assign2_gpu

November 2, 2017

1 Computer Vision CSCI-GA.2272-001 Assignment 2

1.1 Introduction

This assignment is an introduction to using PyTorch for training simple neural net models. Two different datasets will be used: - MNIST digits [handwritten digits] - CIFAR-10 [32x32 resolution color images of 10 object classes].

1.2 Requirements

You should perform this assignment in PyTorch, modify this ipython notebook
    To install PyTorch, follow instructions at http://pytorch.org/

```
In [1]: from __future__ import print_function
        import argparse
        import torch
        import torch.nn as nn
        import torch.nn.functional as F
        import torch.optim as optim
        from torchvision import datasets, transforms, utils
        from torch.autograd import Variable

In [2]: # options
        dataset = 'mnist' # options: 'mnist' | 'cifar10'
        batch_size = 64    # input batch size for training
        epochs = 10        # number of epochs to train
        lr = 0.01          # learning rate
```

1.3 Warmup [10%]

It is always good practice to visually inspect your data before trying to train a model, since it lets you check for problems and get a feel for the task at hand.
    MNIST is a dataset of 70,000 grayscale hand-written digits (0 through 9). 60,000 of these are training images. 10,000 are a held out test set.
    CIFAR-10 is a dataset of 60,000 color images (32 by 32 resolution) across 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The train/test split is 50k/10k.
    Use `matplotlib` and ipython notebook's visualization capabilities to display some of these images. See this PyTorch tutorial page for hints on how to achieve this.
    ** Relevant Cell: "Data Loading" **

```
In [3]:  # Data Loading
         # Warning: this cell might take some time when you run it for the first time,
         #          because it will download the datasets from the internet
         if dataset == 'mnist':
             data_transform = transforms.Compose([
                 transforms.ToTensor(),
                 transforms.Normalize((0.1307,), (0.3081,))
         #        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
             ])
             trainset = datasets.MNIST(root='.', train=True, download=True, transform=data_transf
             testset = datasets.MNIST(root='.', train=False, download=True, transform=data_transf

         elif dataset == 'cifar10':
             data_transform = transforms.Compose([
                 transforms.ToTensor(),
                 transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
             ])
             trainset = datasets.CIFAR10(root='.', train=True, download=True, transform=data_tran
             testset = datasets.CIFAR10(root='.', train=False, download=True, transform=data_tran

         train_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size, shuffle=True
         test_loader  = torch.utils.data.DataLoader(testset, batch_size=batch_size, shuffle=False

In [4]:  %matplotlib inline

         import matplotlib.pyplot as plt
         import numpy as np

         class UnNormalize(object):
             def __init__(self, mean, std):
                 self.mean = mean
                 self.std = std

             def __call__(self, tensor):
                 """
                 Args:
                     tensor (Tensor): Tensor image of size (C, H, W) to be normalized.
                 Returns:
                     Tensor: Normalized image.
                 """
                 for t, m, s in zip(tensor, self.mean, self.std):
                     t.mul_(s).add_(m)
                     # The normalize code -> t.sub_(m).div_(s)
                 return tensor


         # Does not work for some unknown (yet to be found) reason
         def imshow2(img):
```

2

```python
    if dataset =='mnist':
        unorm = UnNormalize(mean=(0.1307,), std=(0.3081,))
    elif dataset == 'cifar10':
        unorm = UnNormalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))

    img = unorm(img)
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()


# functions to show an image
def imshow(img):
    if dataset =='mnist':
        img = img*0.3081 + 0.1307 # unnormalize
    elif dataset == 'cifar10':
        img = img / 2 + 0.5       # unnormalize

    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()


# get some random training images
dataiter = iter(train_loader)
images, labels = dataiter.next()

# show images
imshow(utils.make_grid(images))
```
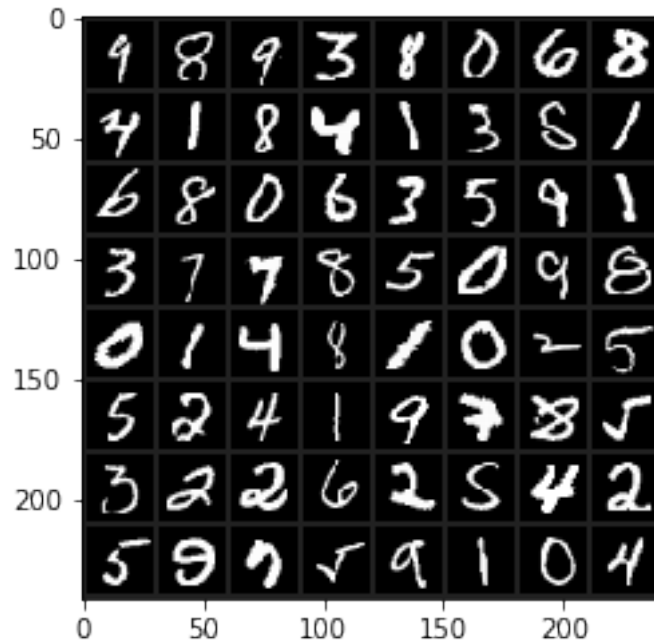
1.4 Training a Single Layer Network on MNIST [20%]

Start by running the training on MNIST. By default if you run this notebook successfully, it will train on MNIST.

This will initialize a single layer model train it on the 50,000 MNIST training images for 10 epochs (passes through the training data).

The loss function cross_entropy computes a Logarithm of the Softmax on the output of the neural network, and then computes the negative log-likelihood w.r.t. the given target.

The default values for the learning rate, batch size and number of epochs are given in the "options" cell of this notebook. Unless otherwise specified, use the default values throughout this assignment.

Note the decrease in training loss and corresponding decrease in validation errors.

Paste the output into your report.

(a): Add code to plot out the network weights as images (one for each output, of size 28 by 28) after the last epoch. Grab a screenshot of the figure and include it in your report. (Hint threads: #1 #2)

```
In [5]: ## network and optimizer
        if dataset == 'mnist':
            num_inputs = 784
        elif dataset == 'cifar10':
            num_inputs = 3072

        num_outputs = 10 # same for both CIFAR10 and MNIST, both have 10 classes as outputs
```

4

```python
class Net(nn.Module):
    def __init__(self, num_inputs, num_outputs):
        super(Net, self).__init__()
        self.linear = nn.Linear(num_inputs, num_outputs)

    def forward(self, input):
        input = input.view(-1, num_inputs) # reshape input to batch x num_inputs
        output = self.linear(input)
        return output

network = Net(num_inputs, num_outputs)
network.cuda()
optimizer = optim.SGD(network.parameters(), lr=lr)
```

```python
In [6]: def train(epoch):
    network.train()
    loss_collect = []
    for batch_idx, (data, target) in enumerate(train_loader):
        # wrap inputs in Variable
        data, target = Variable(data.cuda()), Variable(target.cuda())

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        output = network(data)
        loss = F.cross_entropy(output, target)
        loss.backward()
        optimizer.step()
        loss_collect.append(loss.data[0])

        # print statistics
        if batch_idx % 100 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.data[0]))

    return loss_collect

def test():
    network.eval()
    test_loss = 0
    correct = 0
    for data, target in test_loader:
        data, target = Variable(data.cuda(), volatile=True), Variable(target.cuda())
        output = network(data)
        test_loss += F.cross_entropy(output, target, size_average=False).data[0] # sum u
        pred = output.data.max(1, keepdim=True)[1] # get the index of the max log-probab
```

```
                correct += pred.eq(target.data.view_as(pred)).cpu().sum()

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

In [7]: loss_g = []
        for epoch in range(1, 1 + epochs):
            loss_g.extend(train(epoch))
            test()
```

```
Train Epoch: 1 [0/60000 (0%)]          Loss: 2.432553
Train Epoch: 1 [6400/60000 (11%)]        Loss: 0.830794
Train Epoch: 1 [12800/60000 (21%)]        Loss: 0.524474
Train Epoch: 1 [19200/60000 (32%)]        Loss: 0.263197
Train Epoch: 1 [25600/60000 (43%)]        Loss: 0.561578
Train Epoch: 1 [32000/60000 (53%)]        Loss: 0.516961
Train Epoch: 1 [38400/60000 (64%)]        Loss: 0.279667
Train Epoch: 1 [44800/60000 (75%)]        Loss: 0.286319
Train Epoch: 1 [51200/60000 (85%)]        Loss: 0.241411
Train Epoch: 1 [57600/60000 (96%)]        Loss: 0.233533

Test set: Average loss: 0.3306, Accuracy: 9089/10000 (91%)

Train Epoch: 2 [0/60000 (0%)]          Loss: 0.436318
Train Epoch: 2 [6400/60000 (11%)]        Loss: 0.494924
Train Epoch: 2 [12800/60000 (21%)]        Loss: 0.476951
Train Epoch: 2 [19200/60000 (32%)]        Loss: 0.452447
Train Epoch: 2 [25600/60000 (43%)]        Loss: 0.411289
Train Epoch: 2 [32000/60000 (53%)]        Loss: 0.194206
Train Epoch: 2 [38400/60000 (64%)]        Loss: 0.402748
Train Epoch: 2 [44800/60000 (75%)]        Loss: 0.380579
Train Epoch: 2 [51200/60000 (85%)]        Loss: 0.403462
Train Epoch: 2 [57600/60000 (96%)]        Loss: 0.501765

Test set: Average loss: 0.3029, Accuracy: 9141/10000 (91%)

Train Epoch: 3 [0/60000 (0%)]          Loss: 0.292264
Train Epoch: 3 [6400/60000 (11%)]        Loss: 0.193598
Train Epoch: 3 [12800/60000 (21%)]        Loss: 0.375727
Train Epoch: 3 [19200/60000 (32%)]        Loss: 0.243552
Train Epoch: 3 [25600/60000 (43%)]        Loss: 0.334908
Train Epoch: 3 [32000/60000 (53%)]        Loss: 0.265759
Train Epoch: 3 [38400/60000 (64%)]        Loss: 0.390069
Train Epoch: 3 [44800/60000 (75%)]        Loss: 0.420537
Train Epoch: 3 [51200/60000 (85%)]        Loss: 0.279145
Train Epoch: 3 [57600/60000 (96%)]        Loss: 0.413346
```

```
Test set: Average loss: 0.2974, Accuracy: 9165/10000 (92%)


Train Epoch: 4 [0/60000 (0%)]          Loss: 0.631925
Train Epoch: 4 [6400/60000 (11%)]       Loss: 0.188420
Train Epoch: 4 [12800/60000 (21%)]       Loss: 0.332395
Train Epoch: 4 [19200/60000 (32%)]       Loss: 0.293472
Train Epoch: 4 [25600/60000 (43%)]       Loss: 0.275343
Train Epoch: 4 [32000/60000 (53%)]       Loss: 0.384579
Train Epoch: 4 [38400/60000 (64%)]       Loss: 0.200331
Train Epoch: 4 [44800/60000 (75%)]       Loss: 0.281195
Train Epoch: 4 [51200/60000 (85%)]       Loss: 0.403520
Train Epoch: 4 [57600/60000 (96%)]       Loss: 0.386153


Test set: Average loss: 0.2867, Accuracy: 9188/10000 (92%)


Train Epoch: 5 [0/60000 (0%)]          Loss: 0.229332
Train Epoch: 5 [6400/60000 (11%)]       Loss: 0.355931
Train Epoch: 5 [12800/60000 (21%)]       Loss: 0.220135
Train Epoch: 5 [19200/60000 (32%)]       Loss: 0.265769
Train Epoch: 5 [25600/60000 (43%)]       Loss: 0.371632
Train Epoch: 5 [32000/60000 (53%)]       Loss: 0.473440
Train Epoch: 5 [38400/60000 (64%)]       Loss: 0.224874
Train Epoch: 5 [44800/60000 (75%)]       Loss: 0.168951
Train Epoch: 5 [51200/60000 (85%)]       Loss: 0.140584
Train Epoch: 5 [57600/60000 (96%)]       Loss: 0.438026


Test set: Average loss: 0.2842, Accuracy: 9198/10000 (92%)


Train Epoch: 6 [0/60000 (0%)]          Loss: 0.235925
Train Epoch: 6 [6400/60000 (11%)]       Loss: 0.184482
Train Epoch: 6 [12800/60000 (21%)]       Loss: 0.232548
Train Epoch: 6 [19200/60000 (32%)]       Loss: 0.148201
Train Epoch: 6 [25600/60000 (43%)]       Loss: 0.135537
Train Epoch: 6 [32000/60000 (53%)]       Loss: 0.306238
Train Epoch: 6 [38400/60000 (64%)]       Loss: 0.247220
Train Epoch: 6 [44800/60000 (75%)]       Loss: 0.412690
Train Epoch: 6 [51200/60000 (85%)]       Loss: 0.286281
Train Epoch: 6 [57600/60000 (96%)]       Loss: 0.187953


Test set: Average loss: 0.2819, Accuracy: 9213/10000 (92%)


Train Epoch: 7 [0/60000 (0%)]          Loss: 0.215312
Train Epoch: 7 [6400/60000 (11%)]       Loss: 0.164351
Train Epoch: 7 [12800/60000 (21%)]       Loss: 0.222654
Train Epoch: 7 [19200/60000 (32%)]       Loss: 0.416943
Train Epoch: 7 [25600/60000 (43%)]       Loss: 0.121763
Train Epoch: 7 [32000/60000 (53%)]       Loss: 0.147365
```

```
Train Epoch: 7 [38400/60000 (64%)]        Loss: 0.351305
Train Epoch: 7 [44800/60000 (75%)]        Loss: 0.373331
Train Epoch: 7 [51200/60000 (85%)]        Loss: 0.400591
Train Epoch: 7 [57600/60000 (96%)]        Loss: 0.196848

Test set: Average loss: 0.2786, Accuracy: 9225/10000 (92%)

Train Epoch: 8 [0/60000 (0%)]        Loss: 0.296293
Train Epoch: 8 [6400/60000 (11%)]         Loss: 0.629407
Train Epoch: 8 [12800/60000 (21%)]        Loss: 0.157068
Train Epoch: 8 [19200/60000 (32%)]        Loss: 0.124238
Train Epoch: 8 [25600/60000 (43%)]        Loss: 0.494769
Train Epoch: 8 [32000/60000 (53%)]        Loss: 0.183174
Train Epoch: 8 [38400/60000 (64%)]        Loss: 0.290588
Train Epoch: 8 [44800/60000 (75%)]        Loss: 0.218211
Train Epoch: 8 [51200/60000 (85%)]        Loss: 0.203376
Train Epoch: 8 [57600/60000 (96%)]        Loss: 0.347856

Test set: Average loss: 0.2780, Accuracy: 9206/10000 (92%)

Train Epoch: 9 [0/60000 (0%)]        Loss: 0.294717
Train Epoch: 9 [6400/60000 (11%)]         Loss: 0.240345
Train Epoch: 9 [12800/60000 (21%)]        Loss: 0.240508
Train Epoch: 9 [19200/60000 (32%)]        Loss: 0.254494
Train Epoch: 9 [25600/60000 (43%)]        Loss: 0.152996
Train Epoch: 9 [32000/60000 (53%)]        Loss: 0.227247
Train Epoch: 9 [38400/60000 (64%)]        Loss: 0.204751
Train Epoch: 9 [44800/60000 (75%)]        Loss: 0.318234
Train Epoch: 9 [51200/60000 (85%)]        Loss: 0.164535
Train Epoch: 9 [57600/60000 (96%)]        Loss: 0.195484

Test set: Average loss: 0.2772, Accuracy: 9209/10000 (92%)

Train Epoch: 10 [0/60000 (0%)]        Loss: 0.365687
Train Epoch: 10 [6400/60000 (11%)]         Loss: 0.263153
Train Epoch: 10 [12800/60000 (21%)]         Loss: 0.263110
Train Epoch: 10 [19200/60000 (32%)]         Loss: 0.184863
Train Epoch: 10 [25600/60000 (43%)]         Loss: 0.199894
Train Epoch: 10 [32000/60000 (53%)]         Loss: 0.274847
Train Epoch: 10 [38400/60000 (64%)]         Loss: 0.309169
Train Epoch: 10 [44800/60000 (75%)]         Loss: 0.258047
Train Epoch: 10 [51200/60000 (85%)]         Loss: 0.100652
Train Epoch: 10 [57600/60000 (96%)]         Loss: 0.275231

Test set: Average loss: 0.2759, Accuracy: 9217/10000 (92%)
```
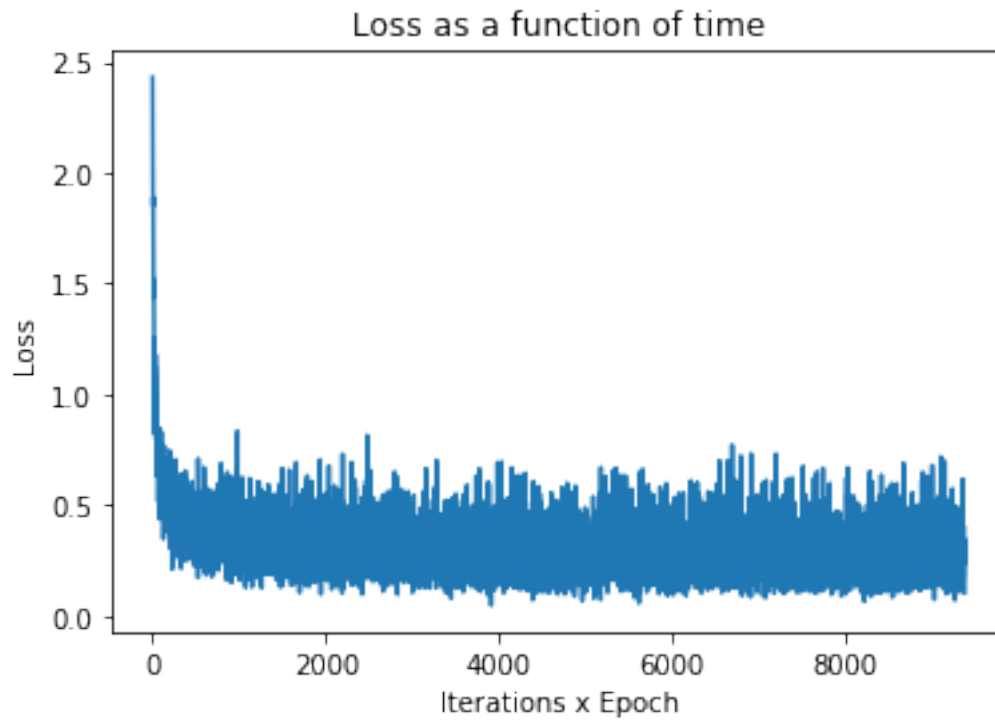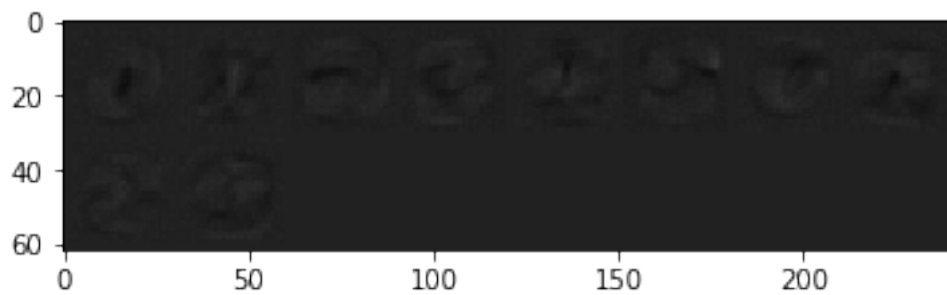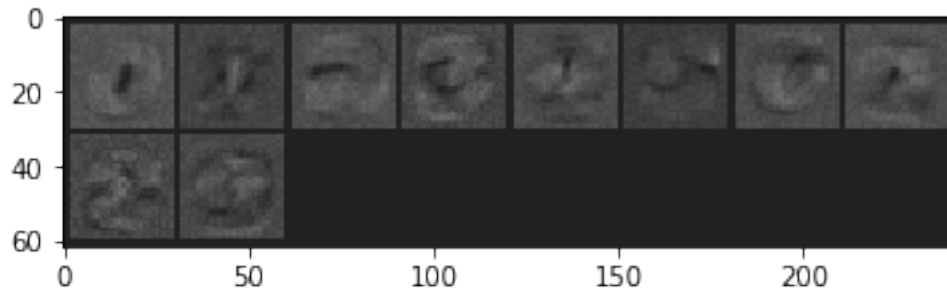
```
In [8]: plt.plot(loss_g)
        plt.xlabel("Iterations x Epoch")
        plt.ylabel("Loss")
        plt.title("Loss as a function of time")
```

```
Out[8]: <matplotlib.text.Text at 0x7faa3008ddd8>
```



```
In [9]: for x in network.modules():
            if isinstance(x, nn.Linear):
                imshow(utils.make_grid(x.weight.data.cpu().view(10,1,28,28)))
                imshow(utils.make_grid(x.weight.data.cpu().view(10,1,28,28),normalize=True, scal
```


9

(b): Reduce the number of training examples to just 50. [Hint: limit the iterator in the `train` function]. Paste the output into your report and explain what is happening to the model.

```
In [10]: from torch.utils.data.sampler import Sampler

         class SubsetSampler(Sampler):
             """Samples elements sequentially, always in the same order.
             Arguments:
                 data_source (Dataset): dataset to sample from
                 num_samples (int): number of samples to draw
             """

             def __init__(self, data_source, num_samples):
                 self.data_source = data_source
                 if num_samples < len(self.data_source):
                     self.num_samples = num_samples
                 else:
                     self.num_samples = len(self.data_source)

             def __iter__(self):
                 return iter(range(self.num_samples))

             def __len__(self):
                 return self.num_samples

In [11]: subset_sampler = SubsetSampler(trainset, 50)
         train_loader = torch.utils.data.DataLoader(trainset, batch_size=1, shuffle=False, sampl
         subset_sampler = SubsetSampler(testset, 50)
         test_loader  = torch.utils.data.DataLoader(testset, batch_size=1, shuffle=False, sample

         network = Net(num_inputs, num_outputs)
         network.cuda()
         optimizer = optim.SGD(network.parameters(), lr=lr)

In [12]: loss_g = []
         for epoch in range(1, 1 + epochs):
```

10

```
            loss_g.extend(train(epoch))
            test()
```

Train Epoch: 1 [0/60000 (0%)]   Loss: 1.818081

Test set: Average loss: 0.0073, Accuracy: 27/10000 (0%)

Train Epoch: 2 [0/60000 (0%)]   Loss: 2.596171

Test set: Average loss: 0.0058, Accuracy: 32/10000 (0%)

Train Epoch: 3 [0/60000 (0%)]   Loss: 0.123463

Test set: Average loss: 0.0053, Accuracy: 34/10000 (0%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 0.063994

Test set: Average loss: 0.0052, Accuracy: 36/10000 (0%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 0.044397

Test set: Average loss: 0.0052, Accuracy: 36/10000 (0%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 0.036086

Test set: Average loss: 0.0051, Accuracy: 36/10000 (0%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 0.030964

Test set: Average loss: 0.0051, Accuracy: 36/10000 (0%)

Train Epoch: 8 [0/60000 (0%)]   Loss: 0.027339

Test set: Average loss: 0.0051, Accuracy: 36/10000 (0%)

Train Epoch: 9 [0/60000 (0%)]   Loss: 0.024580

Test set: Average loss: 0.0051, Accuracy: 36/10000 (0%)

Train Epoch: 10 [0/60000 (0%)]   Loss: 0.022384
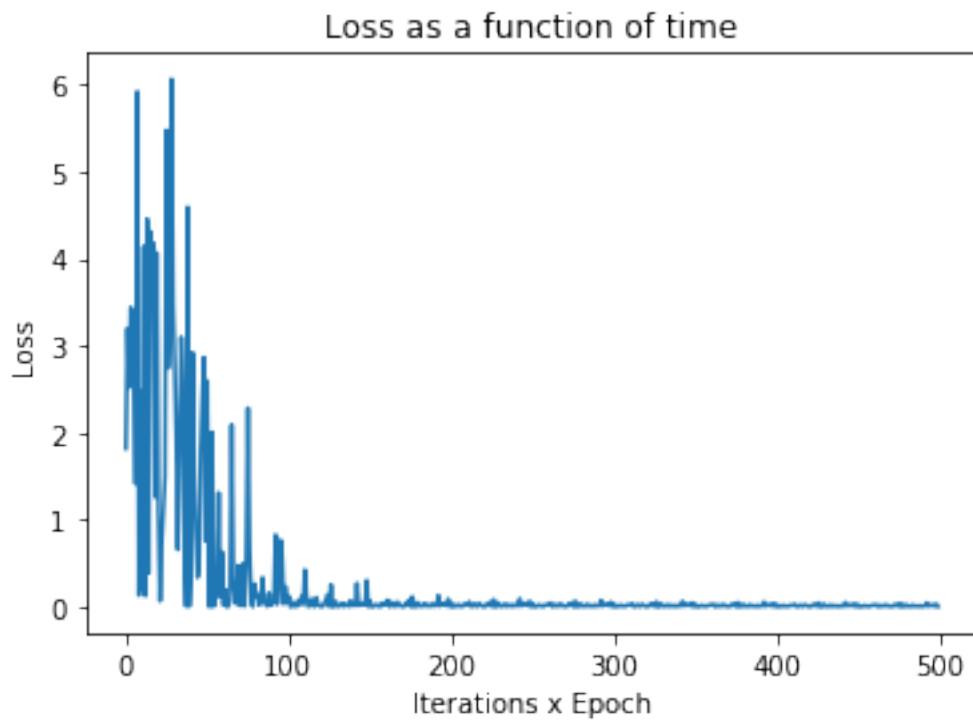
Test set: Average loss: 0.0051, Accuracy: 36/10000 (0%)
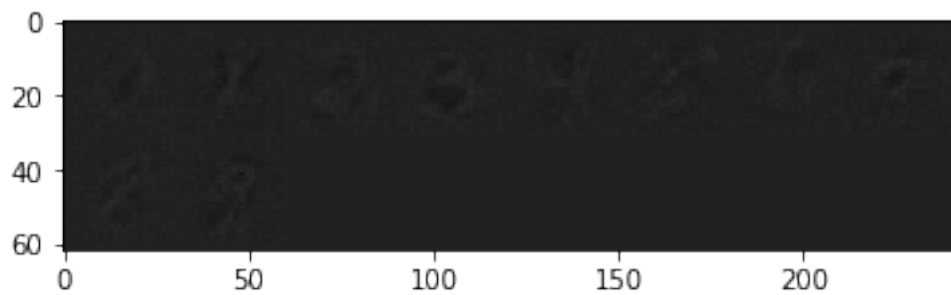

In [13]: plt.plot(loss_g)
         plt.xlabel("Iterations x Epoch")
         plt.ylabel("Loss")
         plt.title("Loss as a function of time")
```

## Loss as a function of time



```
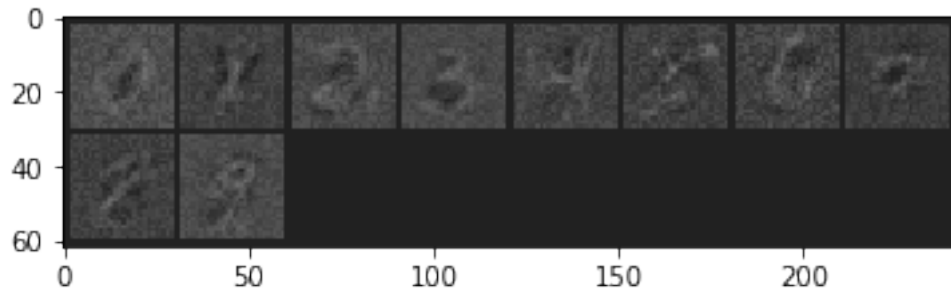In [14]: for x in network.modules():
            if isinstance(x, nn.Linear):
                imshow(utils.make_grid(x.weight.data.cpu().view(10,1,28,28)))
                imshow(utils.make_grid(x.weight.data.cpu().view(10,1,28,28),normalize=True, sca
```

1.4.1   Overfit

As we can see the loss (indicator of training error) quickly becomes very small (tending to 0) while the test error (as training accuracy tends to 0) becomes very high. These are clear indicators of overfitting the data. Since a very small amount of data was provided, the model overfits over it and lacks the ability to generalise at all which leads to very poor test accuracy.

By definition, **overfitting is a modeling error which occurs when a function is too closely fit to a limited set of data points** which is clearly what happens in this case. Overfitting the model generally takes the form of making an overly complex model to explain idiosyncrasies in the data under study. There are many ways to fight overfitting such as the following (not an exhaustive list by any means):

- Get more data (and/or use data augmentation)
- Use regularization, like Dropout and perhaps even L1 and L2
- Feature scale clipping
- Global average pooling
- Make network smaller
- Early stopping

1.5   Training a Multi-Layer Network on MNIST [20%]

- Add an extra layer to the network with 1000 hidden units and a `tanh` non-linearity. [Hint: modify the `Net` class]. Train the model for 10 epochs and save the output into your report.

```
In [15]: train_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size, shuffle=Tru
         test_loader  = torch.utils.data.DataLoader(testset, batch_size=batch_size, shuffle=Fals

         class Net(nn.Module):
             def __init__(self, num_inputs, num_outputs, hidden_units):
                 super(Net, self).__init__()
                 self.fc1 = nn.Linear(num_inputs, hidden_units)
                 self.fc2 = nn.Linear(hidden_units, num_outputs)

             def forward(self, x):
                 x = x.view(-1, num_inputs) # reshape input to batch x num_inputs
                 x = self.fc1(x)
```

```
            x = F.tanh(x)
            x = self.fc2(x)
            return x

        network = Net(num_inputs, num_outputs, hidden_units=1000)
        network.cuda()
        optimizer = optim.SGD(network.parameters(), lr=lr)

In [16]: loss_g = []
         for epoch in range(1, 1 + epochs):
             loss_g.extend(train(epoch))
             test()
```

```
Train Epoch: 1 [0/60000 (0%)]        Loss: 2.344141
Train Epoch: 1 [6400/60000 (11%)]      Loss: 0.829380
Train Epoch: 1 [12800/60000 (21%)]      Loss: 0.549929
Train Epoch: 1 [19200/60000 (32%)]      Loss: 0.466398
Train Epoch: 1 [25600/60000 (43%)]      Loss: 0.401492
Train Epoch: 1 [32000/60000 (53%)]      Loss: 0.534058
Train Epoch: 1 [38400/60000 (64%)]      Loss: 0.398345
Train Epoch: 1 [44800/60000 (75%)]      Loss: 0.492511
Train Epoch: 1 [51200/60000 (85%)]      Loss: 0.323353
Train Epoch: 1 [57600/60000 (96%)]      Loss: 0.269988

Test set: Average loss: 0.3233, Accuracy: 9093/10000 (91%)

Train Epoch: 2 [0/60000 (0%)]        Loss: 0.435899
Train Epoch: 2 [6400/60000 (11%)]      Loss: 0.262041
Train Epoch: 2 [12800/60000 (21%)]      Loss: 0.361291
Train Epoch: 2 [19200/60000 (32%)]      Loss: 0.300726
Train Epoch: 2 [25600/60000 (43%)]      Loss: 0.453793
Train Epoch: 2 [32000/60000 (53%)]      Loss: 0.260048
Train Epoch: 2 [38400/60000 (64%)]      Loss: 0.406989
Train Epoch: 2 [44800/60000 (75%)]      Loss: 0.527357
Train Epoch: 2 [51200/60000 (85%)]      Loss: 0.375693
Train Epoch: 2 [57600/60000 (96%)]      Loss: 0.284525

Test set: Average loss: 0.2813, Accuracy: 9203/10000 (92%)

Train Epoch: 3 [0/60000 (0%)]        Loss: 0.359545
Train Epoch: 3 [6400/60000 (11%)]      Loss: 0.326413
Train Epoch: 3 [12800/60000 (21%)]      Loss: 0.268522
Train Epoch: 3 [19200/60000 (32%)]      Loss: 0.431381
Train Epoch: 3 [25600/60000 (43%)]      Loss: 0.345010
Train Epoch: 3 [32000/60000 (53%)]      Loss: 0.291767
Train Epoch: 3 [38400/60000 (64%)]      Loss: 0.346270
Train Epoch: 3 [44800/60000 (75%)]      Loss: 0.274078
Train Epoch: 3 [51200/60000 (85%)]      Loss: 0.234189
```

```
Train Epoch: 3 [57600/60000 (96%)]        Loss: 0.210701

Test set: Average loss: 0.2619, Accuracy: 9249/10000 (92%)

Train Epoch: 4 [0/60000 (0%)]        Loss: 0.192107
Train Epoch: 4 [6400/60000 (11%)]         Loss: 0.449517
Train Epoch: 4 [12800/60000 (21%)]         Loss: 0.199383
Train Epoch: 4 [19200/60000 (32%)]         Loss: 0.148907
Train Epoch: 4 [25600/60000 (43%)]         Loss: 0.310162
Train Epoch: 4 [32000/60000 (53%)]         Loss: 0.230240
Train Epoch: 4 [38400/60000 (64%)]         Loss: 0.524172
Train Epoch: 4 [44800/60000 (75%)]         Loss: 0.323336
Train Epoch: 4 [51200/60000 (85%)]         Loss: 0.183699
Train Epoch: 4 [57600/60000 (96%)]         Loss: 0.179159

Test set: Average loss: 0.2419, Accuracy: 9318/10000 (93%)

Train Epoch: 5 [0/60000 (0%)]        Loss: 0.305542
Train Epoch: 5 [6400/60000 (11%)]         Loss: 0.322803
Train Epoch: 5 [12800/60000 (21%)]         Loss: 0.099001
Train Epoch: 5 [19200/60000 (32%)]         Loss: 0.143986
Train Epoch: 5 [25600/60000 (43%)]         Loss: 0.172746
Train Epoch: 5 [32000/60000 (53%)]         Loss: 0.277479
Train Epoch: 5 [38400/60000 (64%)]         Loss: 0.230550
Train Epoch: 5 [44800/60000 (75%)]         Loss: 0.179540
Train Epoch: 5 [51200/60000 (85%)]         Loss: 0.131457
Train Epoch: 5 [57600/60000 (96%)]         Loss: 0.210643

Test set: Average loss: 0.2292, Accuracy: 9343/10000 (93%)

Train Epoch: 6 [0/60000 (0%)]        Loss: 0.152416
Train Epoch: 6 [6400/60000 (11%)]         Loss: 0.132874
Train Epoch: 6 [12800/60000 (21%)]         Loss: 0.150181
Train Epoch: 6 [19200/60000 (32%)]         Loss: 0.122614
Train Epoch: 6 [25600/60000 (43%)]         Loss: 0.191909
Train Epoch: 6 [32000/60000 (53%)]         Loss: 0.221607
Train Epoch: 6 [38400/60000 (64%)]         Loss: 0.286850
Train Epoch: 6 [44800/60000 (75%)]         Loss: 0.199715
Train Epoch: 6 [51200/60000 (85%)]         Loss: 0.143983
Train Epoch: 6 [57600/60000 (96%)]         Loss: 0.222106

Test set: Average loss: 0.2131, Accuracy: 9389/10000 (94%)

Train Epoch: 7 [0/60000 (0%)]        Loss: 0.156749
Train Epoch: 7 [6400/60000 (11%)]         Loss: 0.149103
Train Epoch: 7 [12800/60000 (21%)]         Loss: 0.147713
Train Epoch: 7 [19200/60000 (32%)]         Loss: 0.236170
Train Epoch: 7 [25600/60000 (43%)]         Loss: 0.258319
```

```
Train Epoch: 7 [32000/60000 (53%)]          Loss: 0.206529
Train Epoch: 7 [38400/60000 (64%)]          Loss: 0.212687
Train Epoch: 7 [44800/60000 (75%)]          Loss: 0.198772
Train Epoch: 7 [51200/60000 (85%)]          Loss: 0.257168
Train Epoch: 7 [57600/60000 (96%)]          Loss: 0.145119

Test set: Average loss: 0.1977, Accuracy: 9435/10000 (94%)

Train Epoch: 8 [0/60000 (0%)]        Loss: 0.169365
Train Epoch: 8 [6400/60000 (11%)]          Loss: 0.118794
Train Epoch: 8 [12800/60000 (21%)]          Loss: 0.168974
Train Epoch: 8 [19200/60000 (32%)]          Loss: 0.087987
Train Epoch: 8 [25600/60000 (43%)]          Loss: 0.124883
Train Epoch: 8 [32000/60000 (53%)]          Loss: 0.138366
Train Epoch: 8 [38400/60000 (64%)]          Loss: 0.057039
Train Epoch: 8 [44800/60000 (75%)]          Loss: 0.099621
Train Epoch: 8 [51200/60000 (85%)]          Loss: 0.055291
Train Epoch: 8 [57600/60000 (96%)]          Loss: 0.089533

Test set: Average loss: 0.1876, Accuracy: 9466/10000 (95%)

Train Epoch: 9 [0/60000 (0%)]        Loss: 0.124728
Train Epoch: 9 [6400/60000 (11%)]          Loss: 0.228127
Train Epoch: 9 [12800/60000 (21%)]          Loss: 0.278833
Train Epoch: 9 [19200/60000 (32%)]          Loss: 0.133893
Train Epoch: 9 [25600/60000 (43%)]          Loss: 0.203673
Train Epoch: 9 [32000/60000 (53%)]          Loss: 0.151205
Train Epoch: 9 [38400/60000 (64%)]          Loss: 0.233931
Train Epoch: 9 [44800/60000 (75%)]          Loss: 0.270383
Train Epoch: 9 [51200/60000 (85%)]          Loss: 0.123984
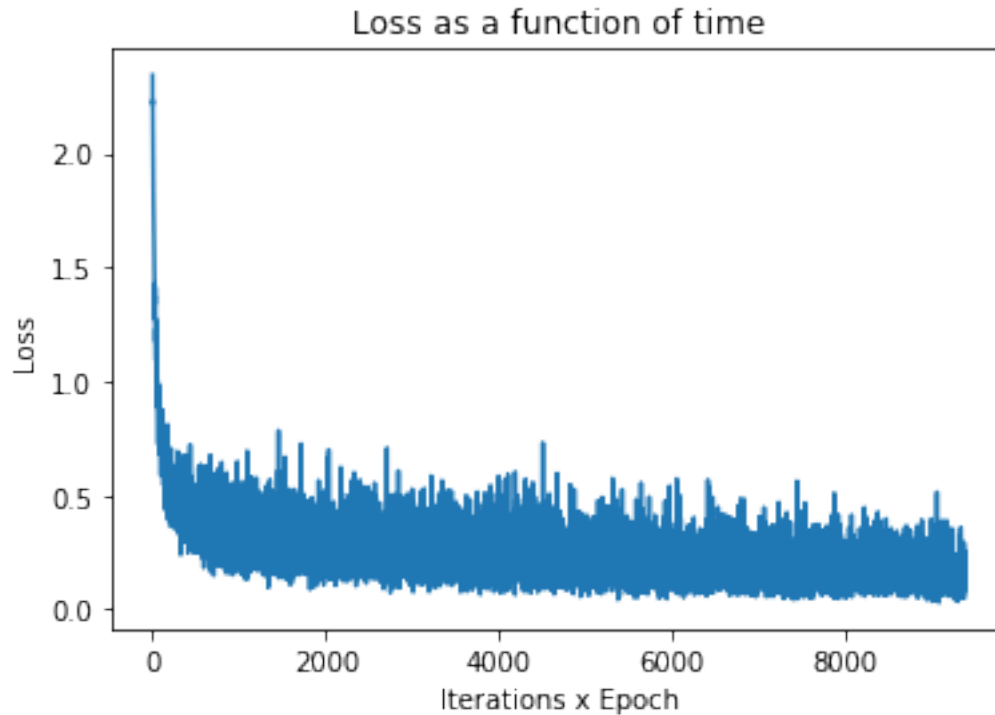Train Epoch: 9 [57600/60000 (96%)]          Loss: 0.243015

Test set: Average loss: 0.1731, Accuracy: 9502/10000 (95%)

Train Epoch: 10 [0/60000 (0%)]        Loss: 0.107806
Train Epoch: 10 [6400/60000 (11%)]          Loss: 0.122718
Train Epoch: 10 [12800/60000 (21%)]          Loss: 0.060518
Train Epoch: 10 [19200/60000 (32%)]          Loss: 0.131182
Train Epoch: 10 [25600/60000 (43%)]          Loss: 0.051315
Train Epoch: 10 [32000/60000 (53%)]          Loss: 0.076039
Train Epoch: 10 [38400/60000 (64%)]          Loss: 0.221091
Train Epoch: 10 [44800/60000 (75%)]          Loss: 0.085397
Train Epoch: 10 [51200/60000 (85%)]          Loss: 0.206541
Train Epoch: 10 [57600/60000 (96%)]          Loss: 0.190997

Test set: Average loss: 0.1630, Accuracy: 9533/10000 (95%)
```

```
In [17]: plt.plot(loss_g)
         plt.xlabel("Iterations x Epoch")
         plt.ylabel("Loss")
         plt.title("Loss as a function of time")
```

Out[17]: <matplotlib.text.Text at 0x7faa146885f8>

- Now set the learning rate to 10 and observe what happens during training. Save the output
  in your report and give a brief explanation

```
In [18]: train_loader = torch.utils.data.DataLoader(trainset, batch_size=batch_size, shuffle=Tru
         test_loader  = torch.utils.data.DataLoader(testset, batch_size=batch_size, shuffle=Fals

         class Net(nn.Module):
             def __init__(self, num_inputs, num_outputs, hidden_units):
                 super(Net, self).__init__()
                 self.fc1 = nn.Linear(num_inputs, hidden_units)
                 self.fc2 = nn.Linear(hidden_units, num_outputs)

             def forward(self, x):
                 x = x.view(-1, num_inputs) # reshape input to batch x num_inputs
                 x = self.fc1(x)
                 x = F.tanh(x)
                 x = self.fc2(x)
```

17

```
            return x

    network = Net(num_inputs, num_outputs, hidden_units=1000)
    network.cuda()
    optimizer = optim.SGD(network.parameters(), lr=10)

In [19]: loss_g = []
    for epoch in range(1, 1 + epochs):
        loss_g.extend(train(epoch))
        test()
```

```
Train Epoch: 1 [0/60000 (0%)]        Loss: 2.264647
Train Epoch: 1 [6400/60000 (11%)]      Loss: 1552.625854
Train Epoch: 1 [12800/60000 (21%)]      Loss: 429.011383
Train Epoch: 1 [19200/60000 (32%)]      Loss: 653.251892
Train Epoch: 1 [25600/60000 (43%)]      Loss: 750.171326
Train Epoch: 1 [32000/60000 (53%)]      Loss: 303.090851
Train Epoch: 1 [38400/60000 (64%)]      Loss: 251.620911
Train Epoch: 1 [44800/60000 (75%)]      Loss: 356.836029
Train Epoch: 1 [51200/60000 (85%)]      Loss: 377.940063
Train Epoch: 1 [57600/60000 (96%)]      Loss: 365.819000


Test set: Average loss: 363.3634, Accuracy: 5395/10000 (54%)


Train Epoch: 2 [0/60000 (0%)]        Loss: 300.932678
Train Epoch: 2 [6400/60000 (11%)]      Loss: 564.763000
Train Epoch: 2 [12800/60000 (21%)]      Loss: 241.000412
Train Epoch: 2 [19200/60000 (32%)]      Loss: 152.962906
Train Epoch: 2 [25600/60000 (43%)]      Loss: 173.837845
Train Epoch: 2 [32000/60000 (53%)]      Loss: 98.620087
Train Epoch: 2 [38400/60000 (64%)]      Loss: 228.648590
Train Epoch: 2 [44800/60000 (75%)]      Loss: 425.462982
Train Epoch: 2 [51200/60000 (85%)]      Loss: 582.802124
Train Epoch: 2 [57600/60000 (96%)]      Loss: 333.972961


Test set: Average loss: 386.5838, Accuracy: 5284/10000 (53%)


Train Epoch: 3 [0/60000 (0%)]        Loss: 438.458282
Train Epoch: 3 [6400/60000 (11%)]      Loss: 136.504471
Train Epoch: 3 [12800/60000 (21%)]      Loss: 288.777313
Train Epoch: 3 [19200/60000 (32%)]      Loss: 317.421295
Train Epoch: 3 [25600/60000 (43%)]      Loss: 454.274750
Train Epoch: 3 [32000/60000 (53%)]      Loss: 526.408508
Train Epoch: 3 [38400/60000 (64%)]      Loss: 241.444458
Train Epoch: 3 [44800/60000 (75%)]      Loss: 137.116394
Train Epoch: 3 [51200/60000 (85%)]      Loss: 427.970093
Train Epoch: 3 [57600/60000 (96%)]      Loss: 285.600677
```

Test set: Average loss: 355.9410, Accuracy: 5020/10000 (50%)

Train Epoch: 4 [0/60000 (0%)]   Loss: 297.969696
Train Epoch: 4 [6400/60000 (11%)]   Loss: 323.365295
Train Epoch: 4 [12800/60000 (21%)]   Loss: 308.344421
Train Epoch: 4 [19200/60000 (32%)]   Loss: 364.809753
Train Epoch: 4 [25600/60000 (43%)]   Loss: 100.907837
Train Epoch: 4 [32000/60000 (53%)]   Loss: 94.421906
Train Epoch: 4 [38400/60000 (64%)]   Loss: 115.261230
Train Epoch: 4 [44800/60000 (75%)]   Loss: 297.078430
Train Epoch: 4 [51200/60000 (85%)]   Loss: 417.447998
Train Epoch: 4 [57600/60000 (96%)]   Loss: 134.018051

Test set: Average loss: 398.5276, Accuracy: 5355/10000 (54%)

Train Epoch: 5 [0/60000 (0%)]   Loss: 373.431091
Train Epoch: 5 [6400/60000 (11%)]   Loss: 216.246796
Train Epoch: 5 [12800/60000 (21%)]   Loss: 158.145477
Train Epoch: 5 [19200/60000 (32%)]   Loss: 390.546051
Train Epoch: 5 [25600/60000 (43%)]   Loss: 131.778931
Train Epoch: 5 [32000/60000 (53%)]   Loss: 264.217712
Train Epoch: 5 [38400/60000 (64%)]   Loss: 311.592957
Train Epoch: 5 [44800/60000 (75%)]   Loss: 116.125450
Train Epoch: 5 [51200/60000 (85%)]   Loss: 583.161194
Train Epoch: 5 [57600/60000 (96%)]   Loss: 135.884491

Test set: Average loss: 154.0317, Accuracy: 7044/10000 (70%)

Train Epoch: 6 [0/60000 (0%)]   Loss: 239.725189
Train Epoch: 6 [6400/60000 (11%)]   Loss: 378.847046
Train Epoch: 6 [12800/60000 (21%)]   Loss: 264.241699
Train Epoch: 6 [19200/60000 (32%)]   Loss: 223.253784
Train Epoch: 6 [25600/60000 (43%)]   Loss: 160.195160
Train Epoch: 6 [32000/60000 (53%)]   Loss: 273.797913
Train Epoch: 6 [38400/60000 (64%)]   Loss: 237.564438
Train Epoch: 6 [44800/60000 (75%)]   Loss: 103.915703
Train Epoch: 6 [51200/60000 (85%)]   Loss: 144.754944
Train Epoch: 6 [57600/60000 (96%)]   Loss: 110.838966

Test set: Average loss: 520.0282, Accuracy: 5722/10000 (57%)

Train Epoch: 7 [0/60000 (0%)]   Loss: 397.853577
Train Epoch: 7 [6400/60000 (11%)]   Loss: 141.418396
Train Epoch: 7 [12800/60000 (21%)]   Loss: 76.453957
Train Epoch: 7 [19200/60000 (32%)]   Loss: 117.712090
Train Epoch: 7 [25600/60000 (43%)]   Loss: 182.777176
Train Epoch: 7 [32000/60000 (53%)]   Loss: 128.429596
Train Epoch: 7 [38400/60000 (64%)]   Loss: 166.386932

```
Train Epoch: 7 [44800/60000 (75%)]        Loss: 214.965927
Train Epoch: 7 [51200/60000 (85%)]        Loss: 226.164246
Train Epoch: 7 [57600/60000 (96%)]        Loss: 304.011200


Test set: Average loss: 153.6968, Accuracy: 7046/10000 (70%)


Train Epoch: 8 [0/60000 (0%)]        Loss: 178.044739
Train Epoch: 8 [6400/60000 (11%)]        Loss: 206.066025
Train Epoch: 8 [12800/60000 (21%)]        Loss: 116.606476
Train Epoch: 8 [19200/60000 (32%)]        Loss: 315.435089
Train Epoch: 8 [25600/60000 (43%)]        Loss: 174.898361
Train Epoch: 8 [32000/60000 (53%)]        Loss: 131.582291
Train Epoch: 8 [38400/60000 (64%)]        Loss: 317.067993
Train Epoch: 8 [44800/60000 (75%)]        Loss: 201.001022
Train Epoch: 8 [51200/60000 (85%)]        Loss: 172.404221
Train Epoch: 8 [57600/60000 (96%)]        Loss: 48.704540


Test set: Average loss: 260.7458, Accuracy: 6494/10000 (65%)


Train Epoch: 9 [0/60000 (0%)]        Loss: 264.489136
Train Epoch: 9 [6400/60000 (11%)]        Loss: 102.425941
Train Epoch: 9 [12800/60000 (21%)]        Loss: 353.628601
Train Epoch: 9 [19200/60000 (32%)]        Loss: 308.383850
Train Epoch: 9 [25600/60000 (43%)]        Loss: 336.653168
Train Epoch: 9 [32000/60000 (53%)]        Loss: 436.231415
Train Epoch: 9 [38400/60000 (64%)]        Loss: 139.269958
Train Epoch: 9 [44800/60000 (75%)]        Loss: 164.501450
Train Epoch: 9 [51200/60000 (85%)]        Loss: 98.646729
Train Epoch: 9 [57600/60000 (96%)]        Loss: 91.241409


Test set: Average loss: 293.0139, Accuracy: 6489/10000 (65%)


Train Epoch: 10 [0/60000 (0%)]        Loss: 437.896942
Train Epoch: 10 [6400/60000 (11%)]        Loss: 101.765732
Train Epoch: 10 [12800/60000 (21%)]        Loss: 371.181915
Train Epoch: 10 [19200/60000 (32%)]        Loss: 88.069504
Train Epoch: 10 [25600/60000 (43%)]        Loss: 282.099945
Train Epoch: 10 [32000/60000 (53%)]        Loss: 73.224792
Train Epoch: 10 [38400/60000 (64%)]        Loss: 356.815826
Train Epoch: 10 [44800/60000 (75%)]        Loss: 133.444214
Train Epoch: 10 [51200/60000 (85%)]        Loss: 145.617142
Train Epoch: 10 [57600/60000 (96%)]        Loss: 107.395584


Test set: Average loss: 286.4534, Accuracy: 6393/10000 (64%)



In [20]: plt.plot(loss_g)
```