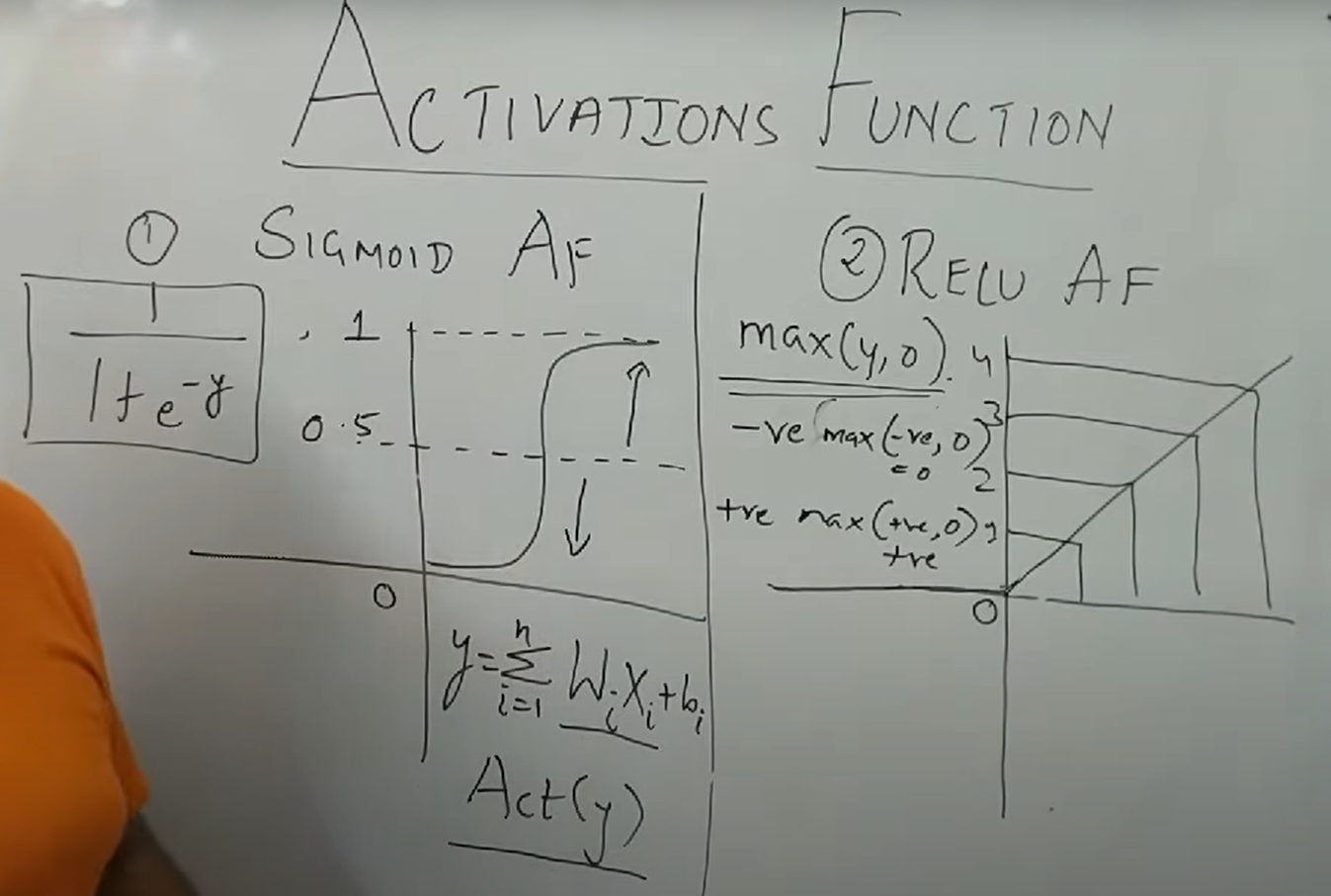
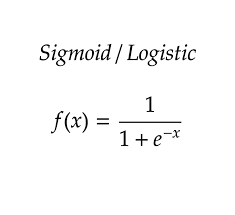
**Unit 3  
Activation Functions – Sigmoid & ReLU**

### 3.1 Sigmoid Activation Function



The **Sigmoid** activation function maps input values into the range **(0, 1)**, making it interpretable as probabilities.

**Formula**:



**Concept**

* Produces an **S-shaped curve**.
* Very negative inputs → output close to 0.
* Very positive inputs → output close to 1.
* Input near 0 → output around 0.5.

**Real-World Applications**

* Logistic regression for **binary classification**.
* Output layers in binary classification neural networks.
* Probability-based tasks like **fraud detection, medical diagnosis**.

**Advantages**

* Smooth and differentiable curve → gradients can be computed easily.
* Outputs can be interpreted as **probabilities**.
* Historically important in early neural networks.

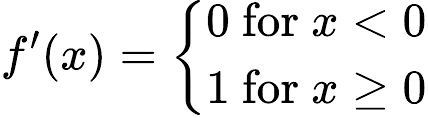
**Disadvantages**

* **Vanishing gradient problem**: for very high/low inputs, gradient approaches 0 → slows or halts learning.
* **Not zero-centered** → can cause inefficient weight updates.
* **Computationally expensive** compared to simpler functions (involves exponentials).
* Can **saturate** (flat regions), limiting learning ability in deep networks.

### 3.2 ReLU (Rectified Linear Unit) Activation Function

The **ReLU** function is widely used in deep learning because of its simplicity and efficiency.

**Formula**:



**Concept**

* If input > 0 → output = input (linear behavior).
* If input ≤ 0 → output = 0 (neuron inactive).
* Creates **sparse activations**, turning off some neurons for certain inputs.

**Real-World Applications**

* Default choice in **deep neural networks**.
* Used extensively in **CNNs for image recognition**, **object detection**, and **deep feedforward networks**.
* Applied in large-scale architectures (e.g., ResNet, VGG, Transformers).

**Advantages**

* Very **computationally efficient** (simple max operation).
* Helps reduce **vanishing gradient problem** → enables deeper networks.
* Encourages **sparse activations**, improving generalization.
* Works well with large datasets and complex architectures.

**Disadvantages**

* **Dying ReLU problem**: neurons may get stuck at 0 for all inputs (never activate again).
* **Unbounded** for positive values → outputs can grow large, risking exploding activations.
* Sensitive to **large learning rates**, which may cause instability.

📝 **Summary**

* **Sigmoid**: maps inputs to [0,1], suitable for probability interpretation. Best for output layers in binary tasks, but limited by vanishing gradients.
* **ReLU**: simple, efficient, and dominant in deep learning. Best for hidden layers in modern architectures, but suffers from the dying ReLU issue.
* Choice depends on task: **Sigmoid for probabilistic outputs**, **ReLU for deep feature extraction and large-scale networks**.