CNN

November 20, 2022

1 Import Libraries

[10]:
```
import torch, torchvision
from torch import nn
from torch import optim
from torchvision.transforms import ToTensor
import torch.nn.functional as F
import matplotlib.pyplot as plt
import copy
```

[11]:
```
numb_batch=64
```

2 Getting Data

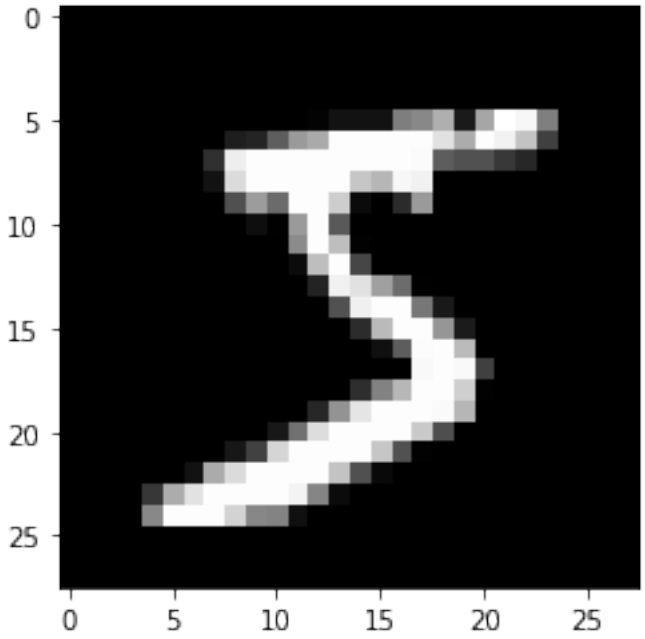
[12]:
```
T=torchvision.transforms.Compose([
    torchvision.transforms.ToTensor()
])

train_data=torchvision.datasets.
 ↪MNIST('mnist_data',train=True,download=True,transform=T);
val_data=torchvision.datasets.
 ↪MNIST('mnist_data',train=False,download=True,transform=T);

train_dl=torch.utils.data.DataLoader(train_data,batch_size=numb_batch)
val_dl=torch.utils.data.DataLoader(val_data,batch_size=numb_batch)
```

[13]:
```
plt.imshow(train_data[0][0][0],cmap="gray")
```

[13]:
```
<matplotlib.image.AxesImage at 0x7fbf75a862d0>
```



3 Create the Model

[14]:
```
def create_lenet():
    model=nn.Sequential(
        nn.Conv2d(1,6,5,padding=2),
        nn.ReLU(),
        nn.AvgPool2d(2,stride=2),

        nn.Conv2d(6,16,5,padding=0),
        nn.ReLU(),
        nn.AvgPool2d(2,stride=2),

        nn.Flatten(),
        nn.Linear(400,120),
        nn.Linear(120,84),
        nn.Linear(84,10)
    )
    return model
```

4 Validate the Model

[15]:
```
def validate(model,data):
    total=0
    correct=0
    for i,(images,labels) in enumerate(data):
        images=images
        x=model(images)
        value,pred=torch.max(x,1)
        pred=pred.data.cpu()
        total+=x.size(0)
        correct+=torch.sum(pred==labels)
    return correct*100./total
```

5 Training Function

[16]:
```
def train(numb_epoch=3, lr=1e-3, device="cpu"):
    accuracies = []
    cnn = create_lenet().to(device)
    cec = nn.CrossEntropyLoss()
    optimizer = optim.Adam(cnn.parameters(), lr=lr)
    max_accuracy = 0
    for epoch in range(numb_epoch):
        for i, (images, labels) in enumerate(train_dl):
            images = images.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()
            pred = cnn(images)
            loss = cec(pred, labels)
            loss.backward()
            optimizer.step()
        accuracy = float(validate(cnn, val_dl))
        accuracies.append(accuracy)
        if accuracy > max_accuracy:
            best_model = copy.deepcopy(cnn)
            max_accuracy = accuracy
            print("Saving Best Model with Accuracy: ", accuracy)
        print('Epoch:', epoch+1, "Accuracy :", accuracy, '%')
    plt.plot(accuracies)
    device=torch.device("cpu")
    return best_model
```

6 Training the model

[ ]:
```
device=torch.device("cpu")
lemet=train(40,device=device)
```

```
Saving Best Model with Accuracy:  96.72000122070312
Epoch: 1 Accuracy : 96.72000122070312 %
```

[ ]:

[ ]: