Table of Contents

1. Installation

```
pip install torch torchvision torchaudio
```

Verify Installation:

```
import torch
print(torch.__version__)
```

---

2. Import PyTorch

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
```

---

3. Tensors

Create Tensors:

```
x = torch.tensor([5.0, 10.0, 15.0])
y = torch.zeros(3)               # Zero tensor
z = torch.ones(3)                # Ones tensor
rand = torch.rand(3, 3)          # Random tensor
```

Tensor Shape & Type:

```
print(x.shape)
print(x.dtype)
```

Reshape Tensors:

```
x = torch.rand(4, 4)
x_reshaped = x.view(2, 8)
```

---

4. Tensor Operations

```
a = torch.tensor([1, 2, 3])
b = torch.tensor([4, 5, 6])

# Basic Operations
c = a + b
d = a * b
e = torch.dot(a, b)              # Dot product
f = torch.sum(a)                 # Sum of elements
```

Matrix Multiplication:

```
x = torch.rand(2, 3)
y = torch.rand(3, 2)
z = torch.mm(x, y)
```

Element-wise Operations:

```
x = torch.rand(3)
y = torch.exp(x)                  # Exponential
z = torch.sqrt(x)                 # Square root
```

---

5. Autograd (Automatic Differentiation)

```
x = torch.tensor(2.0, requires_grad=True)
y = x**3 + 5*x
y.backward()                      # Calculate gradients
print(x.grad)                     # dy/dx = 3x² + 5
```

---

6. Neural Network Basics

Simple Neural Network:

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(4, 3)
        self.fc2 = nn.Linear(3, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.sigmoid(self.fc2(x))
        return x

net = Net()
print(net)
```

---

7. Loss and Optimizer

```
criterion = nn.MSELoss()                # Mean Squared Error
optimizer = optim.SGD(net.parameters(), lr=0.01)
```

---

8. Training a Neural Network

```
for epoch in range(100):
    optimizer.zero_grad()               # Zero gradients
    outputs = net(x)                    # Forward pass
    loss = criterion(outputs, y)        # Compute loss
    loss.backward()                     # Backward pass
    optimizer.step()                    # Update weights
```

---

9. Loading Data (Datasets & Dataloaders)

```
from torchvision import datasets, transforms

transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

train_data = datasets.MNIST(root='./data', train=True, download=True,
transform=transform)
train_loader = torch.utils.data.DataLoader(dataset=train_data,
batch_size=64, shuffle=True)
```

---

10. GPU Acceleration

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

```
model = Net().to(device)
```

---

11. Saving and Loading Models

```
# Save Model
torch.save(net.state_dict(), 'model.pth')

# Load Model
model = Net()
model.load_state_dict(torch.load('model.pth'))
model.eval()
```

---

12. Testing the Model

```
with torch.no_grad():
    for data in train_loader:
        images, labels = data
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
```

---

13. Common PyTorch Functions

Function	Description
torch.tensor()	Creates a tensor.
torch.rand()	Random tensor.
torch.zeros()	Zero tensor.
torch.ones()	Ones tensor.
torch.mm()	Matrix multiplication.
torch.sum()	Sum of elements.
torch.cat()	Concatenate tensors.

Function	Description
torch.max()	Maximum value.
torch.mean()	Mean value.
torch.relu()	ReLU activation function.

---

14. Plotting with PyTorch

```
import matplotlib.pyplot as plt

x = torch.linspace(-10, 10, 100)
y = torch.sin(x)

plt.plot(x.numpy(), y.numpy())
plt.title('Sine Wave')
plt.show()
```

---

15. Example: Linear Regression

```
# Data
x = torch.rand(100, 1)
y = 3 * x + 2 + torch.randn(100, 1) * 0.1

# Model
model = nn.Linear(1, 1)
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training Loop
for epoch in range(500):
    y_pred = model(x)
    loss = criterion(y_pred, y)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

```
# Visualization
plt.scatter(x.numpy(), y.numpy())
plt.plot(x.numpy(), y_pred.detach().numpy(), color='red')
plt.title('Linear Regression with PyTorch')
plt.show()
```