iteration: 5600 Train Loss: 2.302030185774038 Test Loss: 2.302218595582889
Epoch: 6 Mean Train Loss: 2.3022989991876934 Train Accuracy: 10.76%
Iteration: 5800 Train Loss: 2.3031179900364185 Test Loss: 2.30250625534041
Iteration: 6000 Train Loss: 2.3014223493528925 Test Loss: 2.3024132647638496
Iteration: 6200 Train Loss: 2.303352673507862 Test Loss: 2.3024367977255045
Iteration: 6400 Train Loss: 2.3037508896417336 Test Loss: 2.3023794763573684
Epoch: 7 Mean Train Loss: 2.302323425562426 Train Accuracy: 10.72%
Iteration: 6600 Train Loss: 2.3029915410616297 Test Loss: 2.302340685576936
Iteration: 6800 Train Loss: 2.3025892830769505 Test Loss: 2.302256501480371
Iteration: 7000 Train Loss: 2.3028267669912985 Test Loss: 2.3025747046885874
```

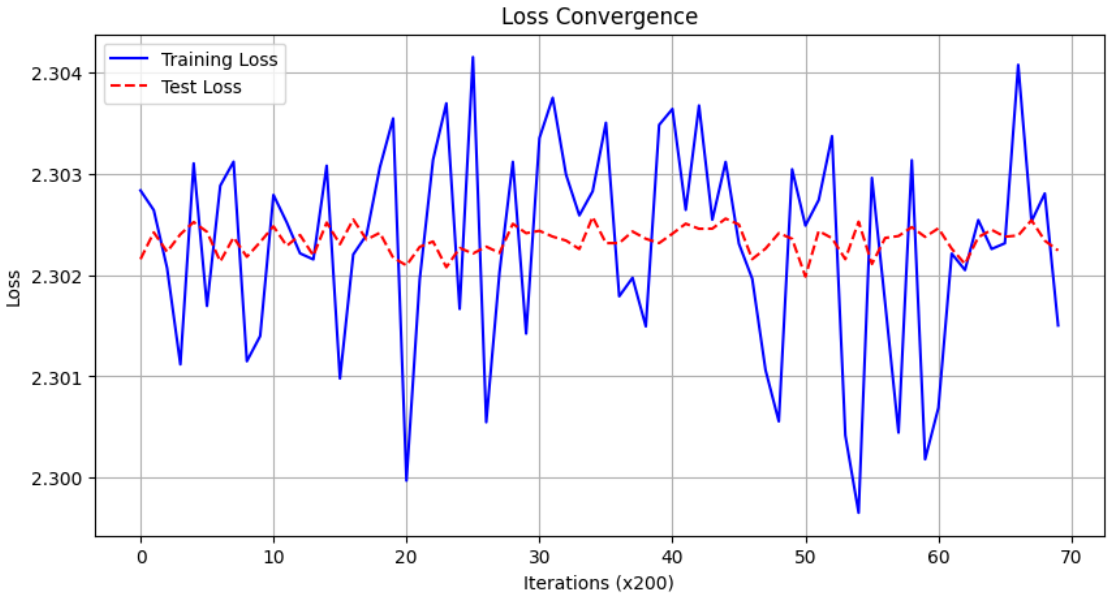
```
Iteration: 7200 Train Loss: 2.3035033652681713 Test Loss: 2.3023151409524605
Iteration: 7400 Train Loss: 2.3017903348577224 Test Loss: 2.302315593884123
Epoch: 8 Mean Train Loss: 2.302385282950612 Train Accuracy: 10.78%
Iteration: 7600 Train Loss: 2.301973517903684 Test Loss: 2.302430573493896
Iteration: 7800 Train Loss: 2.3014932696792907 Test Loss: 2.3023582137775636
Iteration: 8000 Train Loss: 2.303482190364634 Test Loss: 2.3023156169916437
Iteration: 8200 Train Loss: 2.303640000317605 Test Loss: 2.302408466441715
Iteration: 8400 Train Loss: 2.302643928670726 Test Loss: 2.302507303035961
Epoch: 9 Mean Train Loss: 2.3024091257941603 Train Accuracy: 10.43%
Iteration: 8600 Train Loss: 2.303673810627661 Test Loss: 2.302457038097321
Iteration: 8800 Train Loss: 2.3025459114675195 Test Loss: 2.3024581949948923
Iteration: 9000 Train Loss: 2.303117365636826 Test Loss: 2.3025572760523283
Iteration: 9200 Train Loss: 2.3023097681575164 Test Loss: 2.302498823459637
Epoch: 10 Mean Train Loss: 2.3024027501035307 Train Accuracy: 10.49%
Iteration: 9400 Train Loss: 2.3019640133321944 Test Loss: 2.3021571394970937
Iteration: 9600 Train Loss: 2.3010632722344666 Test Loss: 2.302260939447395
Iteration: 9800 Train Loss: 2.3005557721035874 Test Loss: 2.302413416918024
Iteration: 10000 Train Loss: 2.3030443643722647 Test Loss: 2.3023580109319015
Iteration: 10200 Train Loss: 2.302488257874133 Test Loss: 2.301985377701643
Epoch: 11 Mean Train Loss: 2.3024055563469474 Train Accuracy: 10.60%
Iteration: 10400 Train Loss: 2.302740933447902 Test Loss: 2.3024355187368144
Iteration: 10600 Train Loss: 2.3033709829908053 Test Loss: 2.3023638083232845
Iteration: 10800 Train Loss: 2.3004183482597518 Test Loss: 2.3021564874300378
Iteration: 11000 Train Loss: 2.299652974230691 Test Loss: 2.3025270966285483
Iteration: 11200 Train Loss: 2.3029596411129685 Test Loss: 2.302109295236208
Epoch: 12 Mean Train Loss: 2.3023600129269792 Train Accuracy: 10.68%
Iteration: 11400 Train Loss: 2.301718733976613 Test Loss: 2.3023668451542103
Iteration: 11600 Train Loss: 2.300444844937325 Test Loss: 2.3023852078115263
Iteration: 11800 Train Loss: 2.3031340910724207 Test Loss: 2.3024732957858927
Iteration: 12000 Train Loss: 2.300181518043325 Test Loss: 2.3023766982375613
Epoch: 13 Mean Train Loss: 2.3023982845005055 Train Accuracy: 10.70%
Iteration: 12200 Train Loss: 2.3006900440521383 Test Loss: 2.302461684392212
Iteration: 12400 Train Loss: 2.3022121196187504 Test Loss: 2.3022602944129775
Iteration: 12600 Train Loss: 2.3020494339333935 Test Loss: 2.3021096148923528
Iteration: 12800 Train Loss: 2.302543411298277 Test Loss: 2.302373929144136
Iteration: 13000 Train Loss: 2.302257315262663 Test Loss: 2.3024467962806576
Epoch: 14 Mean Train Loss: 2.302422005460032 Train Accuracy: 10.61%
Iteration: 13200 Train Loss: 2.302314149740933 Test Loss: 2.3023786111170774
Iteration: 13400 Train Loss: 2.3040752563020988 Test Loss: 2.302390908997469
Iteration: 13600 Train Loss: 2.3025294978735147 Test Loss: 2.302541648342272
Iteration: 13800 Train Loss: 2.3028059679487933 Test Loss: 2.3023396480308467
Iteration: 14000 Train Loss: 2.3015024717182384 Test Loss: 2.3022430473958364
Epoch: 15 Mean Train Loss: 2.302386108704044 Train Accuracy: 10.43%
Model trained!
```

[23]:
```
momentum_model_halfalpha.evaluate(X_train, Y_train, X_test, Y_test)
```

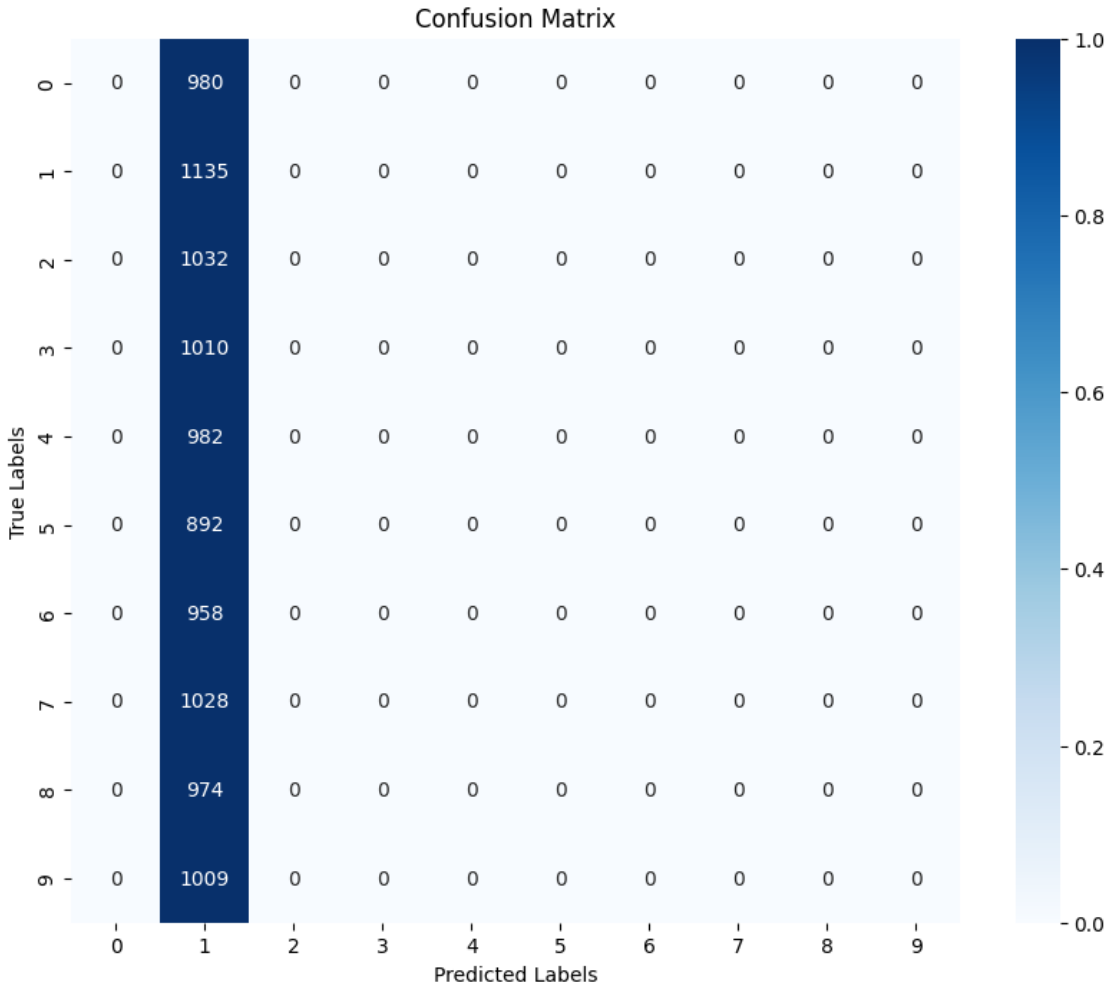
```
Train loss: 2.3025131587720344 Train accuracy: 11.24%
```

```
Test loss: 2.302426730233801 Test accuracy: 11.35%

Confusion Matrix:
 [[   0  980    0    0    0    0    0    0    0    0]
 [   0 1135    0    0    0    0    0    0    0    0]
 [   0 1032    0    0    0    0    0    0    0    0]
 [   0 1010    0    0    0    0    0    0    0    0]
 [   0  982    0    0    0    0    0    0    0    0]
 [   0  892    0    0    0    0    0    0    0    0]
 [   0  958    0    0    0    0    0    0    0    0]
 [   0 1028    0    0    0    0    0    0    0    0]
 [   0  974    0    0    0    0    0    0    0    0]
 [   0 1009    0    0    0    0    0    0    0    0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       980
           1       0.11      1.00      0.20      1135
           2       0.00      0.00      0.00      1032
           3       0.00      0.00      0.00      1010
           4       0.00      0.00      0.00       982
           5       0.00      0.00      0.00       892
           6       0.00      0.00      0.00       958
           7       0.00      0.00      0.00      1028
           8       0.00      0.00      0.00       974
           9       0.00      0.00      0.00      1009

    accuracy                           0.11     10000
   macro avg       0.01      0.10      0.02     10000
weighted avg       0.01      0.11      0.02     10000

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[24]:
```
momentum_model_halfalpha.plot_losses()
```



[25]:
```
momentum_model_halfalpha.plot_confusion_matrix(X_test, Y_test)
```



11.1.3 Conclusion

Such a high regularization term doesn't let the models learn at all. Most didgits are classified as either 1 (Momentum) or 7 (GD), showing the bias introduced by the regularization term in the model.

11.2 Weight Decay 0.1

11.2.1 SGD

[29]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.1),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.1),
```

```
    ReLU(),
    FNNLayer(250, 100, weight_decay = 0.1),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.1)
]

sgd_model_1by10alpha = NN(layers)
optimizer = SGD()
loss = LogLoss()
sgd_model_1by10alpha.compile(loss=loss, optimizer=optimizer)
sgd_model_1by10alpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 1.367842782262861 Test Loss: 1.3121800212745725
Iteration: 400 Train Loss: 1.3229758630084394 Test Loss: 1.306041785583772
Iteration: 600 Train Loss: 1.4534855568941418 Test Loss: 1.3861890767250347
Iteration: 800 Train Loss: 1.4995705179141627 Test Loss: 1.451012067373891
Epoch: 1 Mean Train Loss: 1.4329701540684647 Train Accuracy: 66.94%
Iteration: 1000 Train Loss: 1.4163052282825659 Test Loss: 1.492545427602849
Iteration: 1200 Train Loss: 1.5545758052792824 Test Loss: 1.5281121003488751
Iteration: 1400 Train Loss: 1.488846803071123 Test Loss: 1.5338520274881131
Iteration: 1600 Train Loss: 1.5935913840145028 Test Loss: 1.546415191576921
Iteration: 1800 Train Loss: 1.7164191432838272 Test Loss: 1.552775285541667
Epoch: 2 Mean Train Loss: 1.5380085006845625 Train Accuracy: 48.71%
Iteration: 2000 Train Loss: 1.5873631491164826 Test Loss: 1.5556573089504095
Iteration: 2200 Train Loss: 1.5171323246188277 Test Loss: 1.563399133041899
Iteration: 2400 Train Loss: 1.620177469333624 Test Loss: 1.5623530776924384
Iteration: 2600 Train Loss: 1.6051899314000049 Test Loss: 1.5708070176075193
Iteration: 2800 Train Loss: 1.48330534720781 Test Loss: 1.5621056462865022
Epoch: 3 Mean Train Loss: 1.5640604685483372 Train Accuracy: 42.68%
Iteration: 3000 Train Loss: 1.6138066125571036 Test Loss: 1.5622825255850696
Iteration: 3200 Train Loss: 1.509233182515696 Test Loss: 1.5655733022337561
Iteration: 3400 Train Loss: 1.5485449958905315 Test Loss: 1.5643808288399477
Iteration: 3600 Train Loss: 1.5175005380309199 Test Loss: 1.5655021592580096
Epoch: 4 Mean Train Loss: 1.5658062264788941 Train Accuracy: 41.84%
Iteration: 3800 Train Loss: 1.410913377857665 Test Loss: 1.5628848216282765
Iteration: 4000 Train Loss: 1.5801876071730527 Test Loss: 1.5627082371453451
Iteration: 4200 Train Loss: 1.4914057453814005 Test Loss: 1.562780031147761
Iteration: 4400 Train Loss: 1.5562955106454268 Test Loss: 1.562995034420668
Iteration: 4600 Train Loss: 1.5291641617585108 Test Loss: 1.5576505854964342
Epoch: 5 Mean Train Loss: 1.5639014715428767 Train Accuracy: 41.78%
Iteration: 4800 Train Loss: 1.3984859675743178 Test Loss: 1.5582178388902355
Iteration: 5000 Train Loss: 1.5401100694603294 Test Loss: 1.5535444040456006
Iteration: 5200 Train Loss: 1.5852044201971698 Test Loss: 1.559685226363012
Iteration: 5400 Train Loss: 1.5758740917684082 Test Loss: 1.5562438316049518
Iteration: 5600 Train Loss: 1.7111707210413194 Test Loss: 1.5577999154154734
Epoch: 6 Mean Train Loss: 1.558660212966716 Train Accuracy: 42.27%
Iteration: 5800 Train Loss: 1.5980677185483267 Test Loss: 1.5590063205080404
```

```
Iteration: 6000 Train Loss: 1.6123621032919702 Test Loss: 1.5534708779792912
Iteration: 6200 Train Loss: 1.5876763199153219 Test Loss: 1.5515157431787254
Iteration: 6400 Train Loss: 1.3179707040043112 Test Loss: 1.5500432261400667
Epoch: 7 Mean Train Loss: 1.555566963872417 Train Accuracy: 42.51%
Iteration: 6600 Train Loss: 1.7292581177703636 Test Loss: 1.5521013425742098
Iteration: 6800 Train Loss: 1.5814979932414603 Test Loss: 1.5533913802120936
Iteration: 7000 Train Loss: 1.5459623793726802 Test Loss: 1.5521505238488422
Iteration: 7200 Train Loss: 1.6583779403703662 Test Loss: 1.5555518877396375
Iteration: 7400 Train Loss: 1.3301024629522467 Test Loss: 1.5535253537104223
Epoch: 8 Mean Train Loss: 1.554886917650892 Train Accuracy: 42.56%
Iteration: 7600 Train Loss: 1.495824682808939 Test Loss: 1.5523515164308146
Iteration: 7800 Train Loss: 1.6681234016085233 Test Loss: 1.5559922144028557
Iteration: 8000 Train Loss: 1.6384642397090041 Test Loss: 1.5492352651667216
Iteration: 8200 Train Loss: 1.678136034801143 Test Loss: 1.5511354955411043
Iteration: 8400 Train Loss: 1.6126576816923845 Test Loss: 1.552567716006603
Epoch: 9 Mean Train Loss: 1.5528451146111797 Train Accuracy: 42.54%
Iteration: 8600 Train Loss: 1.4535061394943687 Test Loss: 1.5525535342447867
Iteration: 8800 Train Loss: 1.5263122574008423 Test Loss: 1.5542994025605603
Iteration: 9000 Train Loss: 1.4906687154568075 Test Loss: 1.551421175279842
Iteration: 9200 Train Loss: 1.568530500443696 Test Loss: 1.5483373642757616
Epoch: 10 Mean Train Loss: 1.5531401757431758 Train Accuracy: 42.52%
Iteration: 9400 Train Loss: 1.5781660273952012 Test Loss: 1.5417417179783384
Iteration: 9600 Train Loss: 1.3567132795195223 Test Loss: 1.553901226039975
Iteration: 9800 Train Loss: 1.644651360281732 Test Loss: 1.5534986170544385
Iteration: 10000 Train Loss: 1.47830819799871 Test Loss: 1.5539009108695774
Iteration: 10200 Train Loss: 1.5451232577742597 Test Loss: 1.5480989066593267
Epoch: 11 Mean Train Loss: 1.551353874189286 Train Accuracy: 42.43%
Iteration: 10400 Train Loss: 1.5428995626968907 Test Loss: 1.5525495914814489
Iteration: 10600 Train Loss: 1.672199860964869 Test Loss: 1.5470613611496036
Iteration: 10800 Train Loss: 1.4996622999238394 Test Loss: 1.5527215943455823
Iteration: 11000 Train Loss: 1.4602092929535286 Test Loss: 1.5491046091425975
Iteration: 11200 Train Loss: 1.528347525812084 Test Loss: 1.5549025450794367
Epoch: 12 Mean Train Loss: 1.551525561088404 Train Accuracy: 42.41%
Iteration: 11400 Train Loss: 1.6347577697260993 Test Loss: 1.5477860492411968
Iteration: 11600 Train Loss: 1.3984765479081829 Test Loss: 1.5524056928820202
Iteration: 11800 Train Loss: 1.570000921858052 Test Loss: 1.551076638800224
Iteration: 12000 Train Loss: 1.6469912000838218 Test Loss: 1.555531825935023
Epoch: 13 Mean Train Loss: 1.551986177544757 Train Accuracy: 42.35%
Iteration: 12200 Train Loss: 1.4382342920283593 Test Loss: 1.5465575363023072
Iteration: 12400 Train Loss: 1.5258841313320604 Test Loss: 1.5547491442663675
Iteration: 12600 Train Loss: 1.631010716089805 Test Loss: 1.5476619134715048
Iteration: 12800 Train Loss: 1.55294732887561 Test Loss: 1.5480873906020851
Iteration: 13000 Train Loss: 1.3677784469192986 Test Loss: 1.5497756469184416
Epoch: 14 Mean Train Loss: 1.5502177267680834 Train Accuracy: 42.31%
Iteration: 13200 Train Loss: 1.646543549688583 Test Loss: 1.559513165919889
Iteration: 13400 Train Loss: 1.6162543033519052 Test Loss: 1.552315684359734
Iteration: 13600 Train Loss: 1.6066375950297402 Test Loss: 1.5513251586543453
Iteration: 13800 Train Loss: 1.5552585417566527 Test Loss: 1.5466751448443123
```

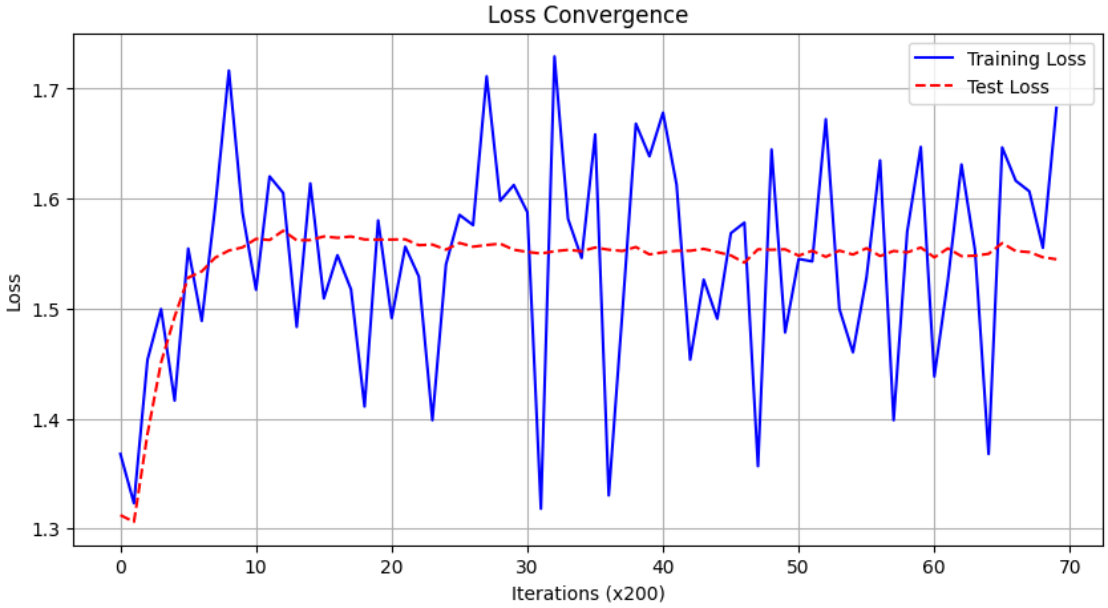
```
Iteration: 14000 Train Loss: 1.6824921750869004 Test Loss: 1.544821007687295
Epoch: 15 Mean Train Loss: 1.5522740859558608 Train Accuracy: 42.32%
Model trained!
```

[30]:
```
sgd_model_1by10alpha.evaluate(X_train, Y_train, X_test, Y_test)
```

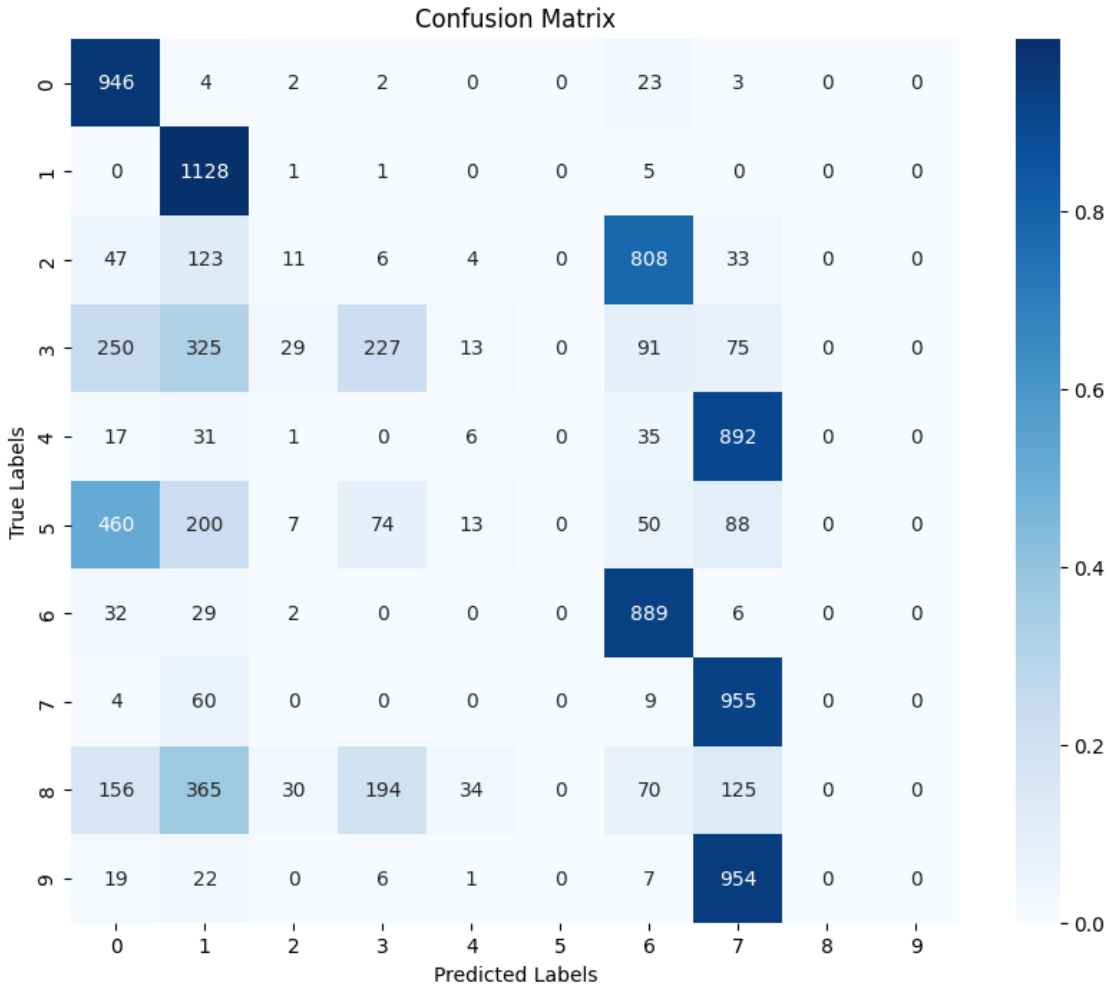
```
Train loss: 1.5462759224615779 Train accuracy: 42.20%
Test loss: 1.5464187583510585 Test accuracy: 41.62%

Confusion Matrix:
 [[ 946    4    2    2    0    0   23    3    0    0]
 [   0 1128    1    1    0    0    5    0    0    0]
 [  47  123   11    6    4    0  808   33    0    0]
 [ 250  325   29  227   13    0   91   75    0    0]
 [  17   31    1    0    6    0   35  892    0    0]
 [ 460  200    7   74   13    0   50   88    0    0]
 [  32   29    2    0    0    0  889    6    0    0]
 [   4   60    0    0    0    0    9  955    0    0]
 [ 156  365   30  194   34    0   70  125    0    0]
 [  19   22    0    6    1    0    7  954    0    0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.49      0.97      0.65       980
           1       0.49      0.99      0.66      1135
           2       0.13      0.01      0.02      1032
           3       0.45      0.22      0.30      1010
           4       0.08      0.01      0.01       982
           5       0.00      0.00      0.00       892
           6       0.45      0.93      0.60       958
           7       0.31      0.93      0.46      1028
           8       0.00      0.00      0.00       974
           9       0.00      0.00      0.00      1009

    accuracy                           0.42     10000
   macro avg       0.24      0.41      0.27     10000
weighted avg       0.25      0.42      0.28     10000


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
```

```
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[31]:
```
sgd_model_1by10alpha.plot_losses()
```



[32]:
```
sgd_model_1by10alpha.plot_confusion_matrix(X_test, Y_test)
```



11.2.2 Momentum

[33]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.1),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.1),
    ReLU(),
    FNNLayer(250, 100, weight_decay = 0.1),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.1)
]

momentum_model_1by10alpha = NN(layers)
```

```
optimizer = Momentum()
loss = LogLoss()
momentum_model_1by10alpha.compile(loss=loss, optimizer=optimizer)
momentum_model_1by10alpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 1.602932668450796 Test Loss: 1.5813581236344243
Iteration: 400 Train Loss: 1.4078767116390842 Test Loss: 1.5779135656979288
Iteration: 600 Train Loss: 1.5644803409400647 Test Loss: 1.5626653209876156
Iteration: 800 Train Loss: 1.531104490045791 Test Loss: 1.555662753435049
Epoch: 1 Mean Train Loss: 1.5704497176679004 Train Accuracy: 45.25%
Iteration: 1000 Train Loss: 1.6842977226910953 Test Loss: 1.5784349595020657
Iteration: 1200 Train Loss: 1.6318491270235613 Test Loss: 1.5625157333619808
Iteration: 1400 Train Loss: 1.4812259207716028 Test Loss: 1.5494324914243573
Iteration: 1600 Train Loss: 1.6383358544640496 Test Loss: 1.5691547954745908
Iteration: 1800 Train Loss: 1.6380349682399016 Test Loss: 1.5564486219229805
Epoch: 2 Mean Train Loss: 1.5617485898851968 Train Accuracy: 42.22%
Iteration: 2000 Train Loss: 1.5610238197040747 Test Loss: 1.5573987128918667
Iteration: 2200 Train Loss: 1.5721829708752284 Test Loss: 1.547304758958461
Iteration: 2400 Train Loss: 1.5203397563494974 Test Loss: 1.5757605739253764
Iteration: 2600 Train Loss: 1.423533404828254 Test Loss: 1.550300971945181
Iteration: 2800 Train Loss: 1.5039096770688902 Test Loss: 1.551399848988138
Epoch: 3 Mean Train Loss: 1.560500016024415 Train Accuracy: 42.05%
Iteration: 3000 Train Loss: 1.4808378062117566 Test Loss: 1.558689675874804
Iteration: 3200 Train Loss: 1.663122788152102 Test Loss: 1.56058703820683
Iteration: 3400 Train Loss: 1.5592356614652378 Test Loss: 1.5525396703342076
Iteration: 3600 Train Loss: 1.4223528216687003 Test Loss: 1.5351720569423557
Epoch: 4 Mean Train Loss: 1.558302850778738 Train Accuracy: 42.24%
Iteration: 3800 Train Loss: 1.4653563623272563 Test Loss: 1.5485690334458946
Iteration: 4000 Train Loss: 1.5675517295192374 Test Loss: 1.5782369665643412
Iteration: 4200 Train Loss: 1.5555127326140514 Test Loss: 1.5667703486260502
Iteration: 4400 Train Loss: 1.5097860924549784 Test Loss: 1.5764977504404376
Iteration: 4600 Train Loss: 1.3215092075036758 Test Loss: 1.5426061328961231
Epoch: 5 Mean Train Loss: 1.5600099315113574 Train Accuracy: 42.13%
Iteration: 4800 Train Loss: 1.5370792895590357 Test Loss: 1.5574319542125614
Iteration: 5000 Train Loss: 1.4910156714425244 Test Loss: 1.5637254104456149
Iteration: 5200 Train Loss: 1.3911149617469474 Test Loss: 1.5644925723165335
Iteration: 5400 Train Loss: 1.5365525056710896 Test Loss: 1.5557488997419042
Iteration: 5600 Train Loss: 1.3748366937990297 Test Loss: 1.565752082868413
Epoch: 6 Mean Train Loss: 1.5610821321651651 Train Accuracy: 42.21%
Iteration: 5800 Train Loss: 1.6197896606476898 Test Loss: 1.5688327103172732
Iteration: 6000 Train Loss: 1.5478855022517877 Test Loss: 1.5767167528940742
Iteration: 6200 Train Loss: 1.6672496976416409 Test Loss: 1.5537139364312305
Iteration: 6400 Train Loss: 1.5455067652668093 Test Loss: 1.558559380029114
Epoch: 7 Mean Train Loss: 1.5607668476387497 Train Accuracy: 42.24%
Iteration: 6600 Train Loss: 1.4476118654663062 Test Loss: 1.579313332385348
Iteration: 6800 Train Loss: 1.606852371014202 Test Loss: 1.5687114232190589
Iteration: 7000 Train Loss: 1.5843257682593284 Test Loss: 1.5585235598981773
```

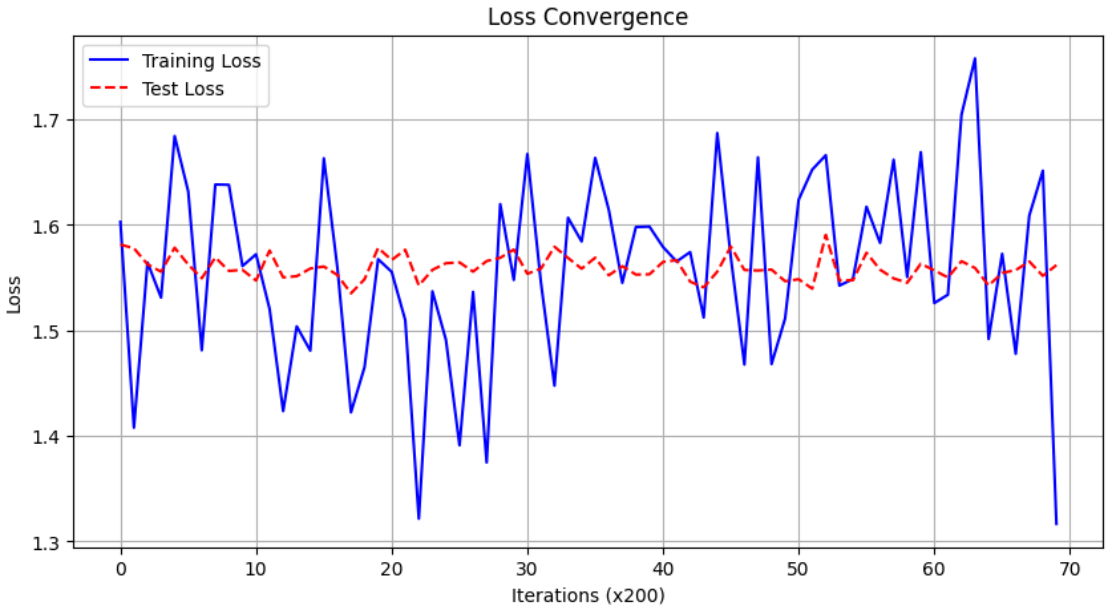
```
Iteration: 7200 Train Loss: 1.6635751974308497 Test Loss: 1.568965843383743
Iteration: 7400 Train Loss: 1.6144271179789622 Test Loss: 1.5522757052552603
Epoch: 8 Mean Train Loss: 1.5605995467076246 Train Accuracy: 42.18%
Iteration: 7600 Train Loss: 1.545093416453863 Test Loss: 1.5609295492571866
Iteration: 7800 Train Loss: 1.5980364069492181 Test Loss: 1.5529567959530748
Iteration: 8000 Train Loss: 1.5984532677896004 Test Loss: 1.5532385409302598
Iteration: 8200 Train Loss: 1.5791403610488561 Test Loss: 1.5649245814550616
Iteration: 8400 Train Loss: 1.565329311165753 Test Loss: 1.5666054089345194
Epoch: 9 Mean Train Loss: 1.5606177796820306 Train Accuracy: 42.25%
Iteration: 8600 Train Loss: 1.5742667025795027 Test Loss: 1.5462720767996863
Iteration: 8800 Train Loss: 1.5122959602492605 Test Loss: 1.5408077615292137
Iteration: 9000 Train Loss: 1.6871304468165063 Test Loss: 1.555757641261953
Iteration: 9200 Train Loss: 1.5691879707724505 Test Loss: 1.5795455120845816
Epoch: 10 Mean Train Loss: 1.5616691690176698 Train Accuracy: 42.15%
Iteration: 9400 Train Loss: 1.4677009112643962 Test Loss: 1.5572982062114935
Iteration: 9600 Train Loss: 1.6641015556139913 Test Loss: 1.5568405980364348
Iteration: 9800 Train Loss: 1.4681211003473393 Test Loss: 1.557684356356251
Iteration: 10000 Train Loss: 1.5114242319703635 Test Loss: 1.5466243669425772
Iteration: 10200 Train Loss: 1.6239816396856503 Test Loss: 1.5485377932974822
Epoch: 11 Mean Train Loss: 1.5594103872401877 Train Accuracy: 42.19%
Iteration: 10400 Train Loss: 1.6524512960555073 Test Loss: 1.5395836644164822
Iteration: 10600 Train Loss: 1.6661371032853354 Test Loss: 1.590592228374918
Iteration: 10800 Train Loss: 1.5424732670089494 Test Loss: 1.5473801939861196
Iteration: 11000 Train Loss: 1.5487820933214964 Test Loss: 1.5476012898975067
Iteration: 11200 Train Loss: 1.617255822657731 Test Loss: 1.5736634668301348
Epoch: 12 Mean Train Loss: 1.560624607313273 Train Accuracy: 42.08%
Iteration: 11400 Train Loss: 1.582983035748969 Test Loss: 1.55760392424715
Iteration: 11600 Train Loss: 1.6618453039115175 Test Loss: 1.5493295754433503
Iteration: 11800 Train Loss: 1.5506045568059548 Test Loss: 1.5452023156589434
Iteration: 12000 Train Loss: 1.6690387445919508 Test Loss: 1.5632353528142047
Epoch: 13 Mean Train Loss: 1.5603084227881443 Train Accuracy: 42.19%
Iteration: 12200 Train Loss: 1.525972612304837 Test Loss: 1.5569470436192834
Iteration: 12400 Train Loss: 1.533907379331431 Test Loss: 1.5502047060663613
Iteration: 12600 Train Loss: 1.7044274991482808 Test Loss: 1.5656441601835387
Iteration: 12800 Train Loss: 1.7578220365616548 Test Loss: 1.5592004192346882
Iteration: 13000 Train Loss: 1.4918871985812387 Test Loss: 1.5425777320399712
Epoch: 14 Mean Train Loss: 1.5593651239676487 Train Accuracy: 42.04%
Iteration: 13200 Train Loss: 1.572656633107334 Test Loss: 1.5545009606728788
Iteration: 13400 Train Loss: 1.4778128230703387 Test Loss: 1.5571888976598394
Iteration: 13600 Train Loss: 1.6089411019382727 Test Loss: 1.565347661687122
Iteration: 13800 Train Loss: 1.651401730712894 Test Loss: 1.5517662821878133
Iteration: 14000 Train Loss: 1.3166479007517562 Test Loss: 1.5619427010511775
Epoch: 15 Mean Train Loss: 1.5600087146416945 Train Accuracy: 42.12%
Model trained!
```

[34]:
```
momentum_model_1by10alpha.evaluate(X_train, Y_train, X_test, Y_test)
```

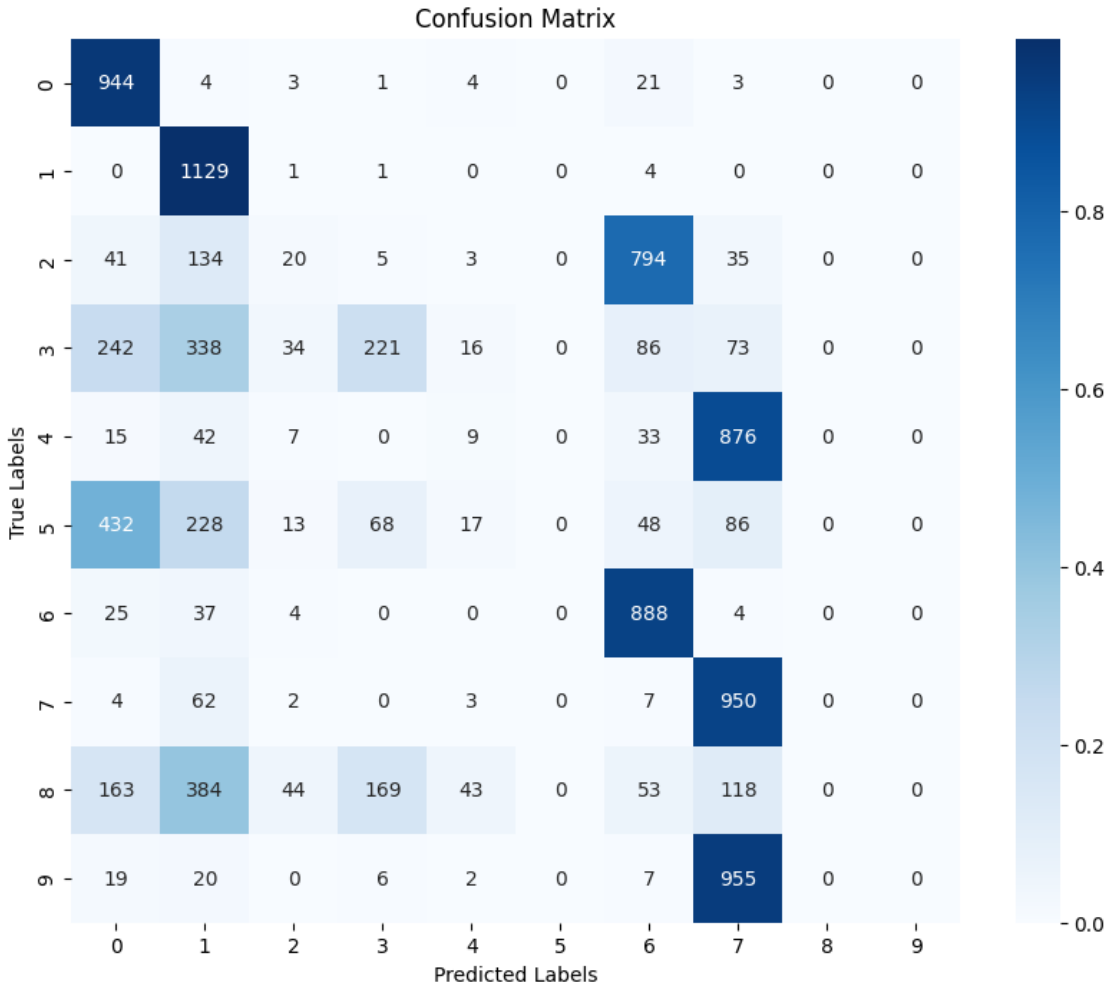
```
Train loss: 1.55828568328238 Train accuracy: 42.02%
```

```
Test loss: 1.5577860633783032 Test accuracy: 41.61%

Confusion Matrix:
 [[ 944    4    3    1    4    0   21    3    0    0]
 [   0 1129    1    1    0    0    4    0    0    0]
 [  41  134   20    5    3    0  794   35    0    0]
 [ 242  338   34  221   16    0   86   73    0    0]
 [  15   42    7    0    9    0   33  876    0    0]
 [ 432  228   13   68   17    0   48   86    0    0]
 [  25   37    4    0    0    0  888    4    0    0]
 [   4   62    2    0    3    0    7  950    0    0]
 [ 163  384   44  169   43    0   53  118    0    0]
 [  19   20    0    6    2    0    7  955    0    0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.96      0.66       980
           1       0.47      0.99      0.64      1135
           2       0.16      0.02      0.03      1032
           3       0.47      0.22      0.30      1010
           4       0.09      0.01      0.02       982
           5       0.00      0.00      0.00       892
           6       0.46      0.93      0.61       958
           7       0.31      0.92      0.46      1028
           8       0.00      0.00      0.00       974
           9       0.00      0.00      0.00      1009

    accuracy                           0.42     10000
   macro avg       0.25      0.41      0.27     10000
weighted avg       0.25      0.42      0.28     10000

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[35]:
```
momentum_model_1by10alpha.plot_losses()
```



[36]:
```
momentum_model_1by10alpha.plot_confusion_matrix(X_test, Y_test)
```



11.2.3 Conclusion

While models are able to predict 0, 1, 6, and 7 well, they fail to classify the other digits correctly. A lot of the samples are misclassified as 1 or 7, showing that the model is biased, the regularization has created too much bias, and left little room for variance, leading to an oversimplified model.

11.3 Weight Decay 0.01

11.3.1 SGD

[42]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.01),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.01),
```

```
    ReLU(),
    FNNLayer(250, 100, weight_decay = 0.01),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.01)
]

sgd_model_1by100alpha = NN(layers)
optimizer = SGD()
loss = LogLoss()
sgd_model_1by100alpha.compile(loss=loss, optimizer=optimizer)
sgd_model_1by100alpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.6478011788003758 Test Loss: 0.6027278642865362
Iteration: 400 Train Loss: 0.49035577070949193 Test Loss: 0.4141076848757327
Iteration: 600 Train Loss: 0.3131555977148571 Test Loss: 0.34514483951678854
Iteration: 800 Train Loss: 0.3401502261537856 Test Loss: 0.3130716323412736
Epoch: 1 Mean Train Loss: 0.5612367379592021 Train Accuracy: 85.32%
Iteration: 1000 Train Loss: 0.22011438787762588 Test Loss: 0.2921545442922081
Iteration: 1200 Train Loss: 0.1984796342604569 Test Loss: 0.27632887711451815
Iteration: 1400 Train Loss: 0.23487276367422702 Test Loss: 0.26579755649723946
Iteration: 1600 Train Loss: 0.34552557173767284 Test Loss: 0.2564873091975458
Iteration: 1800 Train Loss: 0.25915958187430516 Test Loss: 0.25029336146069675
Epoch: 2 Mean Train Loss: 0.26783633671550844 Train Accuracy: 92.97%
Iteration: 2000 Train Loss: 0.1588679850519502 Test Loss: 0.24608308594022127
Iteration: 2200 Train Loss: 0.23016463506576146 Test Loss: 0.24003048362658755
Iteration: 2400 Train Loss: 0.22366758406626314 Test Loss: 0.23782012109787945
Iteration: 2600 Train Loss: 0.21962567486141443 Test Loss: 0.2317542848208299
Iteration: 2800 Train Loss: 0.17362589589539437 Test Loss: 0.22934483814537748
Epoch: 3 Mean Train Loss: 0.2357677140169059 Train Accuracy: 94.02%
Iteration: 3000 Train Loss: 0.206254574458178 Test Loss: 0.229127593963442
Iteration: 3200 Train Loss: 0.2562604179304572 Test Loss: 0.22653940013639334
Iteration: 3400 Train Loss: 0.10502774276196106 Test Loss: 0.22250290109602464
Iteration: 3600 Train Loss: 0.15986194416080357 Test Loss: 0.2200758187709883
Epoch: 4 Mean Train Loss: 0.2220243664271391 Train Accuracy: 94.49%
Iteration: 3800 Train Loss: 0.21956263668226625 Test Loss: 0.21885116574251498
Iteration: 4000 Train Loss: 0.20856083766528535 Test Loss: 0.2171568238377411
Iteration: 4200 Train Loss: 0.25762343872938065 Test Loss: 0.21644297429770443
Iteration: 4400 Train Loss: 0.24869766981717376 Test Loss: 0.214542047157632
Iteration: 4600 Train Loss: 0.2943040980362553 Test Loss: 0.21310085991510286
Epoch: 5 Mean Train Loss: 0.213701098972074 Train Accuracy: 94.83%
Iteration: 4800 Train Loss: 0.23796563163233855 Test Loss: 0.21168905307903998
Iteration: 5000 Train Loss: 0.2118302468677696 Test Loss: 0.2114338369710235
Iteration: 5200 Train Loss: 0.1778091946844191 Test Loss: 0.20943097789820525
Iteration: 5400 Train Loss: 0.1681892399988148 Test Loss: 0.20851803273079886
Iteration: 5600 Train Loss: 0.1645430176353013 Test Loss: 0.20801736602154902
Epoch: 6 Mean Train Loss: 0.2084993892664763 Train Accuracy: 95.02%
Iteration: 5800 Train Loss: 0.2470883813048188 Test Loss: 0.20760013237529862
```

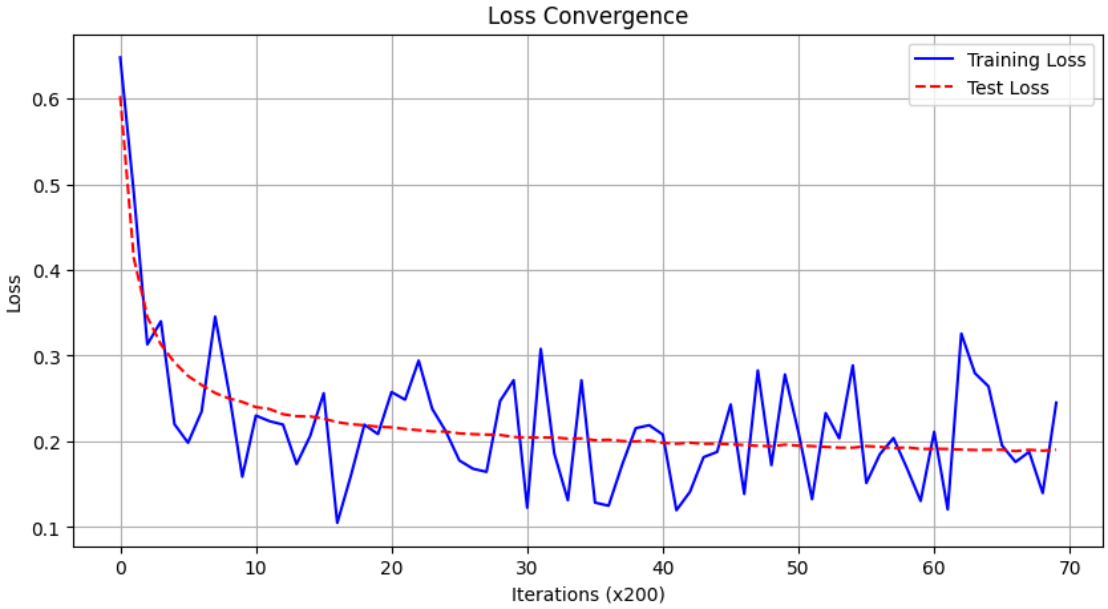
Iteration: 6000 Train Loss: 0.271391233216641 Test Loss: 0.2051274600746179
Iteration: 6200 Train Loss: 0.12271314095378205 Test Loss: 0.20469748310887098
Iteration: 6400 Train Loss: 0.3079063845281177 Test Loss: 0.20456826954068855
Epoch: 7 Mean Train Loss: 0.2036489675616711 Train Accuracy: 95.16%
Iteration: 6600 Train Loss: 0.1858712801202801 Test Loss: 0.20463316826710207
Iteration: 6800 Train Loss: 0.13145208854209262 Test Loss: 0.20329208435263874
Iteration: 7000 Train Loss: 0.27128430505871803 Test Loss: 0.20352118516737863
Iteration: 7200 Train Loss: 0.12868531032754688 Test Loss: 0.2013725995983437
Iteration: 7400 Train Loss: 0.12504261042944242 Test Loss: 0.20192643892418224
Epoch: 8 Mean Train Loss: 0.20055370969540792 Train Accuracy: 95.27%
Iteration: 7600 Train Loss: 0.17276226627293553 Test Loss: 0.2005381763158456
Iteration: 7800 Train Loss: 0.21544020953336168 Test Loss: 0.20003973731165953
Iteration: 8000 Train Loss: 0.21893649413431251 Test Loss: 0.20116699114322567
Iteration: 8200 Train Loss: 0.20793585670022247 Test Loss: 0.19837081424542718
Iteration: 8400 Train Loss: 0.11979208546721619 Test Loss: 0.1972475560488724
Epoch: 9 Mean Train Loss: 0.19722456872261962 Train Accuracy: 95.32%
Iteration: 8600 Train Loss: 0.1412622386930455 Test Loss: 0.19855781683882956
Iteration: 8800 Train Loss: 0.18163752792281215 Test Loss: 0.19723035048489498
Iteration: 9000 Train Loss: 0.1877840417808032 Test Loss: 0.1973841036120303
Iteration: 9200 Train Loss: 0.2431309909865208 Test Loss: 0.1966232491750488
Epoch: 10 Mean Train Loss: 0.19448187505432357 Train Accuracy: 95.46%
Iteration: 9400 Train Loss: 0.1387916829972477 Test Loss: 0.1957907963776475
Iteration: 9600 Train Loss: 0.28276141336317906 Test Loss: 0.19488600915863433
Iteration: 9800 Train Loss: 0.1723660241266252 Test Loss: 0.1944187259023612
Iteration: 10000 Train Loss: 0.2780847414556054 Test Loss: 0.19623304482464815
Iteration: 10200 Train Loss: 0.21036318427906897 Test Loss: 0.19505481541583483
Epoch: 11 Mean Train Loss: 0.19217323085555477 Train Accuracy: 95.50%
Iteration: 10400 Train Loss: 0.1326250751390058 Test Loss: 0.1945726919809512
Iteration: 10600 Train Loss: 0.233111953837994 Test Loss: 0.19362612888725964
Iteration: 10800 Train Loss: 0.20357193727658685 Test Loss: 0.19270418165359876
Iteration: 11000 Train Loss: 0.28868218508821075 Test Loss: 0.19269854242549475
Iteration: 11200 Train Loss: 0.1515556482003366 Test Loss: 0.19455874654315428
Epoch: 12 Mean Train Loss: 0.1902322731282699 Train Accuracy: 95.60%
Iteration: 11400 Train Loss: 0.18511856705756302 Test Loss: 0.1936659332796097
Iteration: 11600 Train Loss: 0.20399871357874366 Test Loss: 0.19202169129174682
Iteration: 11800 Train Loss: 0.16852545761125853 Test Loss: 0.19309814148888518
Iteration: 12000 Train Loss: 0.13062961275406773 Test Loss: 0.19121395649674608
Epoch: 13 Mean Train Loss: 0.18833743881919301 Train Accuracy: 95.61%
Iteration: 12200 Train Loss: 0.2111474746032284 Test Loss: 0.19141809230837198
Iteration: 12400 Train Loss: 0.12068265842911428 Test Loss: 0.19120129207407266
Iteration: 12600 Train Loss: 0.3257779171098382 Test Loss: 0.1904588409700953
Iteration: 12800 Train Loss: 0.2796558236141335 Test Loss: 0.1900018761717644
Iteration: 13000 Train Loss: 0.26428643233821125 Test Loss: 0.19017941221997892
Epoch: 14 Mean Train Loss: 0.18662371130863262 Train Accuracy: 95.71%
Iteration: 13200 Train Loss: 0.19530107160594573 Test Loss: 0.19028416484739116
Iteration: 13400 Train Loss: 0.17602749139145435 Test Loss: 0.1887100391009828
Iteration: 13600 Train Loss: 0.18789079770207356 Test Loss: 0.1902500609088396
Iteration: 13800 Train Loss: 0.13980694292048657 Test Loss: 0.18926418064364361

```
Iteration: 14000 Train Loss: 0.2451998809293833 Test Loss: 0.19030727352269625
Epoch: 15 Mean Train Loss: 0.18549118857868008 Train Accuracy: 95.68%
Model trained!
```

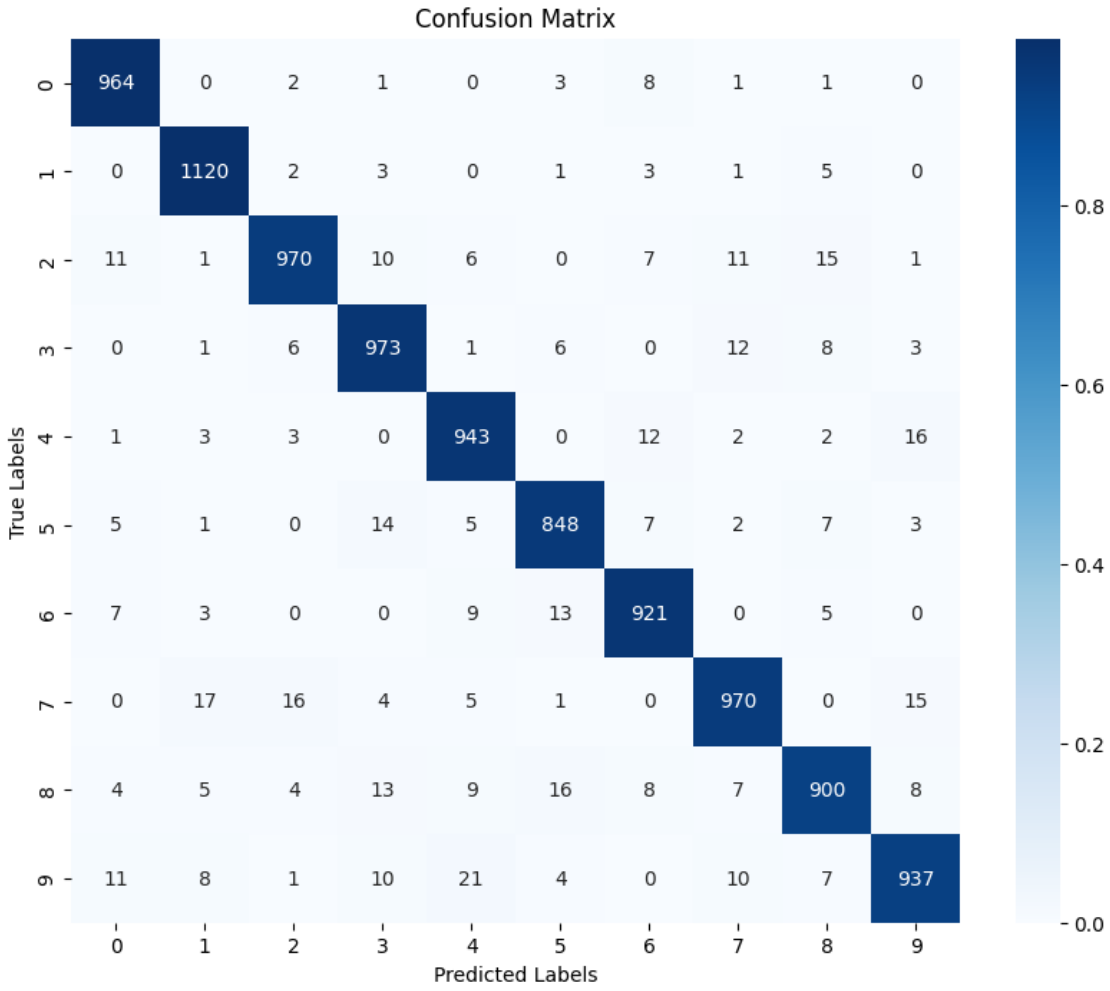
[43]:
```
sgd_model_1by100alpha.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.18284910755199604 Train accuracy: 95.83%
Test loss: 0.19130740941414318 Test accuracy: 95.46%

Confusion Matrix:
 [[ 964    0    2    1    0    3    8    1    1    0]
 [   0 1120    2    3    0    1    3    1    5    0]
 [  11    1  970   10    6    0    7   11   15    1]
 [   0    1    6  973    1    6    0   12    8    3]
 [   1    3    3    0  943    0   12    2    2   16]
 [   5    1    0   14    5  848    7    2    7    3]
 [   7    3    0    0    9   13  921    0    5    0]
 [   0   17   16    4    5    1    0  970    0   15]
 [   4    5    4   13    9   16    8    7  900    8]
 [  11    8    1   10   21    4    0   10    7  937]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       980
           1       0.97      0.99      0.98      1135
           2       0.97      0.94      0.95      1032
           3       0.95      0.96      0.95      1010
           4       0.94      0.96      0.95       982
           5       0.95      0.95      0.95       892
           6       0.95      0.96      0.96       958
           7       0.95      0.94      0.95      1028
           8       0.95      0.92      0.94       974
           9       0.95      0.93      0.94      1009

    accuracy                           0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000
```

[44]:
```
sgd_model_1by100alpha.plot_losses()
```



[45]:
```
sgd_model_1by100alpha.plot_confusion_matrix(X_test, Y_test)
```



11.3.2 Momentum

[46]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.01),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.01),
    ReLU(),
    FNNLayer(250, 100, weight_decay = 0.01),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.01)
]

momentum_model_1by100alpha = NN(layers)
```

```
optimizer = Momentum()
loss = LogLoss()
momentum_model_1by100alpha.compile(loss=loss, optimizer=optimizer)
momentum_model_1by100alpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.35832900072222074 Test Loss: 0.3076311752388113
Iteration: 400 Train Loss: 0.1818202912544873 Test Loss: 0.26489773891357865
Iteration: 600 Train Loss: 0.2582076104590476 Test Loss: 0.24015472183859732
Iteration: 800 Train Loss: 0.22304528152663256 Test Loss: 0.23836360249474361
Epoch: 1 Mean Train Loss: 0.3262232575408285 Train Accuracy: 91.25%
Iteration: 1000 Train Loss: 0.16036344210723985 Test Loss: 0.21637387479982273
Iteration: 1200 Train Loss: 0.27083984994038507 Test Loss: 0.2132798623922533
Iteration: 1400 Train Loss: 0.12122505837701608 Test Loss: 0.21431841648083247
Iteration: 1600 Train Loss: 0.26026568433131714 Test Loss: 0.21211364862405474
Iteration: 1800 Train Loss: 0.2548242944991669 Test Loss: 0.22047773093641143
Epoch: 2 Mean Train Loss: 0.2221437552869206 Train Accuracy: 94.38%
Iteration: 2000 Train Loss: 0.25465836245371887 Test Loss: 0.20731163431395122
Iteration: 2200 Train Loss: 0.6851621589257922 Test Loss: 0.21367113986559988
Iteration: 2400 Train Loss: 0.2875837650303972 Test Loss: 0.20659466894246448
Iteration: 2600 Train Loss: 0.1314394803700453 Test Loss: 0.21631092256376003
Iteration: 2800 Train Loss: 0.19906300497683246 Test Loss: 0.21647307971555801
Epoch: 3 Mean Train Loss: 0.2139841907097664 Train Accuracy: 94.58%
Iteration: 3000 Train Loss: 0.18646578665503927 Test Loss: 0.21214049579247093
Iteration: 3200 Train Loss: 0.0804360397104288 Test Loss: 0.20718919955229528
Iteration: 3400 Train Loss: 0.204398327423711 Test Loss: 0.20567321064768238
Iteration: 3600 Train Loss: 0.21236508948479785 Test Loss: 0.2169014124115793
Epoch: 4 Mean Train Loss: 0.20937804377236585 Train Accuracy: 94.71%
Iteration: 3800 Train Loss: 0.18932052207152417 Test Loss: 0.2077170714522437
Iteration: 4000 Train Loss: 0.370408354118295 Test Loss: 0.20098482830685582
Iteration: 4200 Train Loss: 0.15593634530030923 Test Loss: 0.2095490638210708
Iteration: 4400 Train Loss: 0.119652347574978 Test Loss: 0.20541102066782005
Iteration: 4600 Train Loss: 0.18559755269016548 Test Loss: 0.20064081917155663
Epoch: 5 Mean Train Loss: 0.20654307866362212 Train Accuracy: 94.81%
Iteration: 4800 Train Loss: 0.19407171933647888 Test Loss: 0.19919725768889043
Iteration: 5000 Train Loss: 0.20031978213902096 Test Loss: 0.20019168297240839
Iteration: 5200 Train Loss: 0.12179799399988475 Test Loss: 0.20529288967368925
Iteration: 5400 Train Loss: 0.285078355428226 Test Loss: 0.21494613702803111
Iteration: 5600 Train Loss: 0.1792929729463268 Test Loss: 0.20119274685254804
Epoch: 6 Mean Train Loss: 0.20899740465688812 Train Accuracy: 94.75%
Iteration: 5800 Train Loss: 0.2148471628235626 Test Loss: 0.20614649676663258
Iteration: 6000 Train Loss: 0.13105853820622948 Test Loss: 0.2031409852192247
Iteration: 6200 Train Loss: 0.07606334204234061 Test Loss: 0.21112135448201869
Iteration: 6400 Train Loss: 0.13320706978842797 Test Loss: 0.19963827868638187
Epoch: 7 Mean Train Loss: 0.20664437473776012 Train Accuracy: 94.81%
Iteration: 6600 Train Loss: 0.2203763954811295 Test Loss: 0.2092937095544759
Iteration: 6800 Train Loss: 0.2250696486878426 Test Loss: 0.2023729555910635
Iteration: 7000 Train Loss: 0.22089832290205386 Test Loss: 0.20588287522674795
```

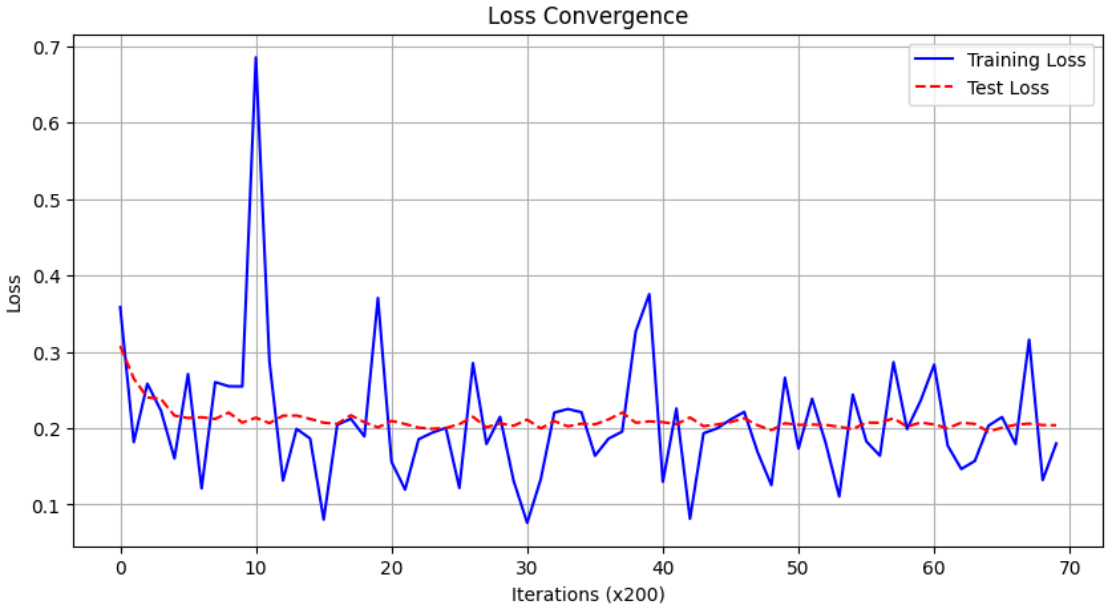
```
Iteration: 7200 Train Loss: 0.1639298211916402 Test Loss: 0.204936531803054
Iteration: 7400 Train Loss: 0.18633610659792013 Test Loss: 0.211507793896316
Epoch: 8 Mean Train Loss: 0.20757046934379175 Train Accuracy: 94.78%
Iteration: 7600 Train Loss: 0.19548936348142548 Test Loss: 0.22037606811324292
Iteration: 7800 Train Loss: 0.3264632731347608 Test Loss: 0.20716692879348314
Iteration: 8000 Train Loss: 0.37543309672818637 Test Loss: 0.20890957147377082
Iteration: 8200 Train Loss: 0.1298103675837711 Test Loss: 0.20791468235160154
Iteration: 8400 Train Loss: 0.22575575998438513 Test Loss: 0.2051533179409347
Epoch: 9 Mean Train Loss: 0.2045905873911592 Train Accuracy: 94.82%
Iteration: 8600 Train Loss: 0.08147140318730496 Test Loss: 0.2139973043997065
Iteration: 8800 Train Loss: 0.1932853174633157 Test Loss: 0.202580414520583
Iteration: 9000 Train Loss: 0.1997035317943149 Test Loss: 0.20481708615806374
Iteration: 9200 Train Loss: 0.2113964583141325 Test Loss: 0.20798162023680428
Epoch: 10 Mean Train Loss: 0.20778750674774762 Train Accuracy: 94.75%
Iteration: 9400 Train Loss: 0.22125700437150372 Test Loss: 0.21300109770562425
Iteration: 9600 Train Loss: 0.1678525256682732 Test Loss: 0.20369254980854973
Iteration: 9800 Train Loss: 0.12541391562863202 Test Loss: 0.1973828085733312
Iteration: 10000 Train Loss: 0.26589322592343534 Test Loss: 0.20637858624142724
Iteration: 10200 Train Loss: 0.17344063161195217 Test Loss: 0.20429678835420173
Epoch: 11 Mean Train Loss: 0.2049622500291536 Train Accuracy: 94.89%
Iteration: 10400 Train Loss: 0.23843153755087523 Test Loss: 0.20471880802747416
Iteration: 10600 Train Loss: 0.1807118409989948 Test Loss: 0.2043092771847739
Iteration: 10800 Train Loss: 0.11064318198897771 Test Loss: 0.2017521453243058
Iteration: 11000 Train Loss: 0.2438966556811657 Test Loss: 0.1988694415652443
Iteration: 11200 Train Loss: 0.18268306731368733 Test Loss: 0.20735907177536936
Epoch: 12 Mean Train Loss: 0.20576274689987692 Train Accuracy: 94.83%
Iteration: 11400 Train Loss: 0.16393757172687257 Test Loss: 0.20707922047779104
Iteration: 11600 Train Loss: 0.28637454387811345 Test Loss: 0.21295466256628107
Iteration: 11800 Train Loss: 0.19882105506211484 Test Loss: 0.20205880919110253
Iteration: 12000 Train Loss: 0.23590003820453526 Test Loss: 0.2074123224847028
Epoch: 13 Mean Train Loss: 0.20625788114779006 Train Accuracy: 94.87%
Iteration: 12200 Train Loss: 0.2831102521537756 Test Loss: 0.20473879452237956
Iteration: 12400 Train Loss: 0.17703956234590695 Test Loss: 0.19992215560732973
Iteration: 12600 Train Loss: 0.14651395084619734 Test Loss: 0.207210958443725
Iteration: 12800 Train Loss: 0.15711883211768551 Test Loss: 0.2054709602065978
Iteration: 13000 Train Loss: 0.20317072318005863 Test Loss: 0.1953527385334781
Epoch: 14 Mean Train Loss: 0.20716508578390408 Train Accuracy: 94.71%
Iteration: 13200 Train Loss: 0.21454999962955956 Test Loss: 0.2006607249753352
Iteration: 13400 Train Loss: 0.17924311378680488 Test Loss: 0.2045132444518202
Iteration: 13600 Train Loss: 0.3158644800908519 Test Loss: 0.2059499562401379
Iteration: 13800 Train Loss: 0.1321077211939701 Test Loss: 0.20410811250859368
Iteration: 14000 Train Loss: 0.1799730696919135 Test Loss: 0.2038483217969464
Epoch: 15 Mean Train Loss: 0.20508393130811192 Train Accuracy: 94.88%
Model trained!
```

[47]:
```
momentum_model_1by100alpha.evaluate(X_train, Y_train, X_test, Y_test)
```

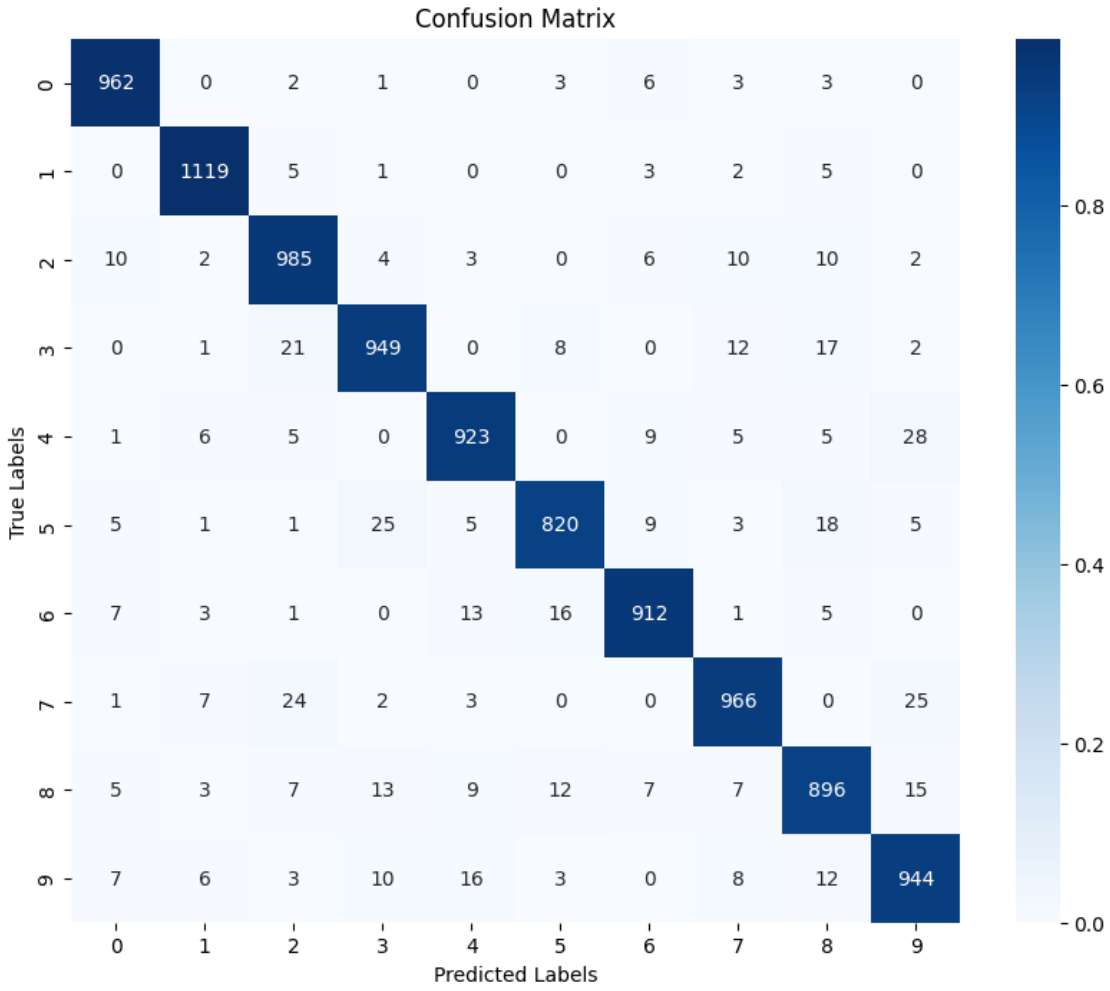
```
Train loss: 0.19606497892555544 Train accuracy: 95.26%
```

```
Test loss: 0.20456957419414493 Test accuracy: 94.76%

Confusion Matrix:
 [[ 962    0    2    1    0    3    6    3    3    0]
 [   0 1119    5    1    0    0    3    2    5    0]
 [  10    2  985    4    3    0    6   10   10    2]
 [   0    1   21  949    0    8    0   12   17    2]
 [   1    6    5    0  923    0    9    5    5   28]
 [   5    1    1   25    5  820    9    3   18    5]
 [   7    3    1    0   13   16  912    1    5    0]
 [   1    7   24    2    3    0    0  966    0   25]
 [   5    3    7   13    9   12    7    7  896   15]
 [   7    6    3   10   16    3    0    8   12  944]]

Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       980
           1       0.97      0.99      0.98      1135
           2       0.93      0.95      0.94      1032
           3       0.94      0.94      0.94      1010
           4       0.95      0.94      0.94       982
           5       0.95      0.92      0.94       892
           6       0.96      0.95      0.95       958
           7       0.95      0.94      0.94      1028
           8       0.92      0.92      0.92       974
           9       0.92      0.94      0.93      1009

    accuracy                           0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000
```

[48]:
```
momentum_model_1by100alpha.plot_losses()
```



[49]:
```
momentum_model_1by100alpha.plot_confusion_matrix(X_test, Y_test)
```



11.3.3 Conclusion

For both the models, the train and test accuracies are very close, meaning that the regularization has done a good job of rpeventing overfitting.

11.4 Weight Decay 0.001

11.4.1 SGD

[52]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.001),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.001),
    ReLU(),
```

```
    FNNLayer(250, 100, weight_decay = 0.001),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.001)
]

sgd_model_1by1000alpha = NN(layers)
optimizer = SGD()
loss = LogLoss()
sgd_model_1by1000alpha.compile(loss=loss, optimizer=optimizer)
sgd_model_1by1000alpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.6835975935249171 Test Loss: 0.5842382672525204
Iteration: 400 Train Loss: 0.25429575060854626 Test Loss: 0.37671653321163046
Iteration: 600 Train Loss: 0.3281061832268368 Test Loss: 0.3166508808486087
Iteration: 800 Train Loss: 0.28410361974805665 Test Loss: 0.2746570832434464
Epoch: 1 Mean Train Loss: 0.5455771327433805 Train Accuracy: 85.49%
Iteration: 1000 Train Loss: 0.10130433038612 38 Test Loss: 0.24732943719183298
Iteration: 1200 Train Loss: 0.24151709264219634 Test Loss: 0.23450896490160306
Iteration: 1400 Train Loss: 0.21067371879090568 Test Loss: 0.21765580355891975
Iteration: 1600 Train Loss: 0.12327784020087416 Test Loss: 0.21079470197321035
Iteration: 1800 Train Loss: 0.21285091864952943 Test Loss: 0.1992364534515762
Epoch: 2 Mean Train Loss: 0.2151899523448258 Train Accuracy: 93.89%
Iteration: 2000 Train Loss: 0.10443868589815222 Test Loss: 0.1888553290897144
Iteration: 2200 Train Loss: 0.2523722019441601 Test Loss: 0.18050698269584192
Iteration: 2400 Train Loss: 0.1322794636020016 Test Loss: 0.17667494661582875
Iteration: 2600 Train Loss: 0.07390437223878675 Test Loss: 0.17448849300780545
Iteration: 2800 Train Loss: 0.09191314653407229 Test Loss: 0.16949163728922856
Epoch: 3 Mean Train Loss: 0.16094182471008103 Train Accuracy: 95.33%
Iteration: 3000 Train Loss: 0.17543547720470984 Test Loss: 0.16596543444934259
Iteration: 3200 Train Loss: 0.1779741935980547 Test Loss: 0.16115587975472173
Iteration: 3400 Train Loss: 0.1670209092325498 Test Loss: 0.15913789836824488
Iteration: 3600 Train Loss: 0.17371912143195586 Test Loss: 0.151908198946838
Epoch: 4 Mean Train Loss: 0.13190968162413527 Train Accuracy: 96.18%
Iteration: 3800 Train Loss: 0.06742302449588664 Test Loss: 0.14918016020826072
Iteration: 4000 Train Loss: 0.0625458583119628 Test Loss: 0.14939777162243414
Iteration: 4200 Train Loss: 0.08996413633097136 Test Loss: 0.14370393564813613
Iteration: 4400 Train Loss: 0.08645588029674608 Test Loss: 0.14162677106922028
Iteration: 4600 Train Loss: 0.20534030576416176 Test Loss: 0.14018921908539816
Epoch: 5 Mean Train Loss: 0.11176044863076971 Train Accuracy: 96.81%
Iteration: 4800 Train Loss: 0.1325519648986952 Test Loss: 0.1362341820792594
Iteration: 5000 Train Loss: 0.05821527738987592 Test Loss: 0.13653711331899301
Iteration: 5200 Train Loss: 0.052512534952592206 Test Loss: 0.13503707144192997
Iteration: 5400 Train Loss: 0.03585103030435828 Test Loss: 0.13357897622083453
Iteration: 5600 Train Loss: 0.0716909451005433 Test Loss: 0.1296756733261713
Epoch: 6 Mean Train Loss: 0.098191436611564 Train Accuracy: 97.31%
Iteration: 5800 Train Loss: 0.030522592723933317 Test Loss: 0.12749660781916228
Iteration: 6000 Train Loss: 0.09249334603638844 Test Loss: 0.1273718182912033
```

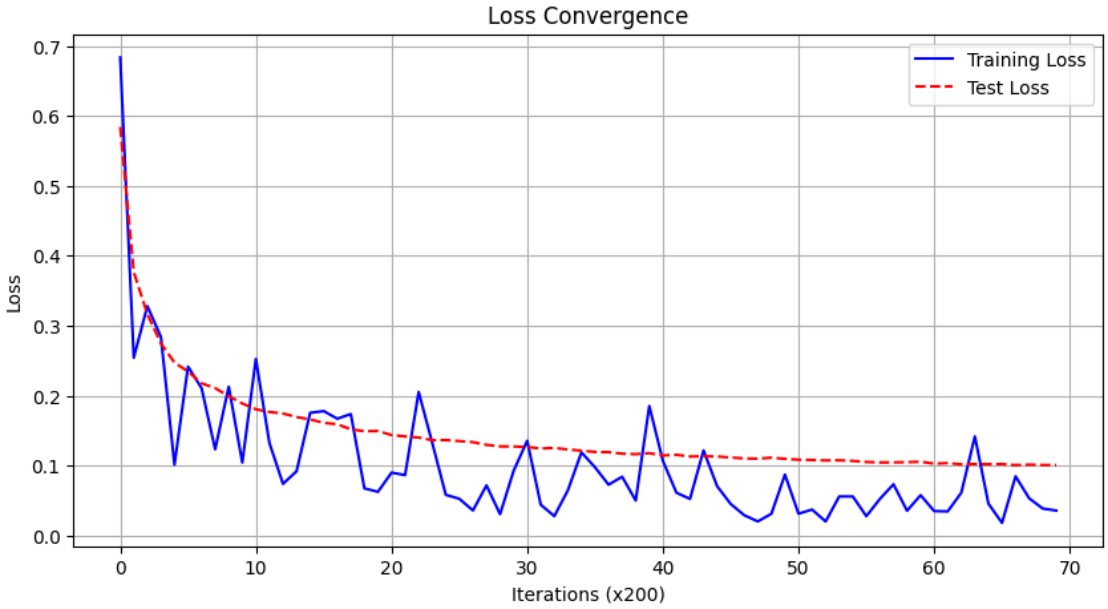
```
Iteration: 6200 Train Loss: 0.13555141274810756 Test Loss: 0.1265680842458521
Iteration: 6400 Train Loss: 0.044037590336645165 Test Loss: 0.12444102441023845
Epoch: 7 Mean Train Loss: 0.08707224224715994 Train Accuracy: 97.63%
Iteration: 6600 Train Loss: 0.027738886242470326 Test Loss: 0.12524849297251378
Iteration: 6800 Train Loss: 0.06476639250728906 Test Loss: 0.12304254810984559
Iteration: 7000 Train Loss: 0.11895782468671429 Test Loss: 0.1213693767351821
Iteration: 7200 Train Loss: 0.09753619964343037 Test Loss: 0.11951135723595105
Iteration: 7400 Train Loss: 0.07269960576051328 Test Loss: 0.11922869766503713
Epoch: 8 Mean Train Loss: 0.07828313184507099 Train Accuracy: 97.91%
Iteration: 7600 Train Loss: 0.08406237212718304 Test Loss: 0.11704799774863427
Iteration: 7800 Train Loss: 0.05011087863833255 Test Loss: 0.11640302591468595
Iteration: 8000 Train Loss: 0.18524886504864188 Test Loss: 0.11774885052063827
Iteration: 8200 Train Loss: 0.10690942390283636 Test Loss: 0.11447664515791596
Iteration: 8400 Train Loss: 0.06128104025681047 Test Loss: 0.11552357766603812
Epoch: 9 Mean Train Loss: 0.07137539345859856 Train Accuracy: 98.16%
Iteration: 8600 Train Loss: 0.05231532249716617 Test Loss: 0.11296562715847541
Iteration: 8800 Train Loss: 0.12165293037515869 Test Loss: 0.11328329538822764
Iteration: 9000 Train Loss: 0.07058363998784874 Test Loss: 0.11308118574757649
Iteration: 9200 Train Loss: 0.045170846358722705 Test Loss: 0.11156976332615356
Epoch: 10 Mean Train Loss: 0.06491010106516283 Train Accuracy: 98.34%
Iteration: 9400 Train Loss: 0.02905385541781904 Test Loss: 0.11022427950299335
Iteration: 9600 Train Loss: 0.020167018030617906 Test Loss: 0.10990635400087372
Iteration: 9800 Train Loss: 0.031089466543380826 Test Loss: 0.11130247558414656
Iteration: 10000 Train Loss: 0.08712244100365864 Test Loss: 0.10971950622870226
Iteration: 10200 Train Loss: 0.03125712432229724 Test Loss: 0.10842176762822206
Epoch: 11 Mean Train Loss: 0.05971262445846225 Train Accuracy: 98.51%
Iteration: 10400 Train Loss: 0.03713930866497677 Test Loss: 0.10795213628162051
Iteration: 10600 Train Loss: 0.02009550008221636 Test Loss: 0.10737274171188693
Iteration: 10800 Train Loss: 0.05590000985152939 Test Loss: 0.10760232205861524
Iteration: 11000 Train Loss: 0.05588060702956266 Test Loss: 0.10667609257830492
Iteration: 11200 Train Loss: 0.027654255240877425 Test Loss: 0.10520115463100942
Epoch: 12 Mean Train Loss: 0.05557041240960449 Train Accuracy: 98.64%
Iteration: 11400 Train Loss: 0.05234826494535239 Test Loss: 0.10450562170848533
Iteration: 11600 Train Loss: 0.07331086065729281 Test Loss: 0.10461966958047295
Iteration: 11800 Train Loss: 0.035453153005361934 Test Loss: 0.10489003607105064
Iteration: 12000 Train Loss: 0.0577129216267726 Test Loss: 0.10552118466902538
Epoch: 13 Mean Train Loss: 0.05157617407043289 Train Accuracy: 98.78%
Iteration: 12200 Train Loss: 0.034953731857497915 Test Loss: 0.10278795041131697
Iteration: 12400 Train Loss: 0.0345202857247852 Test Loss: 0.10357944298293642
Iteration: 12600 Train Loss: 0.061218694730796906 Test Loss: 0.10180935362473012
Iteration: 12800 Train Loss: 0.14170002051603714 Test Loss: 0.10224912216390396
Iteration: 13000 Train Loss: 0.0456060470851517 Test Loss: 0.10202126642834422
Epoch: 14 Mean Train Loss: 0.0487659262326329 Train Accuracy: 98.89%
Iteration: 13200 Train Loss: 0.01809932500399257 Test Loss: 0.10224213164392983
Iteration: 13400 Train Loss: 0.08464786320861753 Test Loss: 0.10069065428240656
Iteration: 13600 Train Loss: 0.053299308575346066 Test Loss: 0.1013747902938757
Iteration: 13800 Train Loss: 0.03861130324172298 Test Loss: 0.10086212245151
Iteration: 14000 Train Loss: 0.035689585706065015 Test Loss: 0.10078309889877432
```

```
Epoch: 15 Mean Train Loss: 0.046119032236865205 Train Accuracy: 98.97%
Model trained!
```

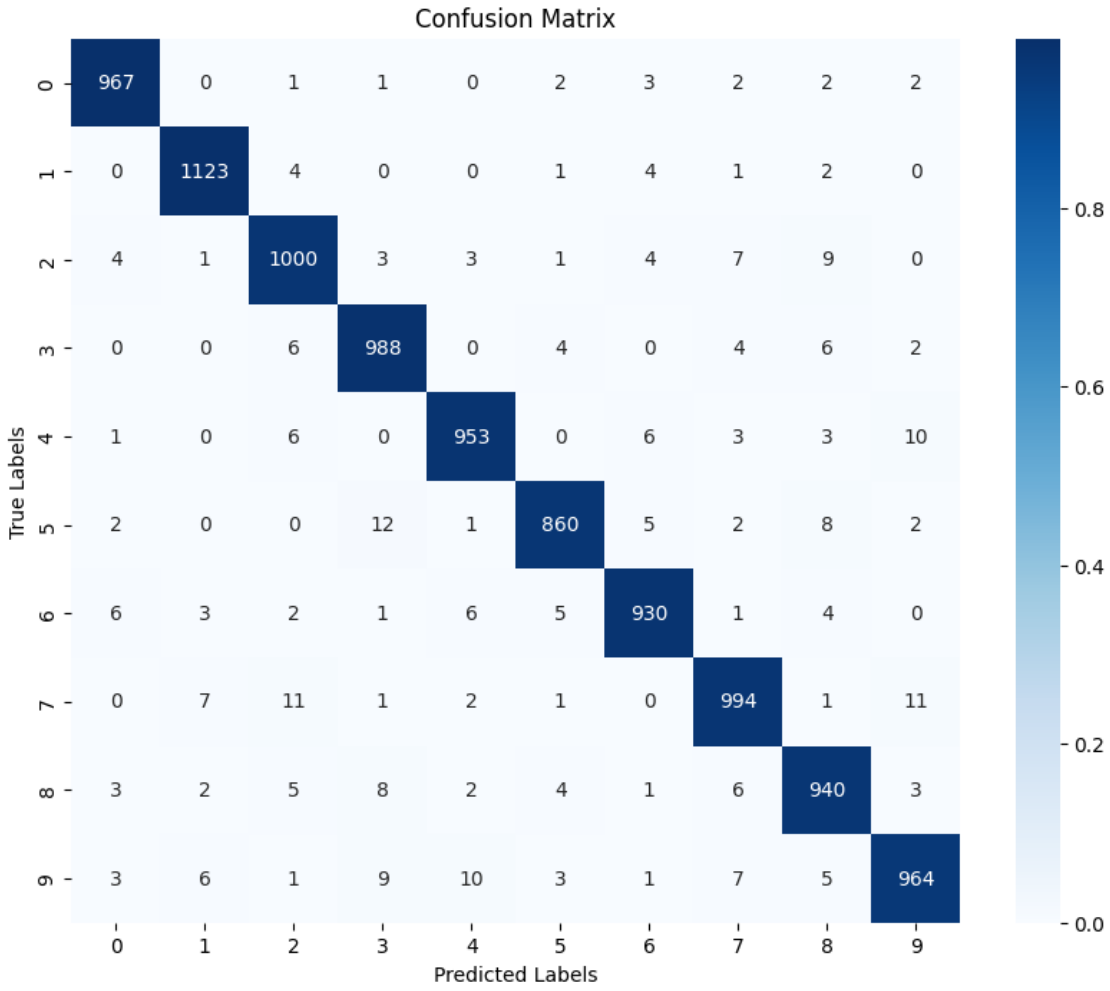
[53]:
```
sgd_model_1by1000alpha.evaluate(X_train, Y_train, X_test, Y_test)
```

```
Train loss: 0.04121106449203172 Train accuracy: 99.13%
Test loss: 0.1004162244304904 Test accuracy: 97.19%

Confusion Matrix:
 [[ 967    0    1    1    0    2    3    2    2    2]
 [   0 1123    4    0    0    1    4    1    2    0]
 [   4    1 1000    3    3    1    4    7    9    0]
 [   0    0    6  988    0    4    0    4    6    2]
 [   1    0    6    0  953    0    6    3    3   10]
 [   2    0    0   12    1  860    5    2    8    2]
 [   6    3    2    1    6    5  930    1    4    0]
 [   0    7   11    1    2    1    0  994    1   11]
 [   3    2    5    8    2    4    1    6  940    3]
 [   3    6    1    9   10    3    1    7    5  964]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       980
           1       0.98      0.99      0.99      1135
           2       0.97      0.97      0.97      1032
           3       0.97      0.98      0.97      1010
           4       0.98      0.97      0.97       982
           5       0.98      0.96      0.97       892
           6       0.97      0.97      0.97       958
           7       0.97      0.97      0.97      1028
           8       0.96      0.97      0.96       974
           9       0.97      0.96      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

[54]:
```
sgd_model_1by1000alpha.plot_losses()
```



[55]:
```
sgd_model_1by1000alpha.plot_confusion_matrix(X_test, Y_test)
```



11.4.2 Momentum

[56]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500, weight_decay = 0.001),
    ReLU(),
    FNNLayer(500, 250, weight_decay = 0.001),
    ReLU(),
    FNNLayer(250, 100, weight_decay = 0.001),
    ReLU(),
    FNNLayer(100, output_dim, weight_decay = 0.001)
]

momentum_model_1by1000alpha = NN(layers)
```

```
optimizer = Momentum()
loss = LogLoss()
momentum_model_1by1000alpha.compile(loss=loss, optimizer=optimizer)
momentum_model_1by1000alpha.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.14017232233120452 Test Loss: 0.2618560127260318
Iteration: 400 Train Loss: 0.2028724547730099 Test Loss: 0.2293107173992485
Iteration: 600 Train Loss: 0.21186401203850772 Test Loss: 0.20011081457850602
Iteration: 800 Train Loss: 0.3124380139871451 Test Loss: 0.18123690615191165
Epoch: 1 Mean Train Loss: 0.2803593746808626 Train Accuracy: 92.02%
Iteration: 1000 Train Loss: 0.033148227378844486 Test Loss: 0.14165150994853565
Iteration: 1200 Train Loss: 0.06784791094896057 Test Loss: 0.15012010711830504
Iteration: 1400 Train Loss: 0.21037503623002932 Test Loss: 0.1295933321537118
Iteration: 1600 Train Loss: 0.052952529474863216 Test Loss: 0.13754443466368582
Iteration: 1800 Train Loss: 0.0872768369087927 Test Loss: 0.11996823820110246
Epoch: 2 Mean Train Loss: 0.11297820664916801 Train Accuracy: 96.83%
Iteration: 2000 Train Loss: 0.02877702931912434 Test Loss: 0.12297897724953132
Iteration: 2200 Train Loss: 0.08024569490625859 Test Loss: 0.11462726713304269
Iteration: 2400 Train Loss: 0.19163044735687124 Test Loss: 0.10977957846692785
Iteration: 2600 Train Loss: 0.04768404097568199 Test Loss: 0.11035559684832269
Iteration: 2800 Train Loss: 0.13400870544097468 Test Loss: 0.10665174422519023
Epoch: 3 Mean Train Loss: 0.07686567838331768 Train Accuracy: 97.82%
Iteration: 3000 Train Loss: 0.04448306795172992 Test Loss: 0.09844167433139839
Iteration: 3200 Train Loss: 0.10112386562169841 Test Loss: 0.11256770863893274
Iteration: 3400 Train Loss: 0.019454135402578315 Test Loss: 0.09581362482374904
Iteration: 3600 Train Loss: 0.07353215434132872 Test Loss: 0.10108876042998101
Epoch: 4 Mean Train Loss: 0.06265721264935109 Train Accuracy: 98.24%
Iteration: 3800 Train Loss: 0.014081715696161269 Test Loss: 0.10633337803665238
Iteration: 4000 Train Loss: 0.06989759209084048 Test Loss: 0.10274851245228216
Iteration: 4200 Train Loss: 0.020197766165766 Test Loss: 0.09652640048991132
Iteration: 4400 Train Loss: 0.013831990720626039 Test Loss: 0.0919925264284695
Iteration: 4600 Train Loss: 0.0912282319501314 Test Loss: 0.09239814673198671
Epoch: 5 Mean Train Loss: 0.05409265556375774 Train Accuracy: 98.50%
Iteration: 4800 Train Loss: 0.08422536871193329 Test Loss: 0.09404788117986547
Iteration: 5000 Train Loss: 0.02674423160663713 Test Loss: 0.08905715942474246
Iteration: 5200 Train Loss: 0.038092600742980695 Test Loss: 0.08936616544414533
Iteration: 5400 Train Loss: 0.02339687750162397 Test Loss: 0.09065311227332723
Iteration: 5600 Train Loss: 0.013162928215895285 Test Loss: 0.09674740024641142
Epoch: 6 Mean Train Loss: 0.04915578755955072 Train Accuracy: 98.67%
Iteration: 5800 Train Loss: 0.09045078772265583 Test Loss: 0.0988331950471664
Iteration: 6000 Train Loss: 0.04173470948992464 Test Loss: 0.09407221614523362
Iteration: 6200 Train Loss: 0.06435020455247738 Test Loss: 0.09461398588502669
Iteration: 6400 Train Loss: 0.031151015949583572 Test Loss: 0.09346582232863643
Epoch: 7 Mean Train Loss: 0.04982857912629627 Train Accuracy: 98.73%
Iteration: 6600 Train Loss: 0.04781751874619305 Test Loss: 0.09279754245390358
Iteration: 6800 Train Loss: 0.01285330627308873 Test Loss: 0.08733022442581716
Iteration: 7000 Train Loss: 0.03447695840971804 Test Loss: 0.10121611346480944
```

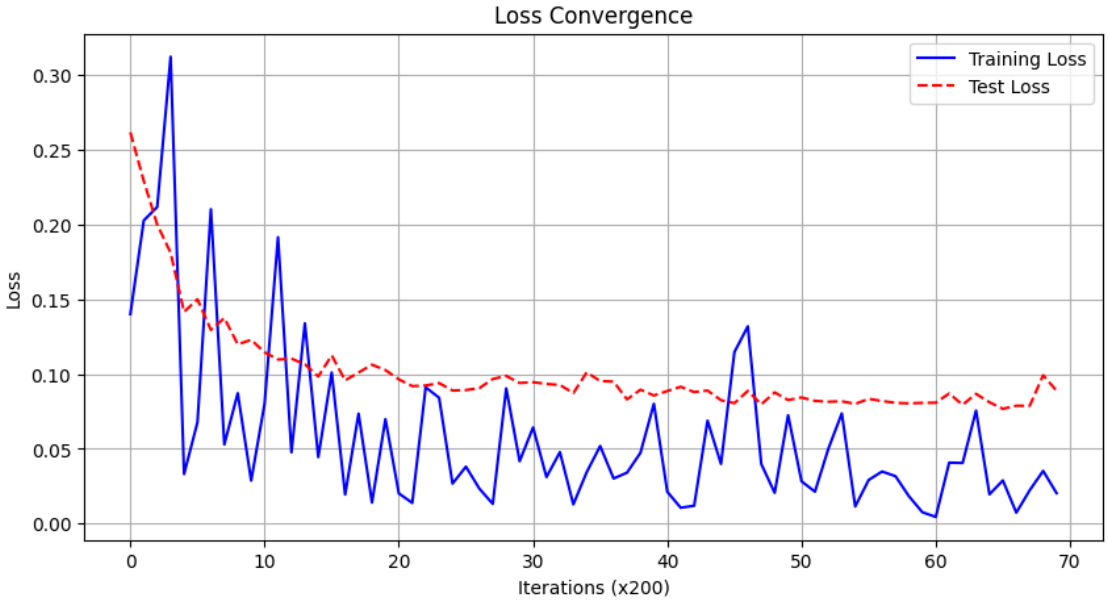
```
Iteration: 7200 Train Loss: 0.051905740923037694 Test Loss: 0.09536962623997453
Iteration: 7400 Train Loss: 0.030170022777738937 Test Loss: 0.09501352812635083
Epoch: 8 Mean Train Loss: 0.04551177484401545 Train Accuracy: 98.81%
Iteration: 7600 Train Loss: 0.03409096461119031 Test Loss: 0.0830994341926956
Iteration: 7800 Train Loss: 0.04736738018292044 Test Loss: 0.0896061423598668
Iteration: 8000 Train Loss: 0.0801652310387229 Test Loss: 0.085738405094537
Iteration: 8200 Train Loss: 0.02128713866148858 Test Loss: 0.08870460763793339
Iteration: 8400 Train Loss: 0.010597265697688919 Test Loss: 0.09157393671248606
Epoch: 9 Mean Train Loss: 0.04061725280246776 Train Accuracy: 98.94%
Iteration: 8600 Train Loss: 0.011936333578124176 Test Loss: 0.08802160123337877
Iteration: 8800 Train Loss: 0.0688718867076607 Test Loss: 0.08902846779244755
Iteration: 9000 Train Loss: 0.03990120975766964 Test Loss: 0.08258019319654941
Iteration: 9200 Train Loss: 0.11472095007893712 Test Loss: 0.08057273042497354
Epoch: 10 Mean Train Loss: 0.04021958087785197 Train Accuracy: 98.99%
Iteration: 9400 Train Loss: 0.13202660520753198 Test Loss: 0.0888104994770528
Iteration: 9600 Train Loss: 0.039866228429556186 Test Loss: 0.07980018922286546
Iteration: 9800 Train Loss: 0.02057412669405713 Test Loss: 0.08779619961568155
Iteration: 10000 Train Loss: 0.07250673758223802 Test Loss: 0.08268182291975451
Iteration: 10200 Train Loss: 0.028301342916912615 Test Loss: 0.08438170122512967
Epoch: 11 Mean Train Loss: 0.04201392170885621 Train Accuracy: 98.96%
Iteration: 10400 Train Loss: 0.02134495694453486 Test Loss: 0.0821316847479729
Iteration: 10600 Train Loss: 0.05012653625672986 Test Loss: 0.08153449995077615
Iteration: 10800 Train Loss: 0.07373819680965894 Test Loss: 0.08183972434240364
Iteration: 11000 Train Loss: 0.011450579480236027 Test Loss: 0.08008243207197409
Iteration: 11200 Train Loss: 0.029174608311892614 Test Loss: 0.08347214084610882
Epoch: 12 Mean Train Loss: 0.039033450894087975 Train Accuracy: 99.02%
Iteration: 11400 Train Loss: 0.03485205057624169 Test Loss: 0.08202513634587535
Iteration: 11600 Train Loss: 0.031725285576177584 Test Loss: 0.08079376811383707
Iteration: 11800 Train Loss: 0.018330030736343487 Test Loss: 0.0804495580544582
Iteration: 12000 Train Loss: 0.007615182260896193 Test Loss: 0.0807888357012485
Epoch: 13 Mean Train Loss: 0.03757873802292434 Train Accuracy: 99.09%
Iteration: 12200 Train Loss: 0.004450343745791111 Test Loss: 0.08087935765304476
Iteration: 12400 Train Loss: 0.04078206046803984 Test Loss: 0.08706630447142773
Iteration: 12600 Train Loss: 0.040559199183130834 Test Loss: 0.07948915593793172
Iteration: 12800 Train Loss: 0.07563565726258326 Test Loss: 0.08684367311397603
Iteration: 13000 Train Loss: 0.019567543671799997 Test Loss: 0.08121622381325698
Epoch: 14 Mean Train Loss: 0.03679511384077449 Train Accuracy: 99.09%
Iteration: 13200 Train Loss: 0.0288734243486701 Test Loss: 0.07678087394326612
Iteration: 13400 Train Loss: 0.007314084108835471 Test Loss: 0.07886284033264011
Iteration: 13600 Train Loss: 0.02230173728627148 Test Loss: 0.07881292606737511
Iteration: 13800 Train Loss: 0.035256496339939915 Test Loss: 0.09926582709239382
Iteration: 14000 Train Loss: 0.020362844171005718 Test Loss: 0.08888825236765963
Epoch: 15 Mean Train Loss: 0.03613594654955485 Train Accuracy: 99.12%
Model trained!
```

[57]:
```
momentum_model_1by1000alpha.evaluate(X_train, Y_train, X_test, Y_test)
```

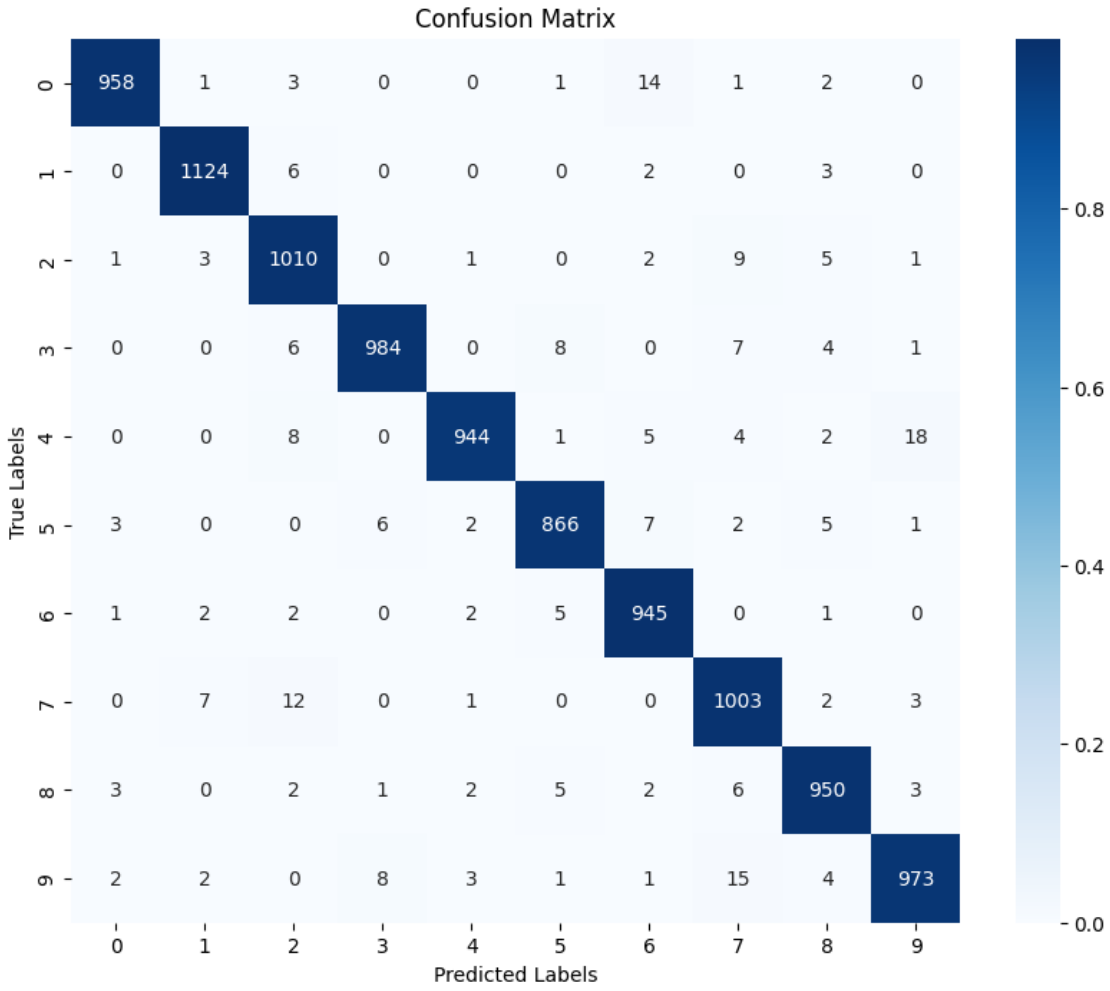
```
Train loss: 0.03628649499066147 Train accuracy: 99.13%
```

```
Test loss: 0.08997512948958328 Test accuracy: 97.57%

Confusion Matrix:
 [[ 958    1    3    0    0    1   14    1    2    0]
 [   0 1124    6    0    0    0    2    0    3    0]
 [   1    3 1010    0    1    0    2    9    5    1]
 [   0    0    6  984    0    8    0    7    4    1]
 [   0    0    8    0  944    1    5    4    2   18]
 [   3    0    0    6    2  866    7    2    5    1]
 [   1    2    2    0    2    5  945    0    1    0]
 [   0    7   12    0    1    0    0 1003    2    3]
 [   3    0    2    1    2    5    2    6  950    3]
 [   2    2    0    8    3    1    1   15    4  973]]

Classification Report:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       980
           1       0.99      0.99      0.99      1135
           2       0.96      0.98      0.97      1032
           3       0.98      0.97      0.98      1010
           4       0.99      0.96      0.97       982
           5       0.98      0.97      0.97       892
           6       0.97      0.99      0.98       958
           7       0.96      0.98      0.97      1028
           8       0.97      0.98      0.97       974
           9       0.97      0.96      0.97      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

[58]:
```
momentum_model_1by1000alpha.plot_losses()
```



[59]:
```
momentum_model_1by1000alpha.plot_confusion_matrix(X_test, Y_test)
```



11.4.3 Conclusion

Regularization value is preety low and performance is similar to when there is 0 weight decay. however, the overfitting is slighlty less in this case.

11.5 Momentum with no weight decay

I have to run it again, because the previous results can't be accessed in this runtime

[60]:
```
input_dim = 28 * 28
output_dim = 10

layers = [
    FNNLayer(input_dim, 500),
    ReLU(),
    FNNLayer(500, 250),
    ReLU(),
```

```
    FNNLayer(250, 100),
    ReLU(),
    FNNLayer(100, output_dim)
]

best_model_momentum = NN(layers)
optimizer = Momentum()
loss = LogLoss()
best_model_momentum.compile(loss=loss, optimizer=optimizer)
best_model_momentum.fit(X_train, Y_train, X_test, Y_test)
```

```
Iteration: 200 Train Loss: 0.29511228595958583 Test Loss: 0.29043277426825964
Iteration: 400 Train Loss: 0.20196479220999358 Test Loss: 0.20543085299189284
Iteration: 600 Train Loss: 0.25417783006938643 Test Loss: 0.20233656358153515
Iteration: 800 Train Loss: 0.17070943502240643 Test Loss: 0.17171421267513934
Epoch: 1 Mean Train Loss: 0.27675109148683513 Train Accuracy: 92.36%
Iteration: 1000 Train Loss: 0.3071961127663855 Test Loss: 0.15803620807412064
Iteration: 1200 Train Loss: 0.06952389685672869 Test Loss: 0.1388915068203119
Iteration: 1400 Train Loss: 0.1205475890193361 Test Loss: 0.13475591142211144
Iteration: 1600 Train Loss: 0.14974644664971043 Test Loss: 0.12260378322325236
Iteration: 1800 Train Loss: 0.021792207181794573 Test Loss: 0.12710132475199182
Epoch: 2 Mean Train Loss: 0.09882154075915348 Train Accuracy: 97.12%
Iteration: 2000 Train Loss: 0.006480515709215295 Test Loss: 0.12043446872030 26
Iteration: 2200 Train Loss: 0.0527387002367363 Test Loss: 0.13469315353325
Iteration: 2400 Train Loss: 0.04847156576259353 Test Loss: 0.11557502755984486
Iteration: 2600 Train Loss: 0.0639364783789816 Test Loss: 0.1008160207204822
Iteration: 2800 Train Loss: 0.17203492756035496 Test Loss: 0.11084747592800874
Epoch: 3 Mean Train Loss: 0.05748608341519606 Train Accuracy: 98.33%
Iteration: 3000 Train Loss: 0.01799088426752149 Test Loss: 0.10799166331285089
Iteration: 3200 Train Loss: 0.040296921897575354 Test Loss: 0.10714253885726978
Iteration: 3400 Train Loss: 0.030644511126134456 Test Loss: 0.1061633220610168
Iteration: 3600 Train Loss: 0.07240964528666824 Test Loss: 0.11005484917856498
Epoch: 4 Mean Train Loss: 0.036755369327803294 Train Accuracy: 98.98%
Iteration: 3800 Train Loss: 0.01037507854891295 Test Loss: 0.10848333155818585
Iteration: 4000 Train Loss: 0.004574767781476898 Test Loss: 0.11094473334659327
Iteration: 4200 Train Loss: 0.0103197200838343 Test Loss: 0.10518804473139037
Iteration: 4400 Train Loss: 0.005557586274700368 Test Loss: 0.11605861841810311
Iteration: 4600 Train Loss: 0.04139361747965342 Test Loss: 0.10531594014369408
Epoch: 5 Mean Train Loss: 0.025836439731458313 Train Accuracy: 99.32%
Iteration: 4800 Train Loss: 0.0050111476526202466 Test Loss: 0.10537485413822921
Iteration: 5000 Train Loss: 0.017850037515308056 Test Loss: 0.11308142701582416
Iteration: 5200 Train Loss: 0.00037733459271372045 Test Loss:
0.11295940039104616
Iteration: 5400 Train Loss: 0.0040100014867448484 Test Loss: 0.10568140728449336
Iteration: 5600 Train Loss: 0.03554287783132157 Test Loss: 0.10169658711372109
Epoch: 6 Mean Train Loss: 0.017193305495359808 Train Accuracy: 99.59%
Iteration: 5800 Train Loss: 0.0016040058204135313 Test Loss: 0.10669379866483653
```

Iteration: 6000 Train Loss: 0.004982041572410928 Test Loss: 0.11003772195184085
Iteration: 6200 Train Loss: 0.0014814117838502536 Test Loss: 0.10851870820106974
Iteration: 6400 Train Loss: 0.019216097011622423 Test Loss: 0.10955111869748288
Epoch: 7 Mean Train Loss: 0.00890121242824097 Train Accuracy: 99.81%
Iteration: 6600 Train Loss: 0.003108528123141075 Test Loss: 0.11066442267383686
Iteration: 6800 Train Loss: 0.0005262082774805184 Test Loss: 0.10844958401567034
Iteration: 7000 Train Loss: 0.005458146607946574 Test Loss: 0.11005792468131766
Iteration: 7200 Train Loss: 0.0016813415400890013 Test Loss: 0.11144693387044761
Iteration: 7400 Train Loss: 0.020080093388082445 Test Loss: 0.10897424387601219
Epoch: 8 Mean Train Loss: 0.004509391112267932 Train Accuracy: 99.92%
Iteration: 7600 Train Loss: 0.0029624470761741213 Test Loss: 0.11165920362843239
Iteration: 7800 Train Loss: 0.001821243484595416 Test Loss: 0.11245564373149737
Iteration: 8000 Train Loss: 0.0003680781037308256 Test Loss: 0.11478343851009562
Iteration: 8200 Train Loss: 0.00031086714750194744 Test Loss:
0.11492134654178497
Iteration: 8400 Train Loss: 0.001824738249331347 Test Loss: 0.11406450855643918
Epoch: 9 Mean Train Loss: 0.0028126653279315255 Train Accuracy: 99.97%
Iteration: 8600 Train Loss: 0.0003550567845530616 Test Loss: 0.11443165233441183
Iteration: 8800 Train Loss: 0.0013682793732224634 Test Loss: 0.11720469497801683
Iteration: 9000 Train Loss: 0.0005069678891118632 Test Loss: 0.11473150293675462
Iteration: 9200 Train Loss: 0.00021807099145491958 Test Loss:
0.11592164054733083
Epoch: 10 Mean Train Loss: 0.0015254104299818262 Train Accuracy: 99.98%
Iteration: 9400 Train Loss: 0.0004334094332221944 Test Loss: 0.11571915518809822
Iteration: 9600 Train Loss: 0.00036192074520393963 Test Loss:
0.11759561502781543
Iteration: 9800 Train Loss: 0.0007553527936478932 Test Loss: 0.11810886310279249
Iteration: 10000 Train Loss: 0.00020656904729928372 Test Loss:
0.1191418742228573
Iteration: 10200 Train Loss: 0.0014558383726768498 Test Loss:
0.11919995483887812
Epoch: 11 Mean Train Loss: 0.0011505184922847862 Train Accuracy: 99.99%
Iteration: 10400 Train Loss: 9.951609224320557e-05 Test Loss:
0.12105544828237275
Iteration: 10600 Train Loss: 0.0019763676467846633 Test Loss:
0.11989768851979128
Iteration: 10800 Train Loss: 0.0001900586876460055 Test Loss:
0.12023577152892619
Iteration: 11000 Train Loss: 0.0004662728011437591 Test Loss:
0.12017551270799742
Iteration: 11200 Train Loss: 0.0003931427233366271 Test Loss:
0.12073087200430863
Epoch: 12 Mean Train Loss: 0.0007664855876472559 Train Accuracy: 100.00%
Iteration: 11400 Train Loss: 0.0008290959258568464 Test Loss:
0.12093562556782354
Iteration: 11600 Train Loss: 0.00026967236119651003 Test Loss:
0.12171350731872146
Iteration: 11800 Train Loss: 0.00047033687865882746 Test Loss: 0.122438467500847

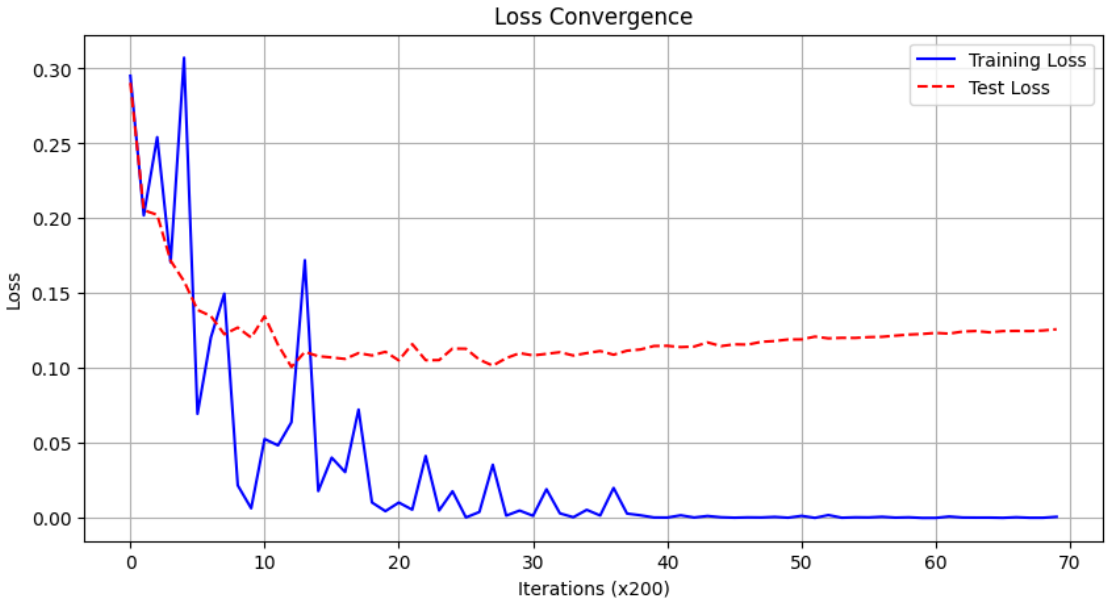
```
Iteration: 12000 Train Loss: 5.294831868412429e-05 Test Loss:
0.12282910495386111
Epoch: 13 Mean Train Loss: 0.0005474458072395725 Train Accuracy: 100.00%
Iteration: 12200 Train Loss: 7.560605360183069e-05 Test Loss:
0.12354006627692421
Iteration: 12400 Train Loss: 0.0010109500102481461 Test Loss:
0.12305231829665247
Iteration: 12600 Train Loss: 0.00034295882124954827 Test Loss:
0.12443945414516007
Iteration: 12800 Train Loss: 0.0002747392302047569 Test Loss:
0.12480252099676799
Iteration: 13000 Train Loss: 0.00027761212762083 Test Loss: 0.12391471790124532
Epoch: 14 Mean Train Loss: 0.0005889750564241362 Train Accuracy: 100.00%
Iteration: 13200 Train Loss: 0.0001267523605892084 Test Loss:
0.12474939130785448
Iteration: 13400 Train Loss: 0.0005130888064323857 Test Loss:
0.12486316965427974
Iteration: 13600 Train Loss: 0.00016462830670730275 Test Loss:
0.1247376267603713
Iteration: 13800 Train Loss: 0.0001768453453740823 Test Loss:
0.12514791124077523
Iteration: 14000 Train Loss: 0.0008078004983985048 Test Loss:
0.12598056855689796
Epoch: 15 Mean Train Loss: 0.00048535631736061375 Train Accuracy: 100.00%
Model trained!
```

[61]:
```
best_model_momentum.evaluate(X_train, Y_train, X_test, Y_test)
```

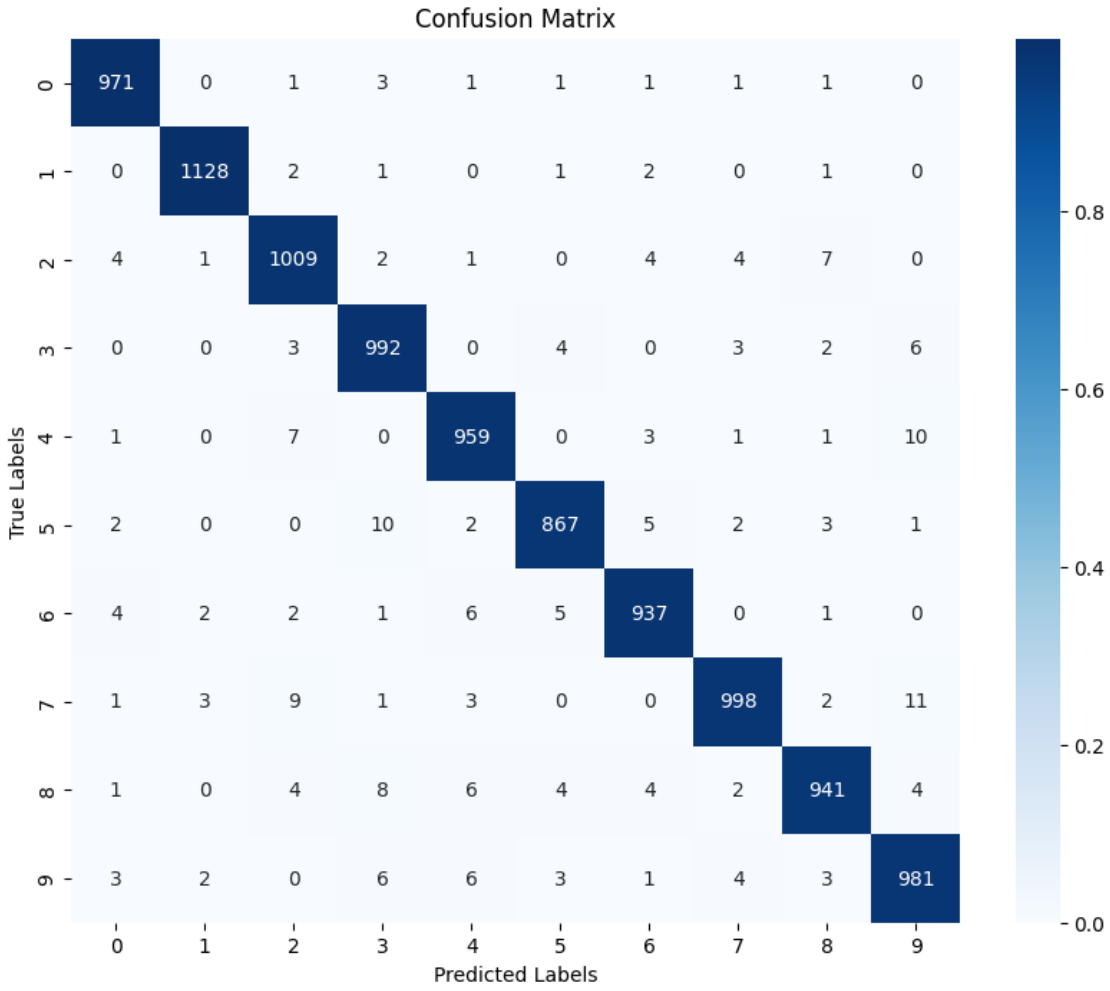
```
Train loss: 0.00036201082777796836 Train accuracy: 100.00%
Test loss: 0.126786125912051 Test accuracy: 97.83%

Confusion Matrix:
 [[ 971    0    1    3    1    1    1    1    1    0]
 [   0 1128    2    1    0    1    2    0    1    0]
 [   4    1 1009    2    1    0    4    4    7    0]
 [   0    0    3  992    0    4    0    3    2    6]
 [   1    0    7    0  959    0    3    1    1   10]
 [   2    0    0   10    2  867    5    2    3    1]
 [   4    2    2    1    6    5  937    0    1    0]
 [   1    3    9    1    3    0    0  998    2   11]
 [   1    0    4    8    6    4    4    2  941    4]
 [   3    2    0    6    6    3    1    4    3  981]]

Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      1135
```

```
              2       0.97      0.98      0.98      1032
              3       0.97      0.98      0.98      1010
              4       0.97      0.98      0.98       982
              5       0.98      0.97      0.98       892
              6       0.98      0.98      0.98       958
              7       0.98      0.97      0.98      1028
              8       0.98      0.97      0.97       974
              9       0.97      0.97      0.97      1009

       accuracy                           0.98     10000
      macro avg       0.98      0.98      0.98     10000
   weighted avg       0.98      0.98      0.98     10000
```

[62]:
```
best_model_momentum.plot_losses()
```



[63]:
```
best_model_momentum.plot_confusion_matrix(X_test, Y_test)
```



11.6 Comparison

We compare the perfromance with the Momentum optimizer, as it has been the best for performer in all the runs so far.

[64]:
```
import matplotlib.pyplot as plt

def plot_models_losses(models, labels, title, attribute):
    plt.figure(figsize=(12, 7))

    colors = ['blue', 'purple', 'green', 'red', 'black']

    for i, model in enumerate(models):
        losses = getattr(model, attribute)
        plt.plot(losses, label=labels[i], color=colors[i])
```

```
    plt.xlabel('Iterations (x200)')
    plt.ylabel('Loss')
    plt.title(title)
    plt.legend()
    plt.grid(True)
    plt.show()

model1 = momentum_model_halfalpha
model2 = momentum_model_1by10alpha
model3 = momentum_model_1by100alpha
model4 = momentum_model_1by1000alpha
model5 = best_model_momentum

models = [model1, model2, model3, model4, model5]
labels = ['0.5', '0.1', '0.01', '0.001', '0']

# Plots training iteration losses
plot_models_losses(models, labels, 'Training Iteration Losses for different␣
 ↪Weight Decays', 'iteration_losses')

# Plots test iteration losses
plot_models_losses(models, labels, 'Test Iteration Losses for different Weight␣
 ↪Decays', 'iteration_test_losses')
```

11.7 Observations

It is clear that models with very high regularization (0.5) are hardly learning, and the accuracy close to 10% shows that the model is almost mimicing the behaviour of making a guess for the class.

We see the test acuuracy improves as we keep decreasing the regularization, however the gap between train accuracy/loss and test accuracy/loss also increases which would indicate overfitting. However, the overfitting is quite minimal for this dataset in all cases, and can be excused in order to get a good accuracy.

Summary of results:

Weight Decay	Test Accuracy	Train Accuracy	Train Loss	Test Loss
0.5	11.35%	11.24%	2.3025	2.3024
0.1	41.61%	42.02%	1.5583	1.5578
0.01	94.76%	95.26%	0.1961	0.2046
0.001	97.57%	99.13%	0.0362	0.0899
0	97.83%	100.00%	0.00036	0.1268