❓ Questions

1. Introduction to Deep Learning & ANNs

- **ML vs. DL:** What are the three main differences between traditional Machine Learning and Deep Learning regarding data requirements, hardware, and feature engineering?
- **Why DL?:** What kind of data is Deep Learning particularly well-suited for, which is often a challenge for traditional ML models?
- **Biological Inspiration:** What is the "firing threshold" of a biological neuron (e.g., -50mV), and what does it inspire in the artificial neuron model?
- **ANN Definition:** What is an Artificial Neural Network (ANN) at its most basic level?

2. The MCP Neuron (1943)

- **Components:** What are the allowed values for inputs (x_i) and weights (w_i) in the McCulloch-Pitts (MCP) neuron model?
- **Activation:** What is the activation rule for an MCP neuron? (How does it decide to output a 1 or a 0?)
- **Logic Gates:** A single MCP neuron can model an OR gate with $w_1=1$, $w_2=1$. What Threshold (T) should be used?
- **Logic Gates:** To model an AND gate (with $w_1=1, w_2=1$), what must the Threshold (T) be?

3. Linear Separability & Its Limits

- **Linear Boundary:** What does it mean for a problem to be "linearly separable"?
- **The XOR Problem:** Why is it impossible for a single MCP neuron or a single Perceptron to solve the XOR (Exclusive OR) problem?
- **Solving XOR:** How is the XOR problem solved using neurons?

4. The Perceptron Model (1958)

- **Improvements:** What key improvements did Rosenblatt's Perceptron make upon the MCP neuron in terms of inputs and weights?
- **Affine Transformation:** The Perceptron combines inputs, weights, and a bias. What is the mathematical formula for this combination, often called the "affine transformation"?
- **Activation Function:** What is the name of the activation function used by the Perceptron (the "hard step"), and what is its output?

5. Training: The Perceptron Algorithm

- **"The Perceptron Trick":** What is the learning rule for the Perceptron? When does the model *actually* update its weights?
- **Update Rule (Case 1):** If a point is falsely negative (actual $y=1$, predicted $\hat{y}=0$), how are the weights (w) and bias (b) updated?
- **Update Rule (Case 2):** If a point is falsely positive (actual $y=0$, predicted $\hat{y}=1$), how are the weights (w) and bias (b) updated?
- **Limitations:** What is the main drawback of the Perceptron learning algorithm in terms of tracking overall model performance?

6. Training: The Modern Approach (Loss & Gradients)

- **Loss Function:** Why do we need a loss function (or cost function)? What does it tell us?
- **Regression Loss:** For a regression task, what is the formula for Mean Squared Error (MSE) loss (or the simplified "Squared Loss" used in the examples)?
- **Forward Propagation:** What are the 4 main steps of Forward Propagation (starting from input x and ending with loss L)?
- **The Problem with 'Sign':** Why can we not use Gradient Descent with the Perceptron's Sign/Step activation function?
- **The 'Sigmoid' Solution:** What is the Sigmoid function, and what property makes it essential for modern neural networks?

7. Backpropagation & Gradient Descent

- **Gradient Descent:** What is the goal of the Gradient Descent algorithm?
- **Update Rule:** What is the mathematical formula for updating a weight (w) using Gradient Descent, a learning rate (η), and the loss (L)?
- **Backpropagation:** What is the purpose of the Backpropagation algorithm?
- **Chain Rule:** What is the mathematical rule that Backpropagation uses to calculate the gradient for each weight in the network?
- **Chain Rule (Example):** To find $\frac{\partial L}{\partial w_1}$, you must chain several derivatives. What are the four parts of this chain as shown in the documents?
- **Practical Training:** What is the difference between "Batch Gradient Descent" and "Stochastic Gradient Descent (SGD)"?
- **Epoch:** What does one "epoch" signify during the training of a neural network?

❓ Questions and Answers

1. Introduction to Deep Learning & ANNs

- **ML vs. DL: What are the three main differences between traditional Machine Learning and Deep Learning regarding data requirements, hardware, and feature engineering?**
    - **Data Requirements:** Traditional ML works well with small to moderate-sized data, while Deep Learning is "data-hungry" and requires very large amounts of data to achieve high performance.
    - **Hardware:** ML models can easily run on standard CPUs. DL models, however, perform complex computations (like matrix multiplication) and require high-performance GPUs or TPUs for efficient training.
    - **Feature Engineering:** In traditional ML, feature engineering is a crucial, manual step where a data scientist must create high-quality features. In DL, the neural network learns the features automatically from the raw data.
- Why DL?: What kind of data is Deep Learning particularly well-suited for, which is often a challenge for traditional ML models?
  Deep Learning is particularly well-suited for handling large, complex, unstructured data, such as images, audio, and text.
- Biological Inspiration: What is the "firing threshold" of a biological neuron (e.g., -50mV), and what does it inspire in the artificial neuron model?
  A biological neuron at rest is around -70mV. It "fires" when the sum of incoming excitatory signals causes its voltage to cross a "firing threshold" of approximately -50mV. This inspired the concept of the Threshold (T) in the MCP neuron and Perceptron models.
- ANN Definition: What is an Artificial Neural Network (ANN) at its most basic level?
  An ANN is a computational model that is designed to mimic the structure and function of the human brain's biological neural networks. It is built from interconnected nodes, or artificial neurons.

---

2. The MCP Neuron (1943)

- **Components: What are the allowed values for inputs (x_i) and weights (w_i) in the McCulloch-Pitts (MCP) neuron model?**
    - **Inputs (x_i):** Must be **binary** (e.g., {0, 1}).
    - **Weights (w_i):** Must be either **+1 (excitatory)** or **-1 (inhibitory)**.
- Activation: What is the activation rule for an MCP neuron? (How does it decide to output a 1 or a 0?)
  The neuron computes a weighted sum of its inputs ($\sum w_i x_i$). If this sum is greater than or equal to a fixed Threshold (T), the neuron outputs 1 (it "fires"); otherwise, it outputs 0.
- Logic Gates: A single MCP neuron can model an OR gate with $w_1=1$, $w_2=1$. What Threshold (T) should be used?

A Threshold (T) = 1 should be used. This way, if x_1 is 1 (sum=1), x_2 is 1 (sum=1), or both are 1 (sum=2), the sum will be ≥ 1 and the neuron will fire.

- Logic Gates: To model an AND gate (with $w_1=1, w_2=1$), what must the Threshold (T) be?
  The Threshold (T) = 2 must be used. This ensures the neuron only fires when both x_1 and x_2 are 1, making the sum exactly 2.

---

3. Linear Separability & Its Limits

- Linear Boundary: What does it mean for a problem to be "linearly separable"?
  A problem is linearly separable if you can draw a single straight line (or a hyperplane in higher dimensions) to perfectly separate the two classes of data points.
- The XOR Problem: Why is it impossible for a single MCP neuron or a single Perceptron to solve the XOR (Exclusive OR) problem?
  It is impossible because the XOR problem is not linearly separable. A single neuron can only create one linear boundary, and you cannot draw a single straight line to separate the (0,1) and (1,0) points from the (0,0) and (1,1) points.
- Solving XOR: How is the XOR problem solved using neurons?
  The XOR problem is solved by using multiple layers of MCP neurons. This allows the network to create multiple decision boundaries, which can be combined to form a non-linear region.

---

4. The Perceptron Model (1958)

- **Improvements: What key improvements did Rosenblatt's Perceptron make upon the MCP neuron in terms of inputs and weights?**
  - **Inputs (x_i):** Can be **real-valued numbers** (e.g., from $-\infty$ to $+\infty$), not just binary {0, 1}.
  - **Weights (w_i):** Can also be **real-valued numbers**, not just fixed at +1 or -1.
- Affine Transformation: The Perceptron combines inputs, weights, and a bias. What is the mathematical formula for this combination, often called the "affine transformation"?
  The formula is $z = w^T x + b$, which expands to $z = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$.
- Activation Function: What is the name of the activation function used by the Perceptron (the "hard step"), and what is its output?
  It is called the Sign function (or step function). It takes the real-valued output of the affine transformation (z) and converts it to a binary output (e.g., 0 or 1).

---

5. Training: The Perceptron Algorithm

- "The Perceptron Trick": What is the learning rule for the Perceptron? When does the model actually update its weights?
  The learning rule is to update the weights and bias only when the model makes a misclassification. If the classification is correct, no action is required.
- Update Rule (Case 1): If a point is falsely negative (actual $y=1$, predicted $\hat{y}=0$), how are the weights (w) and bias (b) updated?
  The weights and bias are updated by "adding" the point's features to them. This pulls the boundary closer to the point:
  - **$w_{new} = w_{old} + x$**
  - **$b_{new} = b_{old} + 1$**
- Update Rule (Case 2): If a point is falsely positive (actual $y=0$, predicted $\hat{y}=1$), how are the weights (w) and bias (b) updated?
  The weights and bias are updated by "subtracting" the point's features. This pushes the boundary away from the point:
  - **$w_{new} = w_{old} - x$**
  - **$b_{new} = b_{old} - 1$**
- Limitations: What is the main drawback of the Perceptron learning algorithm in terms of tracking overall model performance?
  The algorithm updates based on a single point at a time. This means that while it may correctly classify one point, it could simultaneously misclassify a significant number of other points. It lacks a mechanism to track the overall model performance or error.

---

6. Training: The Modern Approach (Loss & Gradients)

- Loss Function: Why do we need a loss function (or cost function)? What does it tell us?
  We need a loss function to aggregate the total error into a single number. This number tells us how well our model is performing overall and whether it is improving or degrading during training.
- Regression Loss: For a regression task, what is the formula for Mean Squared Error (MSE) loss (or the simplified "Squared Loss" used in the examples)?
  The simplified Squared Loss formula used for calculating the gradient is $L = \frac{1}{2}(\hat{y} - y)^2$.
- **Forward Propagation: What are the 4 main steps of Forward Propagation (starting from input x and ending with loss L)?**
  1. Calculate the **Affine Transformation** ($z = w^T x + b$).
  2. Apply the **Activation Function** ($h = f(z)$).
  3. Get the final **Prediction** ($\hat{y} = h$).
  4. Calculate the **Loss** ($L = \frac{1}{2}(\hat{y} - y)^2$).
- The Problem with 'Sign': Why can we not use Gradient Descent with the Perceptron's Sign/Step activation function?
  We cannot use Gradient Descent because the Sign function is not differentiable and not continuous. It has a "hard step" at 0 where a derivative (or slope) cannot be calculated,

and the derivative is 0 everywhere else. This provides no information for updating the weights.

- The 'Sigmoid' Solution: What is the Sigmoid function, and what property makes it essential for modern neural networks?
  The Sigmoid function (e.g., $\sigma(z) = \frac{1}{1+e^{-z}}$) is a smooth, S-shaped curve. Its essential property is that it is continuous and differentiable at all points. This allows us to calculate a gradient and use backpropagation to update the network's weights.

---

7. Backpropagation & Gradient Descent

- Gradient Descent: What is the goal of the Gradient Descent algorithm?
  The goal is to find the optimal values for the weights (w) and bias (b) that minimize the Loss function.
- Update Rule: What is the mathematical formula for updating a weight (w) using Gradient Descent, a learning rate (η), and the loss (L)?
  The formula is: $w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$.
- Backpropagation: What is the purpose of the Backpropagation algorithm?
  The purpose of Backpropagation is to efficiently calculate the gradient (the partial derivative) of the loss function with respect to every single weight and bias in the network. It tells us how much each parameter contributed to the final error.
- Chain Rule: What is the mathematical rule that Backpropagation uses to calculate the gradient for each weight in the network?
  It uses the Chain rule of calculus.
- Chain Rule (Example): To find $\frac{\partial L}{\partial w_1}$, you must chain several derivatives. What are the four parts of this chain as shown in the documents?
  The four parts are:
  $\left(\frac{\partial L}{\partial \hat{y}}\right) \cdot \left(\frac{\partial \hat{y}}{\partial h}\right) \cdot \left(\frac{\partial h}{\partial z}\right) \cdot \left(\frac{\partial z}{\partial w_1}\right)$
- **Practical Training: What is the difference between "Batch Gradient Descent" and "Stochastic Gradient Descent (SGD)"?**
  - **Batch Gradient Descent** processes the **entire dataset** to calculate the gradient before making a single weight update. It is accurate but very slow.
  - **Stochastic Gradient Descent (SGD)** processes **one data point at a time** and updates the weights after each point. It is much faster but the training process can be "noisy" or unstable.
- Epoch: What does one "epoch" signify during the training of a neural network?
  An epoch represents one complete pass of the entire training dataset through the neural network (both forward and backward).