**1. Explanation of Activation Functions**

Activation functions decide whether a neuron should be activated or not. They introduce non-linearity into the network, allowing it to learn complex patterns.

**a) Sigmoid**

* Output range: 000 to 111.
* Formula: σ(x)=11+e−x\sigma(x) = \frac{1}{1 + e^{-x}}σ(x)=1+e−x1​.
* **Pros:** Useful for probability-based outputs.
* **Cons:** Causes **vanishing gradients** in deep networks.

**b) tanh (Hyperbolic Tangent)**

* Output range: −1-1−1 to 111.
* Formula: tanh⁡(x)=ex−e−xex+e−x\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}tanh(x)=ex+e−xex−e−x​.
* **Pros:** Centered around zero, better than sigmoid.
* **Cons:** Still suffers from **vanishing gradients** for large/small values.

**c) ReLU (Rectified Linear Unit)**

* Formula: f(x)=max⁡(0,x)f(x) = \max(0, x)f(x)=max(0,x).
* Output range: 000 to +∞+\infty+∞.
* **Pros:** Fast and avoids vanishing gradient problem.
* **Cons:** Causes **dead neurons** (output stuck at 0 for negative values).

**d) ELU (Exponential Linear Unit)**

* Formula: f(x)={xif x>0α(ex−1)if x≤0f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (e^x - 1) & \text{if } x \leq 0 \end{cases}f(x)={xα(ex−1)​if x>0if x≤0​
* **Pros:** Fixes dying ReLU problem by allowing small negative outputs.
* **Cons:** More computationally expensive.

**e) Leaky ReLU**

* Formula: f(x)={xif x>00.01xif x≤0f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{if } x \leq 0 \end{cases}f(x)={x0.01x​if x>0if x≤0​
* **Pros:** Prevents dead neurons by allowing small gradients for negative values.
* **Cons:** Not always the best choice compared to ELU.

**f) Swish**

* Formula: f(x)=x⋅σ(x)=x⋅11+e−xf(x) = x \cdot \sigma(x) = x \cdot \frac{1}{1 + e^{-x}}f(x)=x⋅σ(x)=x⋅1+e−x1​.
* **Pros:** Smoother than ReLU, helps optimization in deep networks.
* **Cons:** Computationally expensive.

**2. What Happens When You Change the Learning Rate?**

* **Increase learning rate** → Faster convergence but risk of **overshooting** or instability.
* **Decrease learning rate** → More stable training but slower convergence (or getting stuck in local minima).

**3. What Happens When You Increase the Number of Hidden Neurons?**

* **Pros:** More capacity to learn complex patterns.
* **Cons:** More parameters, higher computation time, risk of **overfitting**.

**4. What Happens When You Increase Batch Size?**

* **Large batch size** → Faster training, smoother gradient updates, but may **generalize poorly**.
* **Small batch size** → Noisy updates, better generalization, but slower training.

**5. Why Do We Use Regularization to Avoid Overfitting?**

Regularization prevents overfitting by reducing model complexity. Techniques include:

* **L1/L2 Regularization (Lasso/Ridge)** → Penalizes large weights.
* **Dropout** → Randomly deactivates neurons during training.
* **Early Stopping** → Stops training when validation loss increases.

**6. What Are Loss and Cost Functions?**

* **Loss function** → Measures error for a **single** training example.
* **Cost function** → Average loss over the **entire dataset**.

Examples:

* **MSE (Mean Squared Error)** → Regression tasks.
* **Cross-Entropy Loss** → Classification tasks.

**7. What is Underfitting in Neural Networks?**

* When the model **fails to learn** from the training data.
* Happens due to **too simple architecture**, insufficient training, or high regularization.

**8. Why Do We Use Dropout in Neural Networks?**

* Dropout **randomly disables** neurons during training.
* Prevents over-reliance on certain features → better **generalization**.
* Works like an **ensemble of different subnetworks**.