```
import torch
import torchvision
import torchvision.transforms as transforms


mnist_train_set = torchvision.datasets.FashionMNIST(
    root = './data',
    train = True,
    download = True,
    transform=transforms.Compose([
        transforms.ToTensor()
    ])
)


from torch.utils.data import Dataset
import torch.nn.functional as F


class MnistWithRandomNumberDataset(Dataset):
  def __init__(self, mnist_data, random_nums):
    self.mnist_data = mnist_data
    self.random_nums = random_nums

  def __len__(self):
    return len(self.mnist_data)

  def __getitem__(self, loc):
    img, label = self.mnist_data[loc]
    random_num = self.random_nums[loc]
    sum = label + random_num
    return img, label, random_num, sum


from torch.utils.data import DataLoader
import random
random.seed(23)

# Load MNIST data and random numbers
random_nums = [random.randint(0, 9) for i in range(len(mnist_train_set))]
dataset = MnistWithRandomNumberDataset(mnist_train_set, random_nums)
train_loader = DataLoader(dataset, batch_size = 64, shuffle = True)


import torch.optim as optim
torch.set_grad_enabled(True)
```

```
<torch.autograd.grad_mode.set_grad_enabled at 0x7f626c6cca90>
```

```
def get_num_correct(preds, labels):
  return preds.argmax(dim=1).eq(labels).sum().item()


# Get the first batch
batch = next(iter(train_loader))

# Extract the data and label
images, true_labels, random_nums, true_sums = batch


import torch
import torch.nn as nn

class Network(nn.Module):
  def __init__(self):
    super(Network, self).__init__()

    # Convolutional layers to process the image
    self.conv1 = nn.Conv2d(1, 32, kernel_size=3)
    self.conv2 = nn.Conv2d(32, 64, kernel_size=3)
    self.conv3 = nn.Conv2d(64, 128, kernel_size=3)

    # Fully connected layers to process the image
    self.fc1 = nn.Linear(in_features=128, out_features=120)
    self.fc2 = nn.Linear(in_features=120, out_features=60)
    self.out = nn.Linear(in_features=60, out_features=10)

  def forward(self, x, r):
    #first conv layer
    x = self.conv1(x)
```

```
        x = F.relu(x)
        x = F.max_pool2d(x, kernel_size=2, stride=2)

        #second conv layer
        x = self.conv2(x)
        x = F.relu(x)
        x = F.max_pool2d(x, kernel_size=2, stride=2)

        #third conv layer
        x = self.conv3(x)
        x = F.relu(x)
        x = F.max_pool2d(x, kernel_size=2, stride=2)

        # first fully connected layer
        x = x.view(-1, 128)
        x = self.fc1(x)
        x = F.relu(x)

        # second fully connected layer
        x = self.fc2(x)
        x = F.relu(x)
        x = self.out(x)

        #here we are combining the predicted image label and the the random number using one hot encoding
        max_indices_x = torch.argmax(x, dim=1)
        summed_indices = max_indices_x + r
        sum = F.one_hot(summed_indices, num_classes=19)
        sum = sum.to(dtype=torch.float32)
        sum.requires_grad_()
        return x, sum
```

building network for single batch

```
network = Network()

train_loader = DataLoader(dataset, batch_size = 64, shuffle = True)
optimizer = optim.Adam(network.parameters(), lr=0.01)

batch = next(iter(train_loader)) # Get Batch
images, labels, random_nums, sums = batch

label_preds, sum_preds = network(images, random_nums) # Pass Batch
label_loss = F.cross_entropy(label_preds, labels)
sum_loss = F.cross_entropy(sum_preds, sums)
loss = 0.5 * (label_loss + sum_loss) # Calculate Loss
print('loss1:', loss.item())
print('correct1:', get_num_correct(label_preds, labels))
optimizer.zero_grad()
label_loss.backward()·#·Calculate·Gradients
optimizer.step()·#·Update·Weights


label_preds, sum_preds = network(images, random_nums) # Pass Batch
label_loss = F.cross_entropy(label_preds, labels)
sum_loss = F.cross_entropy(sum_preds, sums)
loss = 0.5 * (label_loss + sum_loss) # Calculate Loss
print('loss2:', loss.item())
print('correct2:', get_num_correct(label_preds, labels))
```

```
loss1: 2.598670244216919
correct1: 9
loss2: 2.5742971897125244
correct2: 9
```

Doing for multiple epochs and batches

```
train_loader = DataLoader(dataset, batch_size = 64, shuffle = True)
optimizer = optim.Adam(network.parameters(), lr=0.01)

for epoch in range(20):

    total_loss = 0
    total_correct_label = 0
    total_loss_label = 0
```

```
total_correct_sum = 0
total_loss_sum = 0

for batch in train_loader: # Get Batch
    images, labels, random_nums, sums = batch

    label_preds, sum_preds = network(images, random_nums) # Pass Batch
    label_loss = F.cross_entropy(label_preds, labels)
    sum_loss = F.cross_entropy(sum_preds, sums)
    loss = 0.5 * (label_loss + sum_loss) # Calculate Loss

    optimizer.zero_grad()
    loss.backward() # Calculate Gradients
    optimizer.step() # Update Weights

    total_loss += loss.item()
    total_loss_label += label_loss.item()
    total_loss_sum += sum_loss.item()
    total_correct_label += get_num_correct(label_preds, labels)
    total_correct_sum += get_num_correct(sum_preds, sums)

print(
    "epoch", epoch,
    "total_correct_label:", total_correct_label,
    "total_loss_label:", total_loss_label,
    "total_correct_sum:", total_correct_sum,
    "total_loss_sum:", total_loss_sum,
    "loss:", total_loss
)
```

```
epoch 0 total_correct_label: 51314 total_loss_label: 379.30297972261906 total_correct_sum: 51314 total_loss_sum: 2040
epoch 1 total_correct_label: 51788 total_loss_label: 353.96588522940874 total_correct_sum: 51788 total_loss_sum: 2033
epoch 2 total_correct_label: 52151 total_loss_label: 344.57045044004917 total_correct_sum: 52151 total_loss_sum: 2027
epoch 3 total_correct_label: 52180 total_loss_label: 337.7876736074686 total_correct_sum: 52180 total_loss_sum: 2027.
epoch 4 total_correct_label: 52216 total_loss_label: 339.0338530316949 total_correct_sum: 52216 total_loss_sum: 2026.
epoch 5 total_correct_label: 52552 total_loss_label: 324.54372161626816 total_correct_sum: 52552 total_loss_sum: 2021
epoch 6 total_correct_label: 52591 total_loss_label: 326.89600083976984 total_correct_sum: 52591 total_loss_sum: 2021
epoch 7 total_correct_label: 52393 total_loss_label: 335.1514900177717 total_correct_sum: 52393 total_loss_sum: 2024.
epoch 8 total_correct_label: 52737 total_loss_label: 322.6487879753113 total_correct_sum: 52737 total_loss_sum: 2018.
epoch 9 total_correct_label: 52687 total_loss_label: 324.9943139180541 total_correct_sum: 52687 total_loss_sum: 2019.
epoch 10 total_correct_label: 53033 total_loss_label: 305.8706995919347 total_correct_sum: 53033 total_loss_sum: 2014
epoch 11 total_correct_label: 52697 total_loss_label: 323.9645846039057 total_correct_sum: 52697 total_loss_sum: 2019
epoch 12 total_correct_label: 52922 total_loss_label: 313.1049950271845 total_correct_sum: 52922 total_loss_sum: 2015
epoch 13 total_correct_label: 52697 total_loss_label: 331.4623031914234 total_correct_sum: 52697 total_loss_sum: 2019
epoch 14 total_correct_label: 52957 total_loss_label: 314.1307446360588 total_correct_sum: 52957 total_loss_sum: 2015
epoch 15 total_correct_label: 52426 total_loss_label: 347.6105978861451 total_correct_sum: 52426 total_loss_sum: 2023
epoch 16 total_correct_label: 52954 total_loss_label: 322.413120046258 total_correct_sum: 52954 total_loss_sum: 2015.
epoch 17 total_correct_label: 53091 total_loss_label: 313.45527363568544 total_correct_sum: 53091 total_loss_sum: 201
epoch 18 total_correct_label: 53361 total_loss_label: 295.9490918368101 total_correct_sum: 53361 total_loss_sum: 2008
epoch 19 total_correct_label: 53092 total_loss_label: 313.57266400009394 total_correct_sum: 53092 total_loss_sum: 201
```