Simple Recipe for Neural Network

Define Class

```
class Network(nn.Module):
    def __init__(self):
        super().__init__()
        #define all the required layers
    def forward(self,feature):
        #define all the function applied on the layers and return output
```

Define all the required statistics

```
#move model to GPU
device= torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = Network(**kwargs).to(device)
optimizer=torch.optim.Adam(model.parameters(),lr=3e-5)
criterion=nn.MSEloss()
```

Loading data

```
#Transform train and test data to tensor
#Load into Dataloader
```

Training Model

```
Set number of epochs
for epoch in epochs:
    loss=0
    for batch,y in train_loader:
        #change batch shape appropriately
        optimizer.zero_grad()
        output=model(batch)
        train_loss=criterion(output,y)
        train_loss.backward()
        optimizer.step()
        loss=train_loss.item()
```

Testing Model

```
with torch.no_grad():
    for batch,y in test_loader:
        #change dimension of batch
        output=model(batch)
        #check result
```