CSE-5368 Neural Networks Fall 2026 Exam_01

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

Prob #	**1**	**2**	**3**	**4**	**Total**
Points	**25**	**25**	**25**	**25**	

Time: 80 Minutes

Solution

CSE-5368 Neural Networks Fall 2026 Exam_01

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

$$L_i = \sum_{j \neq i} max(0, y_j - y_i + \Delta)$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

$$H(p,q) = -\sum_x p(x) log(q(x))$$

$$L_i = -log(\frac{e^{y_i}}{\sum_j e^{y_j}})$$

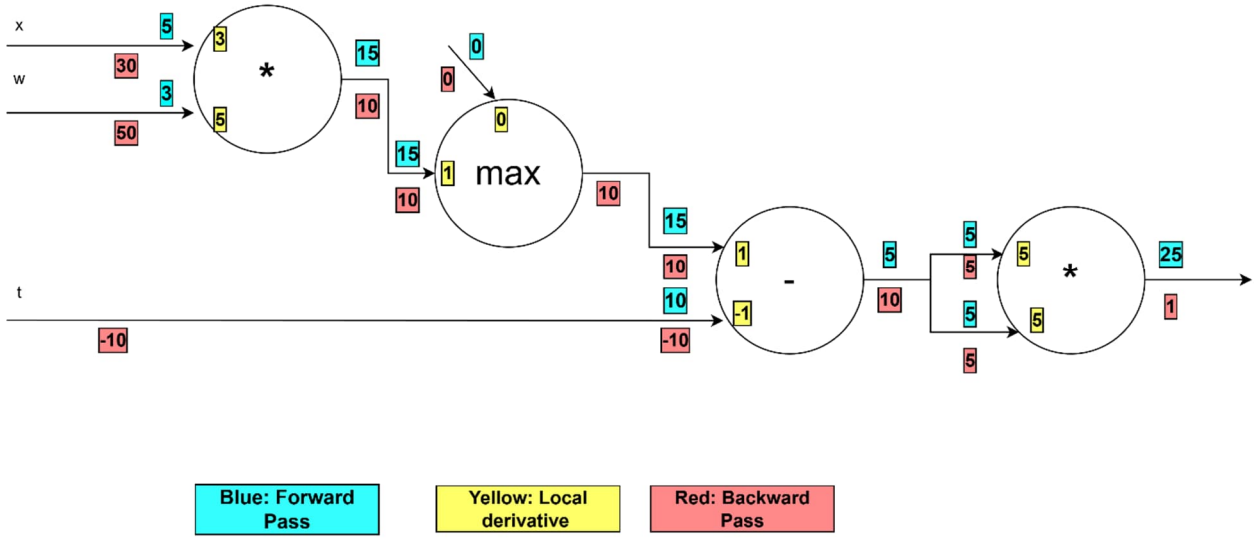
CSE-5368 Neural Networks Fall 2026 Exam_01

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

1. Consider the expression: $\boldsymbol{f(x,w,t) = (ReLU(x * w) - t)^2}$

Given the inputs $\boldsymbol{x = 5,\ \ w = 3,\ \ t = 10}$

Draw the computational graph and calculate the $\frac{\delta f(x,w,t)}{\delta x}$ and $\frac{\delta f(x,w,t)}{\delta w}$, $\frac{\delta f(x,w,t)}{\delta t}$
You MUST SHOW all the numerical values as they flow in the forward and backward path in the computational graph including the local derivatives.



CSE-5368 Neural Networks Fall 2026 Exam_01

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

Problem 1 Continued

CSE-5368 Neural Networks Fall 2026 Exam_01

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

2. Consider a convolutional neural network defined in pytorch.
Note: **Do NOT consider Biases**.

In PyTorch, the input/ ouput tensors shapes are: [batch_size, channels, height, width].

The weight shapes is: [out_channels, in_channels, kernel_height, kernel_width]

Input layer:

Input to this CNN are color images of size **3x100x70** with the **batch size = 30**

Note: Input image has **different horizontal and vertical** resolution.

Next layer is Conv2D layer (`PyTorch: nn.Conv2d`):

Number of filters: **12**, filter size: **6x6**; stride: **1x1**; padding: **2x2**

Q1: What is the shape of the weight matrix for this layer? **Q1:** _____

Q2: What is the shape of the output (tensor) of this layer? **Q2:** _____

Next layer is Conv2D layer (`PyTorch: nn.Conv2d`):

Number of filters**: 9**, filter size: **7x7**; stride: **2x2**; padding: **1x1**

Q3: What is the shape of the weight matrix for this layer? **Q3:** _____

Q4: What is the shape of the output (tensor) of this layer? **Q4:** _____

Next layer is Flatten layer:

Q5: What is the shape of the output (tensor) for this layer? **Q5:** _____

Next layer is Dense layer:

Number of nodes: **50**

Q6: What is the shape of the weight matrix for this layer? **Q6:** _____

Q7: What is the shape of the output (tensor) for this layer? **Q7:** _____

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

Problem 2 Continued

Q1: torch.Size([12, 3, 6, 6])
Q2: torch.Size([30, 12, 99, 69])
Q3: torch.Size([9, 12, 7, 7])
Q4: torch.Size([30, 9, 48, 33])
Q5: torch.Size([30, 14256])
Q6: torch.Size([50, 14256])
Q7: torch.Size([30, 50])

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

3. Complete the code for the following function.

```
import numpy as np

def forward(X, weights):
    out = X
    for W in weights:
        out = 1 / (1 + np.exp(-np.dot(out, W)))
    return out

def centered_diff(X_train, Y_train, weights, epochs, h=0.01):
# This function computes centered-difference gradients
# X_train: training input [num_samples, input_dim]
# Y_train: desired output [num_samples, output_dim]
# weights: list of weight matrices, one per layer
# h: centered-difference step size
# return: A list of gradient matrices
# Notes:
# Use only NumPy. Do not use torch.
# Do not adjust the weights. Just calculate the gradients
# Forward function is already provided.
# Initialize gradient matrices with zeros
# Ignore Bias
# Use MSE loss
# use:(f(x+h) - f(x-h)) / (2*h)
    grads = []
    for layer_idx in range(len(weights)):
        W = weights[layer_idx]
        grad = np.zeros_like(W, dtype=float)
        for i in range(W.shape[0]):
            for j in range(W.shape[1]):
                w_plus = [w.copy() for w in weights]
                w_minus = [w.copy() for w in weights]
                w_plus[layer_idx][i, j] += h
                w_minus[layer_idx][i, j] -= h
                y_plus = forward(X_train, w_plus)
                y_minus = forward(X_train, w_minus)

                e_plus = np.mean((Y_train-y_plus)**2)
                e_minus = np.mean((Y_train-y_minus)**2)

                grad[i, j] = (e_plus - e_minus) / (2 * h)

        grads.append(grad)

    return grads
```

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

Problem 3 Continued

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

4. What will be printed after running the following code segment.

CIFAR-10 contains 50,000 training images, and each image is a color image of size 3 x 32 × 32.

```
from torchvision import datasets, transforms
from torch.utils.data import DataLoader
transform = transforms.Compose([ transforms.ToTensor()])
train_ds = datasets.CIFAR10(root="./data", transform=transform)
train_loader = DataLoader(train_ds, batch_size=200, shuffle=False)
p, q = train_ds[0]
it=iter(train_loader)
x, y = next(it)
print("1:",len(train_ds))
print("2:",len(train_loader))
print("3:",p.shape)
print("4:",len(p))
print("5:",x.shape)
print("6:",len(y))
```

1: _____

2: _____

3: _____

4: _____

5: _____

6: _____

Last Name	First Name	Student ID Number
Solution	«First_Name»	«UTA_ID»

Problem 4 Continued

1: 50000
2: 250
3: torch.Size([3, 32, 32])
4: 3
5: torch.Size([200, 3, 32, 32])
6: 200