COMP9444
Neural Networks and Deep Learning

3c. PyTorch

Typical Structure of a PyTorch Progam

```
# create neural network
net = MyNetwork().to(device)  # CPU or GPU

train_loader = torch.utils.data.DataLoader(...)
test_loader  = torch.utils.data.DataLoader(...)

# choose between SGD, Adam or other optimizer
optimizer = torch.optim.SGD(net.parameters,...)

for epoch in range(1, epochs):
   train(args, net, device, train_loader, optimizer)
   test( args, net, device,  test_loader)
```

Defining a Network Structure

```
class MyNetwork(torch.nn.Module):

   def __init__(self):
      super(MyNetwork, self).__init__()
      # define structure of the network here

   def forward(self, input):
      # apply network and return output
```


Training

```
def train(args, net, device, train_loader, optimizer):

    for batch_idx, (data,target) in enumerate(train_loader):
        optimizer.zero_grad()     # zero the gradients
        output = net(data)        # apply network
        loss = ...                # compute loss function
        loss.backward()           # compute gradients
        optimizer.step()          # update weights
```

Testing

```
def test(args, model, device, test_loader):

    with torch.no_grad():  # suppress updating of gradients
        net.eval()   # toggle batch norm, dropout
        for data, target in test_loader:
            output = model(data)
            test_loss = ...
            print(test_loss)
        net.train()  # toggle batch norm, dropout back again
```