



LORDS INSTITUTE OF ENGINEERING AND TECHNOLOGY
A UGC Autonomous Institution | Approved by AICTE | Recognized by Govt. Of Telangana
Accredited by NBA& NAAC 'A' Grade | Included of 2F of UGC



Department of Computer Science And Engineering

Academic Year: 2026-2027

Student Roll Number:

160923733152

Student Name:

MOHAMMED UFRAAN

Year:

04

Semester:

07

Branch & Section:

CSE -A2
(C)

Course Name:

DEEP LEARNING TECHNIQUES

Course Code:

U23CS702

Assignment Number:

01

Date of Assignment:

11/08/26

Due date:

20/08/26

Date of Submission:

Marks Obtained:

10

Remarks:

Signature of Faculty with date



LORDS INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

Approved by AICTE/Affiliated to Osmania University/Estd.2002

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Assignment-1

MOHAMMED UFRAAN
160923733152

Course Name: Deep Learning Techniques	Course code: U23CS702
Yr & Sem & Sec: IV C	Academic Year: 2026-2027
Marks: 10	Date: 20/08/2026
Name of the Faculty: Md Anjar Ahsan	

Bloom's Taxonomy Levels (TL)		
TL1-Remember	TL2Understand	TL3-Apply
TL4-Analyze	TL5-Evaluate	TL6-Create

Assignment Questions: Roll no:001 to 030

Q.No.	Question	Marks	Course Outcome	BTL Level
1	Consider a neuron with: $x=2$, $w=0.5$, $b=0.2$ The target output is: $y=1$ Using sigmoid activation and a suitable loss function: 1. Calculate the neuron output. 2. Calculate the error. 3. Calculate the gradient. 4. Update the weight using learning rate $\eta=0.1$. Show all calculations.	2	CO1	BTL3
2	Feed-Forward and Backpropagation are complementary processes in neural network training. Justify this statement with a suitable mathematical explanation	2	CO1	BTL4
3	Explain the Softmax activation function mathematically. For the following logits: $Z = [2, 1, 0]$ calculate the Softmax probability for each class. Verify that the sum of probabilities is approximately 1.	3	CO2	BTL3
4	A hospital wants to develop a Deep Learning system to classify patients as Disease / No Disease. Answer: 1. Which neural network architecture would you select? 2. Which activation function would you use in the hidden layers? 3. Which activation function would you use in the output layer? 4. Which loss function would you select? 5. How would you handle overfitting? 6. Which evaluation metrics would you use? 7. How would you validate the model?	3	CO2	BTL4

NOTE:

1. Answers to the assignment questions is to be written on A4 size papers.
2. Each page of the assignment papers should contain full Roll number of the student.
3. The hard copy of the assignment is to be submitted on or before **20-08-2026**.

Faculty Name

Md Anjar Ahsan

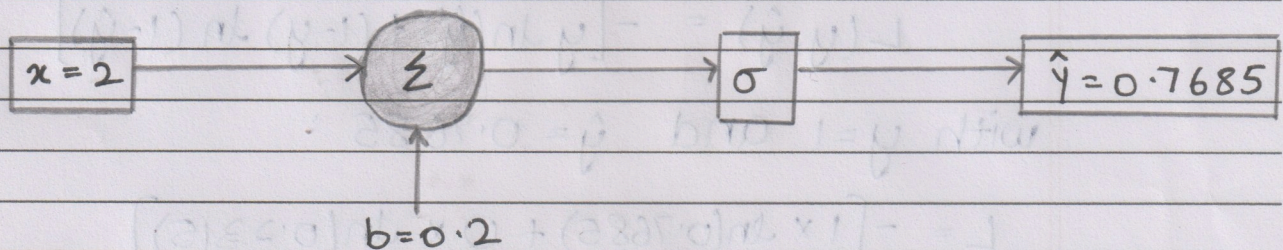
Q.No.	Question	Marks	Course Outcome	BT Level
1	Consider a neuron with $w=2$, $b=0.2$. The input vector is $x=[1, 0.5]^T$. Calculate the neuron output. Calculate the error. Calculate the gradient. Update the weight using learning rate $\eta=0.1$. Show all calculations.	2	CO1	BT1
2	Feed-forward and Backpropagation are complementary processes in neural network training. Justify this statement with a suitable mathematical explanation.	2	CO2	BT1
3	Explain the softmax activation function mathematically. For the following logits: $z=[2, 1, 0]$, calculate the softmax probability for each class. Verify that the sum of probabilities is approximately 1.	3	CO2	BT2
4	A hospital wants to develop a Deep Learning system to detect patients as Diabetic. No Diabetic patients. 1. Which neural network architecture would you select? 2. Which activation function would you use in the hidden layers? 3. Which activation function would you use in the output layer? 4. Which loss function would you select? 5. How would you handle overfitting? 6. Which evaluation metrics would you use? 7. How would you validate the model?	3	CO2	BT1

01) Consider a neuron with: $x = 2$, $w = 0.5$, $b = 0.2$.
The target output is $y = 1$. Using sigmoid activation add a suitable loss function:

- i. calculate the neuron output,
- ii. calculate the error,
- iii. calculate the gradient,
- iv. update the weight using learning rate $\eta = 0.1$.

Show all calculations.

Ans:



i. Computing Neuron Output (forward pass):

the weighted sum (pre-activation) is:

$$\begin{aligned} z &= w \cdot x + b \\ &= 0.5 \times 2 + 0.2 \\ &= 1.0 + 0.2 \\ &= 1.2 \end{aligned}$$

applying the sigmoid activation:

$$\begin{aligned} \hat{y} &= \sigma(z) = \frac{1}{1 + e^{-z}} \\ &= \frac{1}{1 + e^{-1.2}} \end{aligned}$$

since $e^{-1.2} \approx 0.3012$:

$$\hat{y} = \frac{1}{1 + 0.3012} = \frac{1}{1.3012} \approx \boxed{0.7685}$$

(ii) calculate Error (Loss):

since this is binary classification task ($y \in \{0, 1\}$), the suitable loss function is binary cross-entropy:

$$L(y, \hat{y}) = -[y \ln(\hat{y}) + (1-y) \ln(1-\hat{y})]$$

with $y=1$ and $\hat{y}=0.7685$:

$$\begin{aligned} L &= -[1 \times \ln(0.7685) + 0 \times \ln(0.2315)] \\ &= -\ln(0.7685) \approx -(-0.2632) \\ &= \boxed{0.2632} \end{aligned}$$

$$\boxed{\hat{y} = 0.7685} \rightarrow \boxed{L = -\ln(\hat{y}) = 0.2632} \leftarrow \boxed{y = 1}$$

(iii) calculate gradient:

for sigmoid + binary cross-entropy, a key simplification applies:

$$\frac{\partial L}{\partial z} = \hat{y} - y \Rightarrow \frac{\partial L}{\partial w} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w} = (\hat{y} - y) \cdot x$$

substituting values:

$$\begin{aligned} \frac{\partial L}{\partial w} &= (0.7685 - 1) \times 2 = (-0.2315) \times 2 \\ &= \boxed{-0.4630} \end{aligned}$$

(iv) Update weight (Gradient Descent):

$$w_{\text{new}} = w - \eta \cdot \frac{\partial L}{\partial w}$$

with $\eta = 0.1$:

$$w_{\text{new}} = 0.5 - 0.1 \times (-0.4630)$$

$$= 0.5 + 0.0463$$

$$= \boxed{0.5463}$$

Summary of Calculations:

Quantity	Value
Weighted sum $z = wx + b$	1.2
Neuron Output $\hat{y} = \sigma(z)$	0.7685
Loss $L = -\ln(\hat{y})$ (BCE)	0.2632
Gradient $\partial L / \partial w = (\hat{y} - y)x$	-0.4630
Updated weight w_{new}	0.5463

02.) Feed-Forward and Backpropagation are complementary processes in neural network training. Justify this statement with a suitable mathematical explanation.

Ans: A neural network is trained by the repeated alternation of two phases: the feed-forward pass, which produces a prediction, and backpropagation, which computes how every weight must change.

They are complementary because the forward pass defines WHAT the network computes, while backpropagation defines HOW it learns; each phase supplies the information the other phase needs.

→ Forward Pass (Function Evaluation):

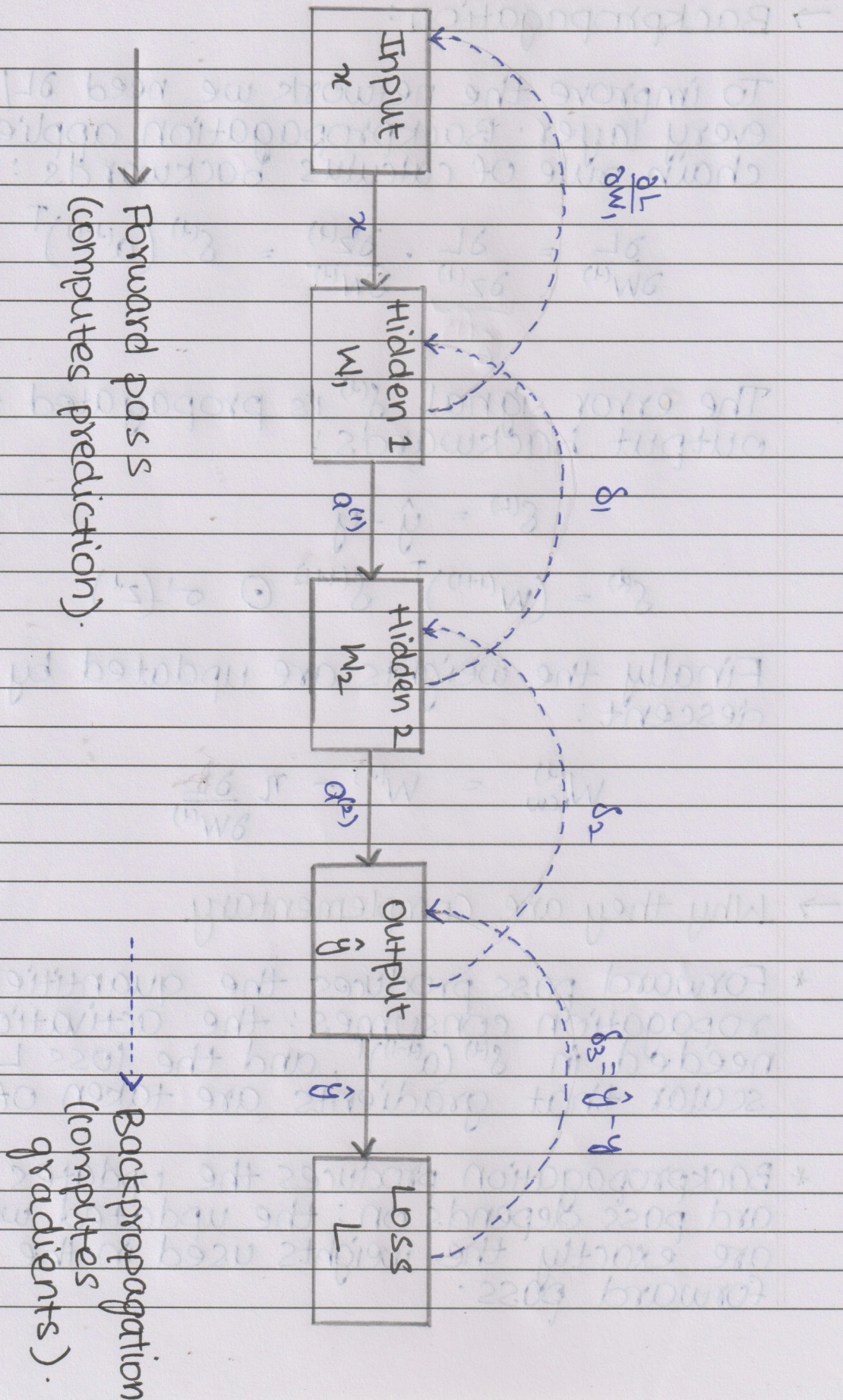
Inputs propagate layer by layer. For a network with L layers, each layer computes:

$$\begin{aligned} z^{(d)} &= W^{(d)} a^{(d-1)} + b^{(d)} \\ a^{(d)} &= \sigma(z^{(d)}) \\ d &= 1, 2, \dots, L \end{aligned}$$

where $a^{(0)} = x$. The final activation gives the prediction $\hat{y} = a^{(L)}$, and loss is computed:

$$L(y, \hat{y}) = -[y \ln \hat{y} + (1-y) \ln(1-\hat{y})]$$

This phase evaluates the network at the current weights - it answers "how well does the network currently perform?".



→ Backpropagation:

To improve the network we need $\partial L / \partial W^{(l)}$ for every layer. Backpropagation applies the chain rule of calculus backwards:

$$\frac{\partial L}{\partial W^{(l)}} = \underbrace{\frac{\partial L}{\partial z^{(l)}}}_{\delta^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial W^{(l)}} = \delta^{(l)} (a^{(l-1)})^T$$

The error signal $\delta^{(l)}$ is propagated from the output backwards:

$$\delta^{(l)} = \hat{y} - y$$
$$\delta^{(l)} = (W^{(l+1)})^T \delta^{(l+1)} \odot \sigma'(z^l)$$

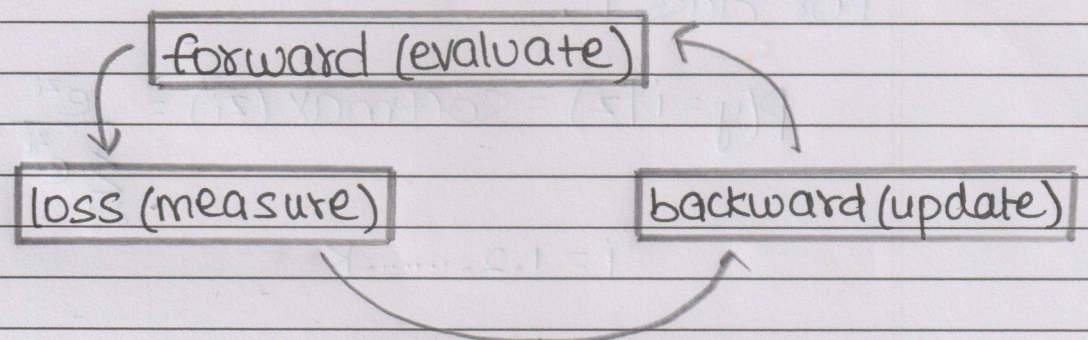
Finally the weights are updated by gradient descent:

$$W_{\text{new}}^{(l)} = W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}}$$

→ Why they are complementary

- * Forward pass produces the quantities backpropagation consumes: the activations $a^{(l)}$ are needed in $\delta^{(l)} (a^{(l-1)})^T$, and the loss L is the scalar that gradients are taken of.
- * Backpropagation produces the updates the forward pass depends on: the updated weights $W_{\text{new}}^{(l)}$ are exactly the weights used in the next forward pass.

- * One cannot work without the other: without the forward pass there is no loss to minimise; without backpropagation there is no way to know in which direction to adjust the weights.
- * Each training iteration is therefore a closed loop:



repeating until convergence.

Hence feed-forward and backpropagation are two halves of one training mechanism: the former evaluates the model, the latter improves it.

03.) Explain the Softmax activation function mathematically. For the following logits: $Z = [2, 1, 0]$ calculate the Softmax probability for each class. Verify that the sum of probabilities is approximately 1.

Ans: Softmax converts a vector of raw scores (logits) $z = [z_1, z_2, \dots, z_k]$ into a probability distribution over K classes.
For class i :

$$P(y = i | z) = \text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}},$$
$$i = 1, 2, \dots, K$$

Key properties:

Normalisation:

the denominator sums the exponentials of all logits, so $\sum_{i=1}^K P_i = 1$.

Range:

each probability lies in $(0, 1)$, hence the output is a valid probability distribution.

Monotonicity:

larger logits get larger probabilities - the function preserves ranking.

Amplification:

the exponential magnifies differences between logits, making the largest logit dominate.

Used in:

the output layer of multi-class classifiers, typically paired with categorical Cross-Entropy loss.

Computation for $Z = [2, 1, 0]$:

Compute exponentials:

$$e^{z_1} = e^2 \approx 7.3891, \quad e^{z_2} = e^1 \approx 2.7183, \quad e^{z_3} = e^0 = 1$$

Compute normalisation constant:

$$\sum_{j=1}^3 e^{z_j} = 7.3891 + 2.7183 + 1.0000 \\ = 11.1074$$

Compute each class probability:

$$P_1 = \frac{e^2}{11.1074} = \frac{7.3891}{11.1074} \approx 0.6652$$

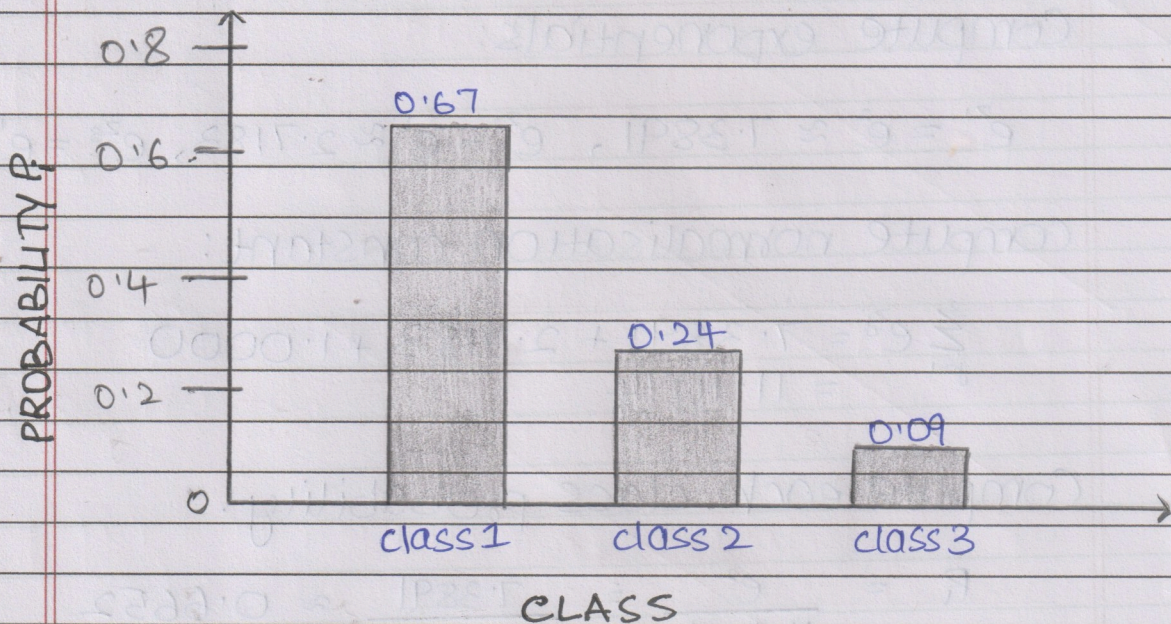
$$P_2 = \frac{e^1}{11.1074} = \frac{2.7183}{11.1074} \approx 0.2447$$

$$P_3 = \frac{e^0}{11.1074} = \frac{1.0000}{11.1074} \approx 0.0900$$

Verify the sum:

$$P_1 + P_2 + P_3 = 0.6652 + 0.2447 + 0.0900 \\ = 1.0000 \quad \checkmark$$

Class	Logit z_i	Exponential e^{z_i}	Softmax P_i
class 1	2	7.3891	0.6652
Class 2	1	2.7183	0.2447
Class 3	0	1.0000	0.0900
SUM	—	11.1074	1.0000



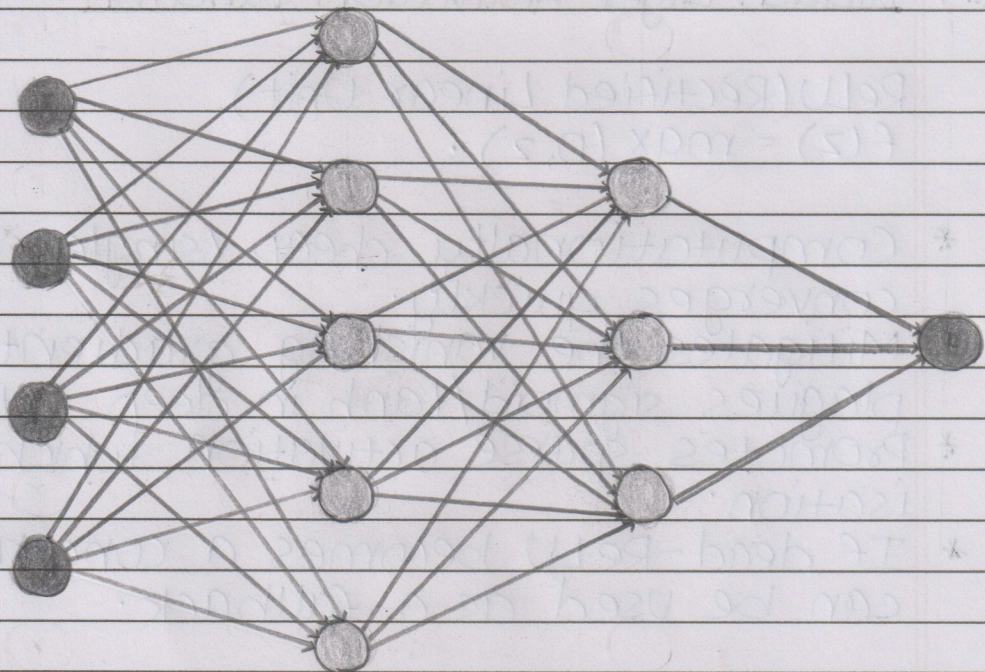
Verification:

the bar chart and table both show that the three softmax probabilities sum to 1.0000, confirming that the output is a valid probability distribution.

04.) A hospital wants to develop a deep learning system to classify patients as Disease / No Disease. Answer:

- (i.) which neural network architecture would you select?
- (ii.) which activation function would you use in the hidden layers?
- (iii.) which activation function would you use in the output layer?
- (iv.) which loss function would you select?
- (v.) how would you handle overfitting?
- (vi.) which evaluation metrics would you use?
- (vii.) how would you validate the model?

Ans:



Input Layer
(Features)

Hidden Layer 1
(ReLU)

Hidden Layer 2
(ReLU)

Output Layer
(Sigmoid)

(i) Neural Network Architecture:

A Multilayer Perceptron (MLP) - a feed forward fully connected with 2-3 hidden layers. This is appropriate because the input is structured tabular medical data (e.g.; age, blood pressure, glucose, lab values), not images or sequences.

CNNs (images) and RNNs/Transformers (sequences) are unnecessary here; an MLP is simpler, faster to train, and typically achieves strong performance on structured tabular data with a moderate number of patients.

(ii) Hidden Layer Activation Function:

ReLU (Rectified Linear Unit),
 $f(z) = \max(0, z)$:

- * Computationally cheap (single comparison) and converges quickly.
- * Mitigates the vanishing gradient problem that plagues sigmoid/tanh in deep networks.
- * Promotes sparse activation, which aids regularisation.
- * If dead-ReLU becomes a concern, Leaky ReLU can be used as a fallback.

(iii) Output Layer Activation Function:

Sigmoid, $\sigma(z) = \frac{1}{1 + e^{-z}}$

- * Output is a single value in $(0,1)$ that is interpreted as the probability of the positive class (Disease).
- * Perfectly suited to the binary Disease / No Disease problem.
- * Decision rule: predict Disease if $\hat{y} \geq 0.5$, otherwise No Disease.

(iv) Loss Function:

Binary Cross-Entropy (BCE):

$$L = -\frac{1}{N} \sum_{i=1}^N [y_i \ln(\hat{y}_i) + (1-y_i) \ln(1-\hat{y}_i)]$$

This is the natural companion to the sigmoid output and penalises confident wrong predictions heavily, which is important in a medical setting.

(v) Handling Overfitting:

- * Dropout: randomly drop neurons (e.g., 0.3-0.5) during training to prevent co-adaptation.
- * L2/weight regularisation: adds $\lambda \|w\|^2$ to the loss and ~~stop training when~~ keep weights small.
- * Early stopping: monitor validation loss and stop training when

it stops improving.

* Data augmentation:

for tabular data, use SMOTE to balance the classes; for imbalanced medical data, weight the loss by inverse class frequency.

* Keep the model small:

fewer parameters relative to the number of patient records.

* Batch normalisation:

to stabilise training and provide a mild regularising effect.

(vi.) Evaluation metrics:

For an imbalanced medical dataset, accuracy alone is misleading. Use:

* Accuracy: overall fraction correct (only informative if classes are balanced).

* Precision: of those predicted as Disease, how many truly have it: $\frac{TP}{TP+FP}$

* Recall/sensitivity: of those who actually have the disease, how many are caught: $\frac{TP}{TP+FN}$

* F1 Score: harmonic mean of precision and recall: $2 \cdot \frac{P \cdot R}{P+R}$

- * AUC-ROC: measures discrimination ability across all thresholds; robust to class imbalance.
- * Confusion Matrix: gives a full picture of TP/FP/TN/FN.
- * Specificity: how well healthy patients are not mislabeled. $\frac{TN}{TN + FP}$.

(vii) Model Validation:

Stratified Train/Validation/Test Split (e.g. 70/15/15) - stratification preserves the disease ratio in each subset; the test set is held out and used only once at the end.

Stratified k-Fold Cross Validation (e.g. $k=5$ or 10) the data is split into k -folds; the model is trained on $k-1$ folds and evaluated on the remaining fold, rotating until every fold has been the validation set. This gives a robust estimate of generalisation when data is limited.

Hyperparameter tuning on the validation folds only (grid search or random search), with the test set reserved for final assessment.

For clinical deployment, also monitor performance on new incoming patient data (temporal validation) to guard against distribution shift.