Deeper Neural Networks : nn.ModuleList()

LATEST SUBMISSION GRADE

100%

1. 1 / 1 point

Consider the constructor for the following neural network class :

```
class Net(nn.Module):
    # Section 1:
    def __init__(self, Layers):
        super(Net,self).__init__()
        self.hidden = nn.ModuleList()
        for input_size,output_size in zip(Layers,Layers[1:]):
            self.hidden.append(nn.Linear(input_size,output_size
```

Let us create an object model = Net([2,3,4,5])

How many neurons are in the first hidden layers ?

- 2
- (•) 3
- 4

✓ **Correct**
correct

2. Consider the forward function , fill out the value for the if statement marked BLANK . 1 / 1 point

```
# Section 2:
    def forward(self, activation):
        L=len(self.hidden)
        for (l, linear_transform) in zip(range(L), self.hidden):
            if #BLANK
                activation = torch.relu(linear_transform(activa
            else:
                activation = linear_transform(activation)
        return activation
```

- l>L
- l > L-1
- (•) l<L-1

✓ **Correct**
correct

3. True or False we use the following Class or . Module for classification : 1 / 1 point

```
class Net(nn.Module):

    # Constructor
    def __init__(self, Layers):
        super(Net, self).__init__()
        self.hidden = nn.ModuleList()
        for input_size, output_size in zip(Layers, Layers[1:]):
            self.hidden.append(nn.Linear(input_size, output_siz

    # Prediction
    def forward(self, activation):
        L = len(self.hidden)
        for (l, linear_transform) in zip(range(L), self.hidden)
            if l < L - 1:
                activation = torch.relu(linear_transform(activa
            else:
                activation = torch.relu(linear_transform(activa
        return activation
```

- (●) false
- ( ) true

✓ **Correct**
correct