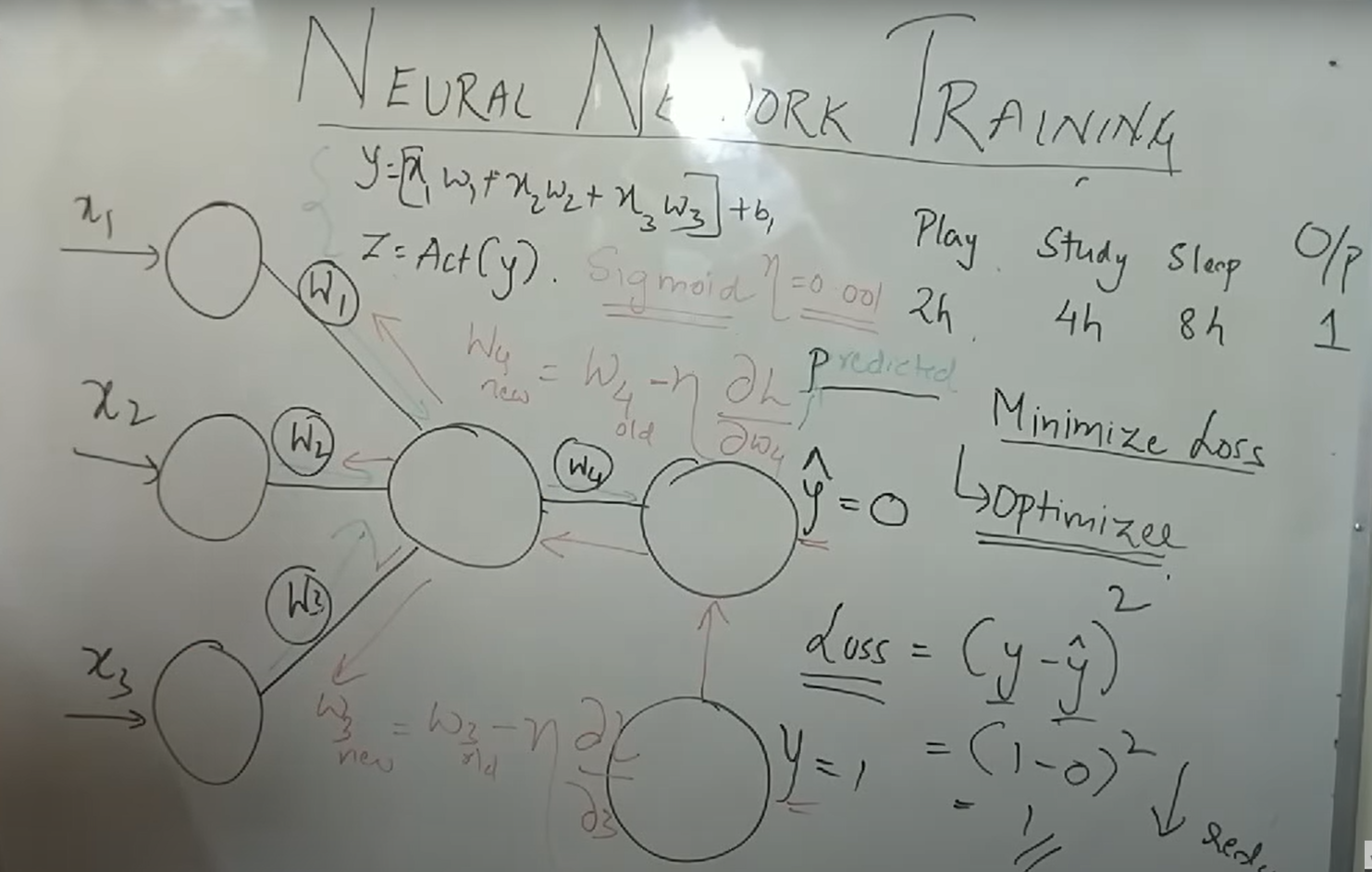
**Unit 4  
Neural Network Training**

### Neural Network Training



### 4.1 Forward Propagation

Forward propagation is the mechanism by which data flows **from input to output** through the network. It models how information is processed inside neurons.

**Steps**

* **Input Layer**: Raw features (x₁, x₂, x₃, …) enter the network. Each feature represents measurable aspects of the data (e.g., hours of play, study, sleep).
* **Weighted Sum + Bias**: Every input is multiplied by a weight (w₁, w₂, …), and a bias term is added:

y=(x1w1+x2w2+x3w3)+by = (x₁w₁ + x₂w₂ + x₃w₃) + b

Theory: Weights determine the strength of influence of each feature. Bias shifts the activation threshold, allowing flexibility in decision boundaries.

* **Activation Function**: The sum is passed through a non-linear function (Sigmoid, ReLU):

z=Act(y)z = Act(y)

Theory: Without activation, the network would behave like a simple linear model. Activation allows the network to learn complex, non-linear mappings.

* **Output Layer**: Produces predicted value (ŷ). For classification, this could be a probability; for regression, a numerical value.

**Purpose**

* Encodes input information into hidden representations.
* Enables generalization by stacking multiple transformations.

### 4.2 Backward Propagation

Backward propagation, or **backprop**, is the process of correcting mistakes by sending error signals backward. It ensures the network improves over time.

**Steps**

1. **Loss Calculation**
   * Error between predicted and actual output is computed with a loss function. Example:

Loss=(y−y^)2Loss = (y - ŷ)^2

* + Theory: The loss quantifies "how wrong" the network is. Lower loss means better predictions.

1. **Gradient Calculation**
   * Using the **chain rule of calculus**, the derivative of the loss is computed with respect to each weight.
   * Theory: The gradient shows the direction and magnitude in which each weight should change to reduce loss.
2. **Weight Update**
   * Weights are updated as:

wnew=wold−η∂L∂ww\_{new} = w\_{old} - η \frac{∂L}{∂w}

* + Here, η (learning rate) controls step size in optimization.
  + Theory: Small η → slow learning; large η → risk of overshooting. Finding the right η is critical for convergence.

**Purpose**

* Minimizes loss by continuously correcting weights.
* Drives the network to **approximate the true mapping** between inputs and outputs.

**Summary**

* Forward propagation models how inputs pass through weighted connections and activations to generate predictions.
* Backward propagation measures the error, computes gradients, and optimizes weights to minimize the loss.
* Together, they form the **training cycle**: forward pass (predict) → backward pass (learn) → repeat until convergence.