CS 795 Assignment 4 - SGD Optimizer

April 7, 2022

[1]:

```python
import torch
```

[2]:

```python
# Device configuration
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
device
```

[2]:

```
device(type='cuda')
```

[3]:

```python
from torchvision import datasets
from torchvision.transforms import ToTensor
train_data = datasets.MNIST(
    root = 'data',
    train = True,
    transform = ToTensor(),
    download = True,
)
test_data = datasets.MNIST(
    root = 'data',
    train = False,
    transform = ToTensor()
)
```

[4]:

```python
print(train_data)
```

```
Dataset MNIST
    Number of datapoints: 60000
    Root location: data
    Split: Train
    StandardTransform
Transform: ToTensor()
```

[5]:

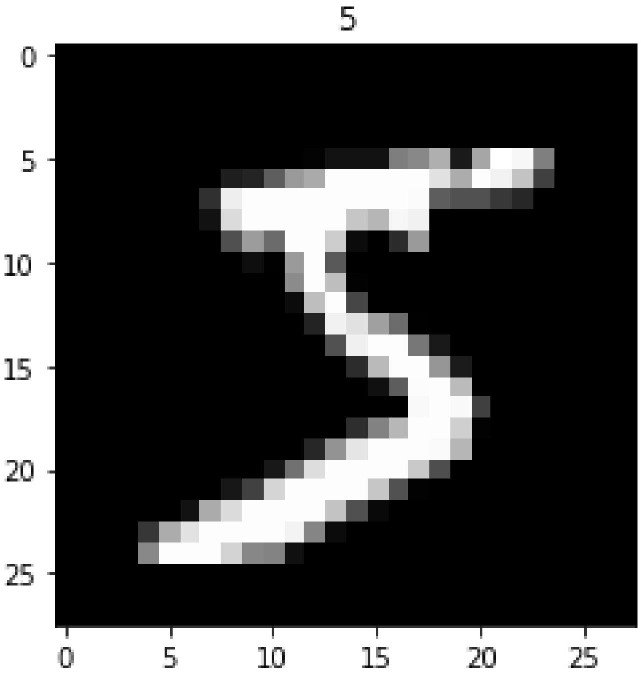
```python
print(test_data)
```

```
Dataset MNIST
    Number of datapoints: 10000
    Root location: data
    Split: Test
    StandardTransform
Transform: ToTensor()
```

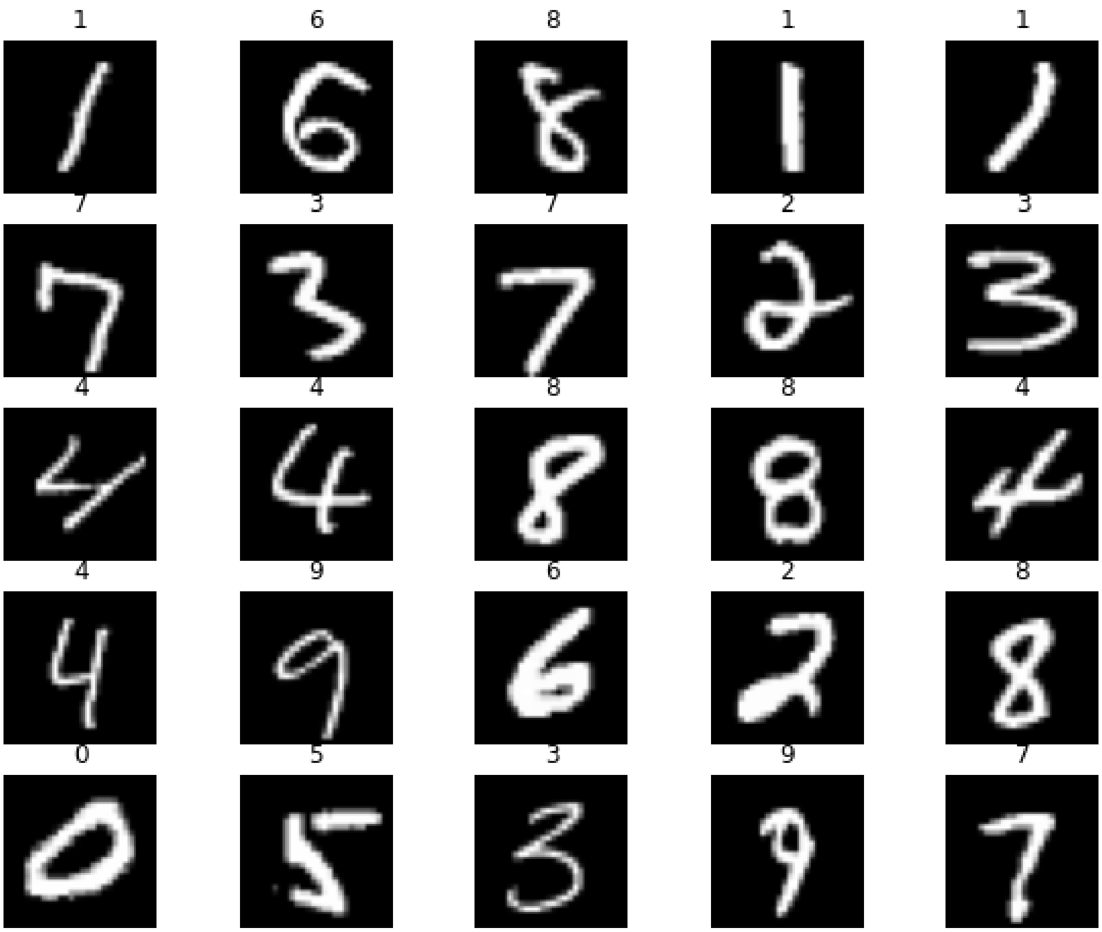
[6]:
```
print(train_data.data.size())
```

```
torch.Size([60000, 28, 28])
```

[7]:
```
import matplotlib.pyplot as plt
plt.imshow(train_data.data[0], cmap='gray')
plt.title('%i' % train_data.targets[0])
plt.show()
```



[8]:
```
figure = plt.figure(figsize=(10, 8))
cols, rows = 5, 5
for i in range(1, cols * rows + 1):
    sample_idx = torch.randint(len(train_data), size=(1,)).item()
    img, label = train_data[sample_idx]
    figure.add_subplot(rows, cols, i)
    plt.title(label)
    plt.axis("off")
    plt.imshow(img.squeeze(), cmap="gray")
plt.show()
```



[9]:

```
from torch.utils.data import DataLoader
loaders = {
    'train' : torch.utils.data.DataLoader(train_data,
                                          batch_size=100,
                                          shuffle=True,
                                          num_workers=1),

    'test'  : torch.utils.data.DataLoader(test_data,
                                          batch_size=100,
                                          shuffle=True,
                                          num_workers=1),
}
loaders
```

[9]:

```
{'train': <torch.utils.data.dataloader.DataLoader at 0x7fe893e55be0>,
 'test': <torch.utils.data.dataloader.DataLoader at 0x7fe893e55a30>}
```

[10]:

```
import torch.nn as nn
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels=1,
                out_channels=16,
                kernel_size=5,
                stride=1,
                padding=2,
            ),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2),
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(16, 32, 5, 1, 2),
            nn.ReLU(),
            nn.MaxPool2d(2),
        )
        # fully connected layer, output 10 classes
        self.out = nn.Linear(32 * 7 * 7, 10)
    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        # flatten the output of conv2 to (batch_size, 32 * 7 * 7)
        x = x.view(x.size(0), -1)
        output = self.out(x)
        return output, x    # return x for visualization
```

[11]:

```
cnn = CNN()
print(cnn)
```

```
CNN(
  (conv1): Sequential(
    (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv2): Sequential(
    (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (out): Linear(in_features=1568, out_features=10, bias=True)
)
```

[12]:

```
loss_func = nn.CrossEntropyLoss()
loss_func
```

[12]:

```
CrossEntropyLoss()
```

[13]:

```
import math
from torch.optim import Optimizer

class SGD(Optimizer):

    def __init__(self, params, lr=.01, momentum=0, dampening=0,
                 weight_decay=0, nesterov=False):
        defaults = dict(lr=lr, momentum=momentum, dampening=dampening,
                        weight_decay=weight_decay, nesterov=nesterov)
        super(SGD, self).__init__(params, defaults)

    def __setstate__(self, state):
        super(SGD, self).__setstate__(state)
        for group in self.param_groups:
            group.setdefault('nesterov', False)

    def step(self, closure=None):
        loss = None
        if closure is not None:
            loss = closure()

        for group in self.param_groups:
            weight_decay = group['weight_decay']
            momentum = group['momentum']
            dampening = group['dampening']
            nesterov = group['nesterov']

            for p in group['params']:
                if p.grad is None:
                    continue
                d_p = p.grad.data
                if weight_decay != 0:
                    d_p.add_(weight_decay, p.data)
                # Apply learning rate
                d_p.mul_(group['lr'])
                if momentum != 0:
                    param_state = self.state[p]
                    if 'momentum_buffer' not in param_state:
                        buf = param_state['momentum_buffer'] = torch.
↪zeros_like(p.data)
                        buf.mul_(momentum).add_(d_p)
                    else:
                        buf = param_state['momentum_buffer']
                        buf.mul_(momentum).add_(1 - dampening, d_p)
                    if nesterov:
                        d_p = d_p.add(momentum, buf)
                    else:
                        d_p = buf

                p.data.add_(-1, d_p)

        return loss
```

[14]:

```
from torch import optim
optimizer = SGD(cnn.parameters(), lr = 0.01)
optimizer
```

[14]:

```
SGD (
Parameter Group 0
    dampening: 0
    lr: 0.01
    momentum: 0
    nesterov: False
    weight_decay: 0
)
```

[15]:

```
from torch.autograd import Variable
num_epochs = 30
history = []
def test():
    # Test the model
    cnn.eval()
    with torch.no_grad():
        correct = 0
        total = 0
        for images, labels in loaders['test']:
            test_output, last_layer = cnn(images)
            b_y = Variable(labels)
            loss = loss_func(test_output, b_y)
            pred_y = torch.max(test_output, 1)[1].data.squeeze()
            accuracy = (pred_y == labels).sum().item() / float(labels.size(0))
            pass
        print('Test Accuracy of the model on the 10000 test images: %.2f' %␣
 ↪accuracy)
        print('Test Loss: {:.4f}', loss.item())

    return accuracy, loss.item()
```

```
def train(num_epochs, cnn, loaders):

    cnn.train()

    # Train the model
    total_step = len(loaders['train'])

    for epoch in range(num_epochs):
        for i, (images, labels) in enumerate(loaders['train']):

            # gives batch data, normalize x when iterate train_loader
            b_x = Variable(images)   # batch x
            b_y = Variable(labels)   # batch y
            output = cnn(b_x)[0]
            loss = loss_func(output, b_y)

            # clear gradients for this training step
            optimizer.zero_grad()

            # backpropagation, compute gradients
            loss.backward()
            # apply gradients
            optimizer.step()

            pred_y = torch.max(output, 1)[1].data.squeeze()
            accuracy = (pred_y == labels).sum().item() / float(labels.size(0))
            if (i+1) % 100 == 0:
                print ('Epoch [{}/{}], Step [{}/{}], Acc: {:.4f} Loss: {:.4f}'
                       .format(epoch + 1, num_epochs, i + 1, total_step,␣
 ↪accuracy, loss.item()))
                pass

        history.append([(accuracy, loss.item()), test()])
    pass
train(num_epochs, cnn, loaders)
```

```
/tmp/ipykernel_1141096/781053462.py:49: UserWarning: This overload of add_ is
deprecated:
        add_(Number alpha, Tensor other)
Consider using one of the following signatures instead:
        add_(Tensor other, *, Number alpha) (Triggered internally at
../torch/csrc/utils/python_arg_parser.cpp:1050.)
  p.data.add_(-1, d_p)
```

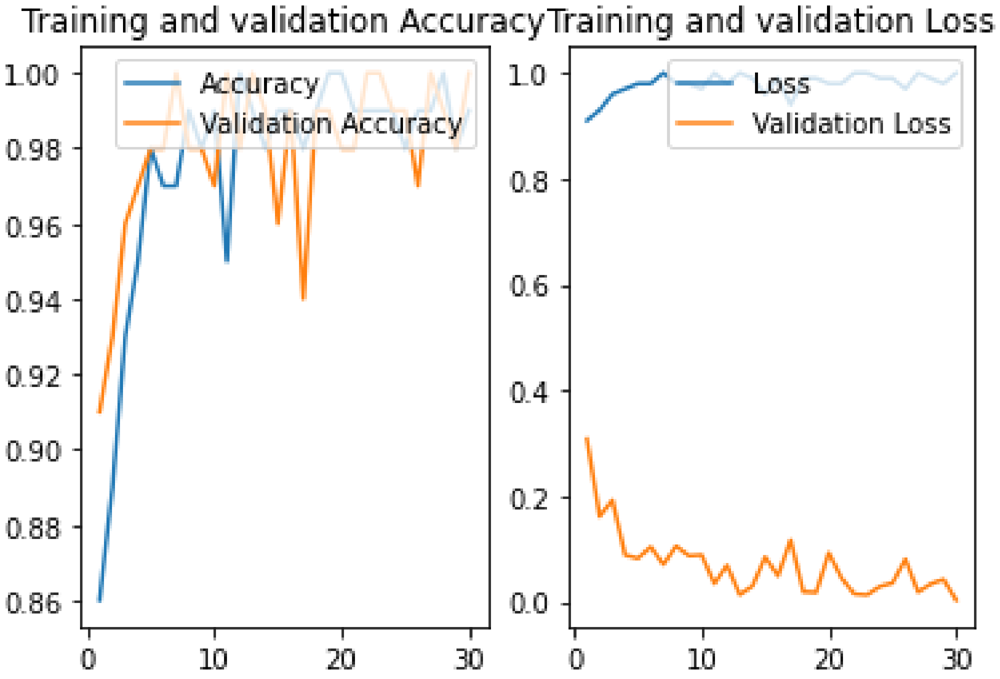
```
Epoch [1/30], Step [100/600], Acc: 0.5800 Loss: 2.1339
Epoch [1/30], Step [200/600], Acc: 0.7700 Loss: 1.1554
Epoch [1/30], Step [300/600], Acc: 0.8600 Loss: 0.5479
Epoch [1/30], Step [400/600], Acc: 0.8700 Loss: 0.4957
Epoch [1/30], Step [500/600], Acc: 0.9400 Loss: 0.2994
Epoch [1/30], Step [600/600], Acc: 0.8600 Loss: 0.3850
Test Accuracy of the model on the 10000 test images: 0.91
Test Loss: {:.4f} 0.3080783784389496
Epoch [2/30], Step [100/600], Acc: 0.9400 Loss: 0.1773
Epoch [2/30], Step [200/600], Acc: 0.9400 Loss: 0.3130
Epoch [2/30], Step [300/600], Acc: 0.9600 Loss: 0.1939
Epoch [2/30], Step [400/600], Acc: 0.8900 Loss: 0.3680
Epoch [2/30], Step [500/600], Acc: 0.9300 Loss: 0.2190
Epoch [2/30], Step [600/600], Acc: 0.8900 Loss: 0.2287
Test Accuracy of the model on the 10000 test images: 0.93
Test Loss: {:.4f} 0.161906898021698
Epoch [3/30], Step [100/600], Acc: 0.9400 Loss: 0.3019
Epoch [3/30], Step [200/600], Acc: 0.9300 Loss: 0.2088
Epoch [3/30], Step [300/600], Acc: 0.9400 Loss: 0.2661
Epoch [3/30], Step [400/600], Acc: 0.9000 Loss: 0.1892
Epoch [3/30], Step [500/600], Acc: 0.9900 Loss: 0.1028
Epoch [3/30], Step [600/600], Acc: 0.9300 Loss: 0.2188
Test Accuracy of the model on the 10000 test images: 0.96
Test Loss: {:.4f} 0.19273878633975983
Epoch [4/30], Step [100/600], Acc: 0.9600 Loss: 0.1795
Epoch [4/30], Step [200/600], Acc: 0.9400 Loss: 0.1431
Epoch [4/30], Step [300/600], Acc: 0.9600 Loss: 0.1107
Epoch [4/30], Step [400/600], Acc: 0.9800 Loss: 0.0863
Epoch [4/30], Step [500/600], Acc: 0.9600 Loss: 0.1264
Epoch [4/30], Step [600/600], Acc: 0.9500 Loss: 0.2360
Test Accuracy of the model on the 10000 test images: 0.97
Test Loss: {:.4f} 0.08885536342859268
Epoch [5/30], Step [100/600], Acc: 0.9500 Loss: 0.1313
Epoch [5/30], Step [200/600], Acc: 0.9500 Loss: 0.1281
Epoch [5/30], Step [300/600], Acc: 0.9800 Loss: 0.0794
Epoch [5/30], Step [400/600], Acc: 0.9600 Loss: 0.1421
Epoch [5/30], Step [500/600], Acc: 0.9600 Loss: 0.1243
Epoch [5/30], Step [600/600], Acc: 0.9800 Loss: 0.0856
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.08267923444509506
Epoch [6/30], Step [100/600], Acc: 0.9400 Loss: 0.1503
Epoch [6/30], Step [200/600], Acc: 0.9600 Loss: 0.0886
Epoch [6/30], Step [300/600], Acc: 0.9700 Loss: 0.0655
Epoch [6/30], Step [400/600], Acc: 0.9600 Loss: 0.1211
Epoch [6/30], Step [500/600], Acc: 0.9800 Loss: 0.1072
Epoch [6/30], Step [600/600], Acc: 0.9700 Loss: 0.1110
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.1043318659067154
Epoch [7/30], Step [100/600], Acc: 0.9800 Loss: 0.0562
Epoch [7/30], Step [200/600], Acc: 1.0000 Loss: 0.0411
Epoch [7/30], Step [300/600], Acc: 0.9800 Loss: 0.0708
Epoch [7/30], Step [400/600], Acc: 1.0000 Loss: 0.0351
Epoch [7/30], Step [500/600], Acc: 0.9700 Loss: 0.1597
Epoch [7/30], Step [600/600], Acc: 0.9700 Loss: 0.1258
Test Accuracy of the model on the 10000 test images: 1.00
Test Loss: {:.4f} 0.0709439367055893
Epoch [8/30], Step [100/600], Acc: 0.9700 Loss: 0.1121
Epoch [8/30], Step [200/600], Acc: 0.9800 Loss: 0.0769
Epoch [8/30], Step [300/600], Acc: 0.9800 Loss: 0.0508
Epoch [8/30], Step [400/600], Acc: 0.9800 Loss: 0.0929
Epoch [8/30], Step [500/600], Acc: 0.9500 Loss: 0.1171
Epoch [8/30], Step [600/600], Acc: 0.9900 Loss: 0.0429
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.10592630505561829
Epoch [9/30], Step [100/600], Acc: 0.9500 Loss: 0.1405
Epoch [9/30], Step [200/600], Acc: 0.9700 Loss: 0.1236
Epoch [9/30], Step [300/600], Acc: 0.9600 Loss: 0.2261
Epoch [9/30], Step [400/600], Acc: 0.9800 Loss: 0.0765
Epoch [9/30], Step [500/600], Acc: 0.9800 Loss: 0.0855
Epoch [9/30], Step [600/600], Acc: 0.9800 Loss: 0.0783
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.0874273031949997
Epoch [10/30], Step [100/600], Acc: 0.9900 Loss: 0.0460
Epoch [10/30], Step [200/600], Acc: 0.9900 Loss: 0.0421
Epoch [10/30], Step [300/600], Acc: 0.9900 Loss: 0.0559
Epoch [10/30], Step [400/600], Acc: 0.9700 Loss: 0.1195
Epoch [10/30], Step [500/600], Acc: 0.9900 Loss: 0.0373
Epoch [10/30], Step [600/600], Acc: 0.9900 Loss: 0.0278
Test Accuracy of the model on the 10000 test images: 0.97
Test Loss: {:.4f} 0.08896807581186295
Epoch [11/30], Step [100/600], Acc: 0.9900 Loss: 0.0279
Epoch [11/30], Step [200/600], Acc: 0.9800 Loss: 0.0478
Epoch [11/30], Step [300/600], Acc: 1.0000 Loss: 0.0285
Epoch [11/30], Step [400/600], Acc: 0.9600 Loss: 0.1064
Epoch [11/30], Step [500/600], Acc: 0.9700 Loss: 0.0929
Epoch [11/30], Step [600/600], Acc: 0.9500 Loss: 0.1744
Test Accuracy of the model on the 10000 test images: 1.00
Test Loss: {:.4f} 0.03506298363208771
Epoch [12/30], Step [100/600], Acc: 0.9800 Loss: 0.0555
Epoch [12/30], Step [200/600], Acc: 0.9800 Loss: 0.1073
Epoch [12/30], Step [300/600], Acc: 0.9800 Loss: 0.0927
Epoch [12/30], Step [400/600], Acc: 1.0000 Loss: 0.0141
Epoch [12/30], Step [500/600], Acc: 1.0000 Loss: 0.0198
Epoch [12/30], Step [600/600], Acc: 1.0000 Loss: 0.0127
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.06938844919204712
Epoch [13/30], Step [100/600], Acc: 0.9900 Loss: 0.0261
Epoch [13/30], Step [200/600], Acc: 0.9900 Loss: 0.0385
Epoch [13/30], Step [300/600], Acc: 1.0000 Loss: 0.0317
Epoch [13/30], Step [400/600], Acc: 0.9900 Loss: 0.0337
Epoch [13/30], Step [500/600], Acc: 0.9600 Loss: 0.0775
Epoch [13/30], Step [600/600], Acc: 0.9900 Loss: 0.0528
Test Accuracy of the model on the 10000 test images: 1.00
Test Loss: {:.4f} 0.0140267014503479
Epoch [14/30], Step [100/600], Acc: 1.0000 Loss: 0.0151
Epoch [14/30], Step [200/600], Acc: 0.9600 Loss: 0.0748
Epoch [14/30], Step [300/600], Acc: 0.9600 Loss: 0.1147
Epoch [14/30], Step [400/600], Acc: 1.0000 Loss: 0.0111
Epoch [14/30], Step [500/600], Acc: 0.9300 Loss: 0.1434
Epoch [14/30], Step [600/600], Acc: 0.9800 Loss: 0.0479
Test Accuracy of the model on the 10000 test images: 0.99
Test Loss: {:.4f} 0.030371250584721565
Epoch [15/30], Step [100/600], Acc: 0.9700 Loss: 0.0537
Epoch [15/30], Step [200/600], Acc: 0.9700 Loss: 0.1515
Epoch [15/30], Step [300/600], Acc: 0.9800 Loss: 0.0334
Epoch [15/30], Step [400/600], Acc: 0.9800 Loss: 0.0587
Epoch [15/30], Step [500/600], Acc: 0.9800 Loss: 0.0477
Epoch [15/30], Step [600/600], Acc: 0.9900 Loss: 0.0428
Test Accuracy of the model on the 10000 test images: 0.96
Test Loss: {:.4f} 0.08501268178224564
Epoch [16/30], Step [100/600], Acc: 0.9900 Loss: 0.0298
Epoch [16/30], Step [200/600], Acc: 0.9800 Loss: 0.0458
Epoch [16/30], Step [300/600], Acc: 0.9800 Loss: 0.0404
Epoch [16/30], Step [400/600], Acc: 1.0000 Loss: 0.0174
Epoch [16/30], Step [500/600], Acc: 0.9900 Loss: 0.0244
Epoch [16/30], Step [600/600], Acc: 0.9900 Loss: 0.0486
Test Accuracy of the model on the 10000 test images: 0.99
Test Loss: {:.4f} 0.049576807767152786
Epoch [17/30], Step [100/600], Acc: 0.9700 Loss: 0.0515
Epoch [17/30], Step [200/600], Acc: 0.9800 Loss: 0.0764
Epoch [17/30], Step [300/600], Acc: 1.0000 Loss: 0.0134
Epoch [17/30], Step [400/600], Acc: 0.9900 Loss: 0.0307
Epoch [17/30], Step [500/600], Acc: 1.0000 Loss: 0.0172
Epoch [17/30], Step [600/600], Acc: 0.9800 Loss: 0.0804
Test Accuracy of the model on the 10000 test images: 0.94
Test Loss: {:.4f} 0.11730123311281204
Epoch [18/30], Step [100/600], Acc: 0.9700 Loss: 0.0901
Epoch [18/30], Step [200/600], Acc: 0.9900 Loss: 0.0448
Epoch [18/30], Step [300/600], Acc: 1.0000 Loss: 0.0284
Epoch [18/30], Step [400/600], Acc: 0.9800 Loss: 0.0834
Epoch [18/30], Step [500/600], Acc: 0.9700 Loss: 0.0990
Epoch [18/30], Step [600/600], Acc: 0.9900 Loss: 0.0504
Test Accuracy of the model on the 10000 test images: 0.99
Test Loss: {:.4f} 0.019461361691355705
Epoch [19/30], Step [100/600], Acc: 1.0000 Loss: 0.0160
Epoch [19/30], Step [200/600], Acc: 0.9900 Loss: 0.0474
Epoch [19/30], Step [300/600], Acc: 0.9800 Loss: 0.0542
Epoch [19/30], Step [400/600], Acc: 0.9900 Loss: 0.0368
Epoch [19/30], Step [500/600], Acc: 0.9900 Loss: 0.0244
Epoch [19/30], Step [600/600], Acc: 1.0000 Loss: 0.0203
Test Accuracy of the model on the 10000 test images: 0.99
Test Loss: {:.4f} 0.018195874989032745
Epoch [20/30], Step [100/600], Acc: 0.9800 Loss: 0.0245
Epoch [20/30], Step [200/600], Acc: 0.9900 Loss: 0.0416
Epoch [20/30], Step [300/600], Acc: 0.9900 Loss: 0.0235
Epoch [20/30], Step [400/600], Acc: 0.9900 Loss: 0.0601
Epoch [20/30], Step [500/600], Acc: 0.9800 Loss: 0.0509
Epoch [20/30], Step [600/600], Acc: 1.0000 Loss: 0.0235
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.09288153797388077
Epoch [21/30], Step [100/600], Acc: 0.9800 Loss: 0.0360
Epoch [21/30], Step [200/600], Acc: 0.9800 Loss: 0.0870
Epoch [21/30], Step [300/600], Acc: 0.9900 Loss: 0.0426
Epoch [21/30], Step [400/600], Acc: 0.9800 Loss: 0.0964
Epoch [21/30], Step [500/600], Acc: 0.9700 Loss: 0.0587
Epoch [21/30], Step [600/600], Acc: 0.9900 Loss: 0.0441
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.04535512626171112
Epoch [22/30], Step [100/600], Acc: 1.0000 Loss: 0.0187
Epoch [22/30], Step [200/600], Acc: 0.9800 Loss: 0.0501
Epoch [22/30], Step [300/600], Acc: 0.9900 Loss: 0.0309
Epoch [22/30], Step [400/600], Acc: 0.9700 Loss: 0.0640
Epoch [22/30], Step [500/600], Acc: 0.9900 Loss: 0.0204
Epoch [22/30], Step [600/600], Acc: 0.9900 Loss: 0.0705
Test Accuracy of the model on the 10000 test images: 1.00
Test Loss: {:.4f} 0.015459936112165451
Epoch [23/30], Step [100/600], Acc: 1.0000 Loss: 0.0152
Epoch [23/30], Step [200/600], Acc: 1.0000 Loss: 0.0220
Epoch [23/30], Step [300/600], Acc: 0.9800 Loss: 0.0498
Epoch [23/30], Step [400/600], Acc: 0.9900 Loss: 0.0283
Epoch [23/30], Step [500/600], Acc: 0.9900 Loss: 0.0305
Epoch [23/30], Step [600/600], Acc: 0.9900 Loss: 0.0476
Test Accuracy of the model on the 10000 test images: 1.00
Test Loss: {:.4f} 0.013581983745098114
Epoch [24/30], Step [100/600], Acc: 0.9700 Loss: 0.0452
Epoch [24/30], Step [200/600], Acc: 1.0000 Loss: 0.0055
Epoch [24/30], Step [300/600], Acc: 0.9800 Loss: 0.0543
Epoch [24/30], Step [400/600], Acc: 0.9800 Loss: 0.0790
Epoch [24/30], Step [500/600], Acc: 1.0000 Loss: 0.0105
Epoch [24/30], Step [600/600], Acc: 0.9900 Loss: 0.0388
Test Accuracy of the model on the 10000 test images: 0.99
Test Loss: {:.4f} 0.029347749426960945
Epoch [25/30], Step [100/600], Acc: 0.9800 Loss: 0.0335
Epoch [25/30], Step [200/600], Acc: 0.9800 Loss: 0.0671
Epoch [25/30], Step [300/600], Acc: 0.9900 Loss: 0.0293
Epoch [25/30], Step [400/600], Acc: 1.0000 Loss: 0.0251
Epoch [25/30], Step [500/600], Acc: 0.9600 Loss: 0.0648
Epoch [25/30], Step [600/600], Acc: 0.9800 Loss: 0.0426
Test Accuracy of the model on the 10000 test images: 0.99
Test Loss: {:.4f} 0.03627597913146019
Epoch [26/30], Step [100/600], Acc: 0.9900 Loss: 0.0279
Epoch [26/30], Step [200/600], Acc: 0.9900 Loss: 0.0336
Epoch [26/30], Step [300/600], Acc: 0.9900 Loss: 0.0245
Epoch [26/30], Step [400/600], Acc: 0.9900 Loss: 0.0485
Epoch [26/30], Step [500/600], Acc: 1.0000 Loss: 0.0211
Epoch [26/30], Step [600/600], Acc: 0.9900 Loss: 0.0292
Test Accuracy of the model on the 10000 test images: 0.97
Test Loss: {:.4f} 0.08091029524803162
Epoch [27/30], Step [100/600], Acc: 0.9800 Loss: 0.0677
Epoch [27/30], Step [200/600], Acc: 1.0000 Loss: 0.0204
Epoch [27/30], Step [300/600], Acc: 1.0000 Loss: 0.0209
Epoch [27/30], Step [400/600], Acc: 1.0000 Loss: 0.0152
Epoch [27/30], Step [500/600], Acc: 0.9900 Loss: 0.0237
Epoch [27/30], Step [600/600], Acc: 0.9900 Loss: 0.0202
Test Accuracy of the model on the 10000 test images: 1.00
Test Loss: {:.4f} 0.0191293153911829
Epoch [28/30], Step [100/600], Acc: 0.9900 Loss: 0.0464
Epoch [28/30], Step [200/600], Acc: 0.9800 Loss: 0.1030
Epoch [28/30], Step [300/600], Acc: 1.0000 Loss: 0.0164
Epoch [28/30], Step [400/600], Acc: 1.0000 Loss: 0.0022
Epoch [28/30], Step [500/600], Acc: 0.9900 Loss: 0.0158
Epoch [28/30], Step [600/600], Acc: 1.0000 Loss: 0.0365
Test Accuracy of the model on the 10000 test images: 0.99
Test Loss: {:.4f} 0.03427070006728172
Epoch [29/30], Step [100/600], Acc: 1.0000 Loss: 0.0092
Epoch [29/30], Step [200/600], Acc: 0.9600 Loss: 0.0785
Epoch [29/30], Step [300/600], Acc: 0.9900 Loss: 0.0418
Epoch [29/30], Step [400/600], Acc: 1.0000 Loss: 0.0149
Epoch [29/30], Step [500/600], Acc: 0.9900 Loss: 0.0453
Epoch [29/30], Step [600/600], Acc: 0.9800 Loss: 0.0504
Test Accuracy of the model on the 10000 test images: 0.98
Test Loss: {:.4f} 0.04289078339934349
Epoch [30/30], Step [100/600], Acc: 0.9900 Loss: 0.0362
Epoch [30/30], Step [200/600], Acc: 1.0000 Loss: 0.0067
Epoch [30/30], Step [300/600], Acc: 1.0000 Loss: 0.0055
Epoch [30/30], Step [400/600], Acc: 0.9900 Loss: 0.0373
Epoch [30/30], Step [500/600], Acc: 0.9900 Loss: 0.0289
Epoch [30/30], Step [600/600], Acc: 0.9900 Loss: 0.0216
Test Accuracy of the model on the 10000 test images: 1.00
Test Loss: {:.4f} 0.0037076130975037813
```

```
[16]: x = range(1, num_epochs+1)
      acc = [i[0][0] for i in history]
```

```
val_acc = [i[1][0] for i in history]
loss = [i[1][0] for i in history]
val_loss = [i[1][1] for i in history]
plt.subplot(1, 2, 1)
plt.plot(x, acc, label="Accuracy")
plt.plot(x, val_acc, label="Validation Accuracy")
plt.legend(loc='upper right')
plt.title("Training and validation Accuracy")

plt.subplot(1, 2, 2)
plt.plot(x, loss, label="Loss")
plt.plot(x, val_loss, label="Validation Loss")
plt.legend(loc='upper right')
plt.title("Training and validation Loss")

plt.show()
```



Training and validation Accuracy

Training and validation Loss

[ ]:

[ ]: