Attempt all the questions.

[CLO 1: Understand and design the structure of deep neural networks]

Q1: Analyze the given code to identify any errors and provide a corrected version of the code.

[10 marks]

	Code	Answer

1	import torch.nn as nn
	import torch
	
	class Neural Network(nn.Module):
	  def _init_(self, input_size, hidden_size, num_classes):
	    super(SimpleNet, self)._init_()
	    self.fc1 = nn.Linear(input_size, hidden_size)
	    self.relu = ReLU()
	    self.fc2 = nn.Linear(hidden_size, num_classes)
	
	  def forward(self, x):
	    x = self.fc1(x)
	    x = self.relu
	    x = self.fc2(x)
	
	model = SimpleNet(784, 100, 10)
	print(model(torch.rand(1, 784)).shape)	

```
    inputs = inputs.view(inputs.size(0), −1)
    outputs = model(inputs)
    loss = criterion(outputs, labels)

    loss.backward()
    optimizer.zero_grad()
    optimizer.step()

  print(f"Epoch {epoch + 1}, Loss: {loss.item():.4f}")
```

it bove

~~also zero g should b above o~~

3.

```
import torch
from torch.utils.data import DataLoader, TensorDataset

x_data = torch.rand(100, 10)
y_data = torch.randint(0, 2, (100,))
dataset = TensorDataset(x_data, y_data)

train_loader = DataLoader(dataset, batch_size
                  = 16, shuffle = True)

for batch in train_loader:
   inputs = batch[:, : −1]
   labels = batch[:, −1]
   print(inputs.shape, labels.shape)
```

~~there is issue in y-da~~

4.

```
import torch
x = torch.rand(32, 3, 28, 28)
x_reshaped = x.view(32, −1, 28)
print(x_reshaped.shape)
```

2025/05/06 14:57

5.	import torch
	a = torch.tensor([[1, 2], [3, 4]])
	b = a.unsqueeze(0)
	print(a.shape, b.shape)	

6.	import torch
	x = torch.tensor(5.0, requires_grad = True)
	y = x ** 2
	z = y.detach()
	print(y.requires_grad, z.requires_grad)	

7.	import torch
	x = torch.tensor(4.0, requires_grad = True)
	y = x ** 2 + 3 * x + 5
	y.backward()
	print(x.grad)	

8.	import torch
	x = torch.tensor(1.0, requires_grad = True)
	y = x ** 3 + x ** 2
	y.backward(retain_graph = True)
	y.backward()
	print(x.grad)	

9.	import torch
	x = torch.tensor(2.0, requires_grad = True)
	y = x ** 3
	for i in range(2):
	    y.backward()
	    print(x.grad)	

10.	import torch
	import numpy as np
	np_array = np.array([1, 2, 3, 4])
	tensor = torch.from_numpy(np_array)
	tensor[2] = 10	

National University of Computer and Emerging Sciences

Islamabad Campus

Part B: Now, consider the same neural network, but replace the SVM loss with Cross-Entropy Loss. Use the above given resulting logits matrix and the corresponding true class labels to compute the Cross-Entropy loss of each example and calculate the gradient of the loss w.r.t all input logits.

Note: Don't need to show all the steps.

Now Findiy cross entropy loss

creating an exp matrix

33.11	16.44	39 1180	60.68
492.79	40.44	10930	181.27
244.69	8997.8	9897.14	12.18
27.11	16.44	8124	90

→ Findiy exp

0	0.02	0.0166	0.96	0.039	1540.62
2	0.042	0.00034	0.9386	0.01	11652.45
1	0.022	0.081	0.895	0.001	11052.87
0	0.02	0.017	0.854	0.095	948.95

10

Findiy loss

$-\log(\log 0.02 + \log 0.9386 + \log 0.081 + \log 0.02)$

$-((1.698 + 0.02 + 1.09 + 2.698))$

calculated loss 5.50697

Findiy gradients

-0.98	0.0166	0.96	0.039
0.042	0.034	-0.063	0.01
0.022	-0.911	0.845	0.001

0061

	f_1 93.11	f_2 16.44	f_3 80.29	f_4 68.62	Y
1.	3.5	2.8	7.3 113.21	4.2 184.931	0
2.	6.2 [illegible]	3.7 40.49	9.3	5.2	2
3.	5.5 247.69	6.8 27.29	9.2 9842.12	2.5 17.88	1
4.	3.3 29.11	2.8 16.49	6.7 813.4	4.5 900.17	0

In case of multiclass svm loss

max(0.8 , 2.8-3.5), ma(7.3-3.5) max(0, 4.2-3.5)

0	0	3.8	0.7
0	0	0	0
0	0	2.4	0
0	0	3.4	1.2

Total loss = 11.5

logits matrix

Now finding derivatives

-2	0	-1	-1
0	0	0	0
0	-1	1	0
-2		1	1

10

0061

[CLO 2: Understand the different layers and their operation]

Q3: Use the space provided for answering the short questions. Be precise while answering the short questions. **[10 marks]**

(a) Prove the SoftMax loss contribution from a single example $L_i = -\log\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ can be equivalently written as $L_i = -e^{f_{y_i}} + \log\sum_j e^{f_j}$.

in case of log $\log\frac{a}{b}$ can be written as ~~log a - log~~

$\left\{\log\frac{a}{b} = \log a - \log b\right\}$

$-\log\frac{e^{f_{y_i}}}{\sum e^{f_j}}$

$-(\log e^{f_{y_i}} - \log\sum_j e^{f_j})$

$-((\log e^{1}).e^{f_{y_i}} - \log\sum_j e^{f_j})$

$= -e^{f_{y_i}} + \log\sum_j e^{f_j}$

02

(b) Difference between gradient descent, stochastic gradient descent, and batch gradient descent, which is more preferred one and why?

normal gradient descent calculate ~~outputs~~ ~~from backward function~~ derivatives in back propogation

(c) Why do we need derivatives w.r.t to inputs while computing back-propagation for affine layer.

because to compute value in backward pass we multiply all gradients local derivatives thats why we calculate gradients

2025/05/06 14:58

(d) How does the loss function of a Multi-Class SVM differ from Cross-Entropy Loss?

In case of multi class svm loss we find difference of each element ~~than~~ ~~given~~ index with y index given compare it with zero in case if it is greater than zero we consider that value and we sum up at the end in case of CSE Loss we find exponentials and then sum up and find probability ~~and~~ of y labels and sum them up at last

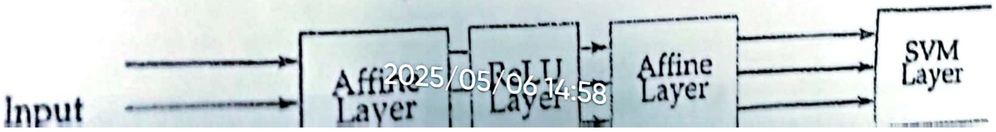
(e) Why is the softmax function typically used before applying Cross-Entropy Loss in multi-class classification?

[CLO 2: Understand the different layers and their operation]

Q4: .... **[30 marks]**

Part A: Consider a 3-layer neural network with a multi-class SVM loss function (architecture diagram is given in Figure 1). The network's final layer consists of 4 neurons, each representing a different output class. We input 4 examples into the network, and each neuron produces a logit score for each example. The resulting logits matrix and the corresponding true class labels are provided below. Compute the multi-class SVM loss of each example and calculate the gradient of the loss w.r.t all input logits.

Note: Don't need to show all the steps.



0061

Part C: Now, consider the same neural network, but replace the SVM loss with Cross-Entropy Loss. Implement a function to compute the loss and its gradient based on the given logits and actual labels.

```
def cross_entropy_loss(logits, y):
"""
Computes the cross − entropy loss and its gradient.

Parameters:
    logits (Tensor): Predicted logits from the model.
    y (Tensor): Ground − truth class labels.

Returns:
    loss (float): Scalar value representing the cross − entropy loss.
    grad (Tensor): Gradient of the loss with respect to the input logits.
"""
```

~~criterion = torch. crossEntropyloss ( )~~

def cross- Entropy loss (logits, y)

exp-matrix= np. exp (logits)

probabibilities_~~ep~~matrix = ~~logits~~ / np. sum (exp-matrix, ax

probability-matrix= logits / propability -matri

loss = -log (np. sum (~~probability matrix~~ np matrix , Y ~~[illegible]~~

~~logitsmatrix = [illegible]~~

return loss

67

~~method 2~~

alternative method using library ~~crite:~~ criterion= ~~cross-Ent~~ .torch

~~criterion~~ def cross - Entropy loss (lo

Loss = critereon (logit

return Loss

2025/05/06 14:59