National University of Computer and Emerging Sciences

FAST School of Computing **Fall-2023** **Islamabad Campus**

CS-452: Deep Learning for Perception

Serial No:

Sessional Exam-I

Total Time: 1 Hour

Total Marks: 50

Saturday, 23rd September, 2023

Course Instructors

Akhtar Jamil

Signature of Invigilator

Student Name Roll No. Course Section Student Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space, write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **three? (3?)** different printed pages including this title page. There are total of **5?** questions.
5. Calculator sharing is strictly prohibited.
6. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Q-4	Q-5	Total
Marks Obtained						
Total Marks	20	10	10	10	10	50

Question 1. [10 x 2 = 20 Marks]

1 What is the vanishing gradient problem in deep neural networks, and how can it be mitigated?

The vanishing gradient problem in deep neural networks refers to the issue where gradients of the loss function become very small during training, causing slow convergence or learning stagnation. It can be mitigated by using proper weight initialization, batch normalization, optimal learning rate, etc.

2 Explain the impact of using the following regularization term in the loss function.

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m} \left(h_w\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \left|w_j\right|$$

It adds a penalty term to the loss function that is proportional to the absolute values of the weights. This encourages many of the weight parameters to become exactly zero, effectively removing certain features or inputs from the model. L1 regularization is useful for feature selection and reducing the complexity of the model, as it drives some weights to be zero, effectively "turning off" irrelevant or less important features. This can lead to simpler and more interpretable models, reduced overfitting, and improved generalization on data with a high dimensionality or many irrelevant features.

3 What is momentum, and why do we need it?

Momentum is an optimization technique that enhances convergence by accumulating past gradient information, allowing faster traversal of complex loss landscapes. Also it helps overcome the local optima. It reduces oscillations and accelerates training in deep learning models by providing memory of gradient history.

4 What is the purpose of pooling layers in CNNs?

Pooling layers in CNNs serve to reduce spatial dimensions, control overfitting, and increase translational invariance by downsampling feature maps through operations like max or average pooling.

5 Suppose you trained a neural network, you observe that the training accuracy was 99.3% while the test accuracy was 52%. What is this condition called? What will you do to overcome this issue?

The condition where the training accuracy is significantly higher than the test accuracy is called "overfitting." To overcome overfitting, you can consider various techniques, such as increasing the size of data, adding regularization, reducing the model's complexity, and employing dropout or batch normalization to prevent excessive reliance on specific features.

6 What do you understand by the term end-to-end learning in the deep learning context?

By end-to-end learning we mean that there is no need of hand crafted features rather the neural network directly maps raw input data to the desired output or task. It means that the entire process, from input to output, is learned automatically by the neural network. This approach is particularly advantageous in complex tasks where traditional feature engineering may be challenging or impractical. It generally leads to lower bias.

7 What is non-uniform learning? Is it good for training a neural network? Explain.

Non-uniform learning refers to the situation where some weights or parameters in the network converge faster or earlier than others during the training process. This can result in imbalanced updates, where certain weights receive more updates or training progress than others. It is not good in terms of training the network as it introduces instability in the whole network.

8 Write at least two advantages of including convolutional layers instead of just fully connected layers for computer vision related tasks.

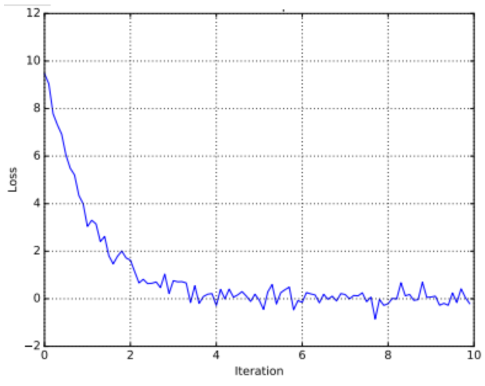
- Highly optimal features are extracted that may increase the accuracy.

- They will have less number of parameters as they share them.

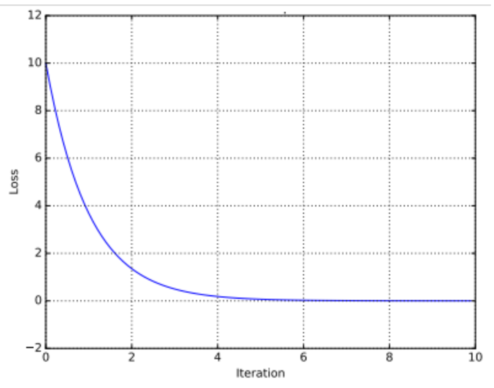
9 Why is it important to place non-linearities between the layers of neural networks?

Non-linearity introduces more degrees of freedom to the model. It lets it capture more complex representations which can be used for the task at hand. A deep neural network without non-linearities is essentially a linear regression.

10 Consider the following two loss curves for two different experiments shown in a) for experiment-1 and b) for experiment-2 for training a neural network. In which experiment a larger batch size is used? Explain your reason.



(a)



(b)

Experiment 2 has a larger batch size compared to Experiment -1 as the output curve is smoother than the other one. In smaller batch sizes, each batch contains fewer examples, leading to a higher variance in the estimated gradients, which may cause significant fluctuations in the loss curve, as shown in a).

Question 2. [5+5 Marks]

a) Imagine a Convolutional Neural Network (CNN) comprising two consecutive convolutional layers, each immediately followed by a pooling layer. In these convolutional layers, a 5x5 filter is applied, while a 2x2 filter with a stride of 1 is used for the pooling operation. The following table also specifies the number of filters for each convolutional layer.

Assuming an input image size of 32 x 32 x 3, your task is to calculate the output dimensions for each convolutional, pooling, and flattened layer in the network.

Seq. No	**Layers**	**No of Filters**	**Output Dimension**
1	Convolutional Layer-1	3	28 x 28 x 3
2	Pooling Layer -1	-	27 x 27 x 3
3	Convolutional Layer-2	5	23 x 23 x 5
4	Pooling Layer -2	-	22 x 22 x 5
5	Flattening Layer	-	1 x 2420

b) Consider the following input image of size 3 x 3. Find the output of convolving this image with the following given filter only for the pixels in the shaded region.

Input Image

0	0	0	0
0	2	1	0
0	3	1	0
0	0	0	0

Input Image

0	0	0	0
0	2	-5	0
0	2	-5	0
0	0	0	0

Filter

-1	0	1
-1	0	1
-1	0	1

Question 3. [5+5 Marks]

a) Explain the working of the gradient descent algorithm for weight update as used in the neural networks.

In the context of training neural networks, the Gradient Descent algorithm begins by initializing the network's weights. It proceeds with forward-propagating data through the network to make predictions, and a loss function quantifies the prediction errors. Backpropagation then computes the gradients of the loss concerning each weight, offering guidance for weight adjustments. These weight updates occur iteratively, as each weight is tuned in the direction opposite to its gradient and scaled by a predefined learning rate. This iterative process is repeated across multiple epochs until a predefined stopping criterion is met. The result is a neural network with updated weights that minimize the loss function, equipping it with the capability to provide accurate predictions on new data.

b) Write code to create two functions: one for computing the SoftMax function and another for calculating its derivative.

```
def softmax(logits):
    exp_logits = np.exp(logits - np.max(logits))
    sum_exp_logits = np.sum(exp_logits)
    softmax_probs = exp_logits / sum_exp_logits
    return softmax_probs
```

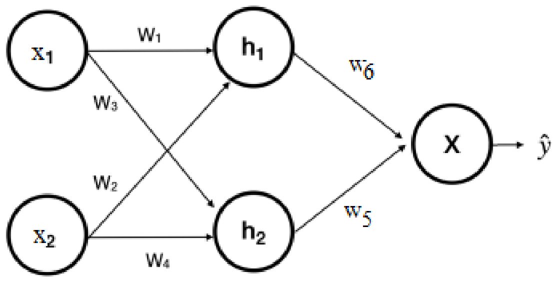
```
def softmax_derivative(logits):
    softmax_probs = softmax(logits)
    n = len(softmax_probs)
    out = -np.outer(softmax_probs, softmax_probs)
+ np.diag(softmax_probs)
    return out
```

Question 4. [10 Marks]

Consider the following ANN network. Assume that each neuron has a sigmoid activation function and all weights are initialized to 0.5. The following data set consists of two samples. Pass the whole data set through the network, and calculate the total error.

x_1	x_2	y
1	2	0
2	0	1

$$J(w) = \frac{1}{2m}\sum_{i=1}^{m}\left(y - y\right)^2$$



For the first sample (1, 2):
h_1 = sigmoid(0.5 * 1 + 0.5 * 2) = sigmoid(1.5) ≈ 0.8176
h_2 = sigmoid(0.5 * 1 + 0.5 * 2) = sigmoid(1.5) ≈ 0.8176

For the second sample (2, 0):
h_1 = sigmoid(0.5 * 2 + 0.5 * 0) = sigmoid(1.0) ≈ 0.7311
h_2 = sigmoid(0.5 * 2 + 0.5 * 0) = sigmoid(1.0) ≈ 0.7311

Output activation:
o_1 = sigmoid(0.5 * 0.8176 + 0.5 * 0.8176) = sigmoid(0.8176) ≈ 0.6935
o_1 = sigmoid(0.5 * 0.7311 + 0.5 * 0.7311) = sigmoid(0.7311) ≈ 0.6753

For the second sample (y = 0):
error_1 = (0 - 0.6935)^2 ≈ 0.4803

For the second sample (y = 1):
error_2 = (1 - 0.6753)^2 ≈ 0.0197

Total Error = error_1 + error_2
Total Error ≈ 0.4803 + 0.0197
Total Error ≈ 0.5000

Average Squared Error = Total Error / Total Number of Samples
Average Squared Error ≈ 0.5000 / 2
Average Squared Error ≈ 0.2500