ASSIGNMENT
Deep Learning using Python (CSE 4685)

Programme: B. Tech (CSE) **Semester: 7th**
Full Marks: 20 **Date of Submission: 20-12-2025**

Subject/Course Learning Outcome	*Taxonomy Level	Question Nos.	Marks
Able to apply the key fields of linear algebra and probability to build complex neural models.	L2	1	2
Able to analyse, design and implement Artificial Neural Networks (ANNs) and optimization techniques for solving complex learning problems.	L3	2,9	4
Able to analyse, build and implement Convolutional Neural Networks (CNNs) using PyTorch for solving classification problems.	L3	3,4,5,10	8
Able to analyse, design and implement autoencoder using PyTorch for nonlinear data handling.	L3	7	2
Able to design and interpret Recurrent Neural Networks (RNNs) and long short-term memory (LSTM) for sequential data analysis.	L3	6	2
Able to develop Generative Adversarial Networks (GANs) to synthesize new data.	L3	8	2

*Bloom's taxonomy levels: Knowledge (L1), Comprehension (L2), Application (L3), Analysis (L4), Evaluation (L5), Creation (L6).

Answer all questions. Each question carries equal mark.

- **Assignment scores/markings depend on neatness, clarity and date of submission.**
- **Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily.**
- **You are allowed to use only those concepts which are covered in the lecture class.**

1. What is the prerequisite for deep learning? Explain in details, how these concepts are helpful to build Deep learning models.

2. Given perceptron weights $w_1=1$, $w_2=-1$, $b=-0.2$. Write the equation of the decision boundary and classify the points (0.5,0.5) and (0.8,0.2)

3. Explain the role of the following layers in CNN:
    - Convolution layer
    - Pooling layer
    - Fully connected layer

4. Table1: Network parameter details

Layer	Parameters
Conv 2D	Filter size=5×5, stride=1, padding=0, #filters=6
Max Pooling 2D	Stride=2, filter size=2×2
Conv 2D	Filter size=5×5, stride=1, padding=0, #filters=16
Max Pooling 2D	Stride=2, filter size=2×2
Conv 2D	Filter size=5×5, stride=1, padding=0, #filters=32
Max Pooling 2D	Stride=2, filter size=2×2
Flatten	
Dense 1	120
output	10

Draw the CNN architecture with the following information given in **Table.1** for image classification. The input image size is 128×128×3. Find the total no. of learnable parameters in the network.

5. State batch normalization. Discuss the limitations of batch normalization and how layer and group normalization overcomes it with example.

6. Draw a RNN network with input(2) →hidden(2) →output(1), calculate the Y predicted for the following given parameter values in **Table.2** with input x_1=[1, 0], x_2=[0, 1] ,h_0=[0,0].

Table2: Network parameter details

Layer	**Weights**	**Bias**
Input to hidden	[0.5 0.8 , −0.3 0.2]	-
Hidden-to hidden	[0.1 −0.4, 0.2 0.3]	0.0,0.1
Hidden to output	[1.0, −1.0]	0.0

7. What is autoencoder? How does it work? What are the loss functions used in autoencoder, elaborate?

8. Answer the followings
    - What is a Generative Adversarial Network (GAN)?
    - Name the two main components of a GAN and state their roles.
    - What is meant by adversarial training?
    - Define generator loss and discriminator loss.

9. Implement a feedforward neural network with backpropagation using PyTorch to learn the XOR logic function for the following network architecture.

   Input layer: 2 neurons, Hidden layer: 2 neurons, Output layer: 1 neuron, Activation function: Sigmoid, Loss function: Mean Squared Error (MSE), Learning rate: 0.1.

10. Implement a Convolutional Neural Network (CNN) using PyTorch to classify images from the MNIST handwritten digits dataset. Display training and testing outcome in-terms of confusion matrix, accuracy, precision, recall and F1-score.

Dataset:
MNIST (Image size: 28 × 28 pixels, Color channels: 1 (grayscale), Number of classes: 10 (digits 0–9), Training samples: 60,000, Testing samples: 10,000)

Architecture details:
Input Layer: Input shape (28, 28, 1)
Convolution Layer 1: Number of filters: 32 Kernel size: 3 × 3 Stride: 1 Padding: valid
Activation function: ReLU
Max Pooling Layer 1: Pool size: 2 × 2 Stride: 2
Convolution Layer 2: Number of filters: 64 Kernel size: 3 × 3 Activation function: ReLUMax Pooling Layer 2: Pool size: 2 × 2
Fully Connected Layer: Number of neurons: 128 Activation function: ReLU
Output Layer: Number of neurons: 10 Activation function: Softmax