1. Explain the Activation Functions in your own language

1. sigmoid
2. tanh
3. ReLU
4. ELU
5. LeakyReLU
6. swish
7.  Activation Functions: a) **Sigmoid**: Sigmoid is an S-shaped activation function. It squashes input values to a range between 0 and 1. It's useful in binary classification problems as it models probability-like outputs. However, it has vanishing gradient problems for deep networks and is rarely used in hidden layers.
8. b) **tanh (Hyperbolic Tangent)**: Tanh is similar to the sigmoid but squashes input values between -1 and 1. It's zero-centered, which can help mitigate gradient problems in deep networks. It's often used in hidden layers of neural networks.
9. c) **ReLU (Rectified Linear Unit)**: ReLU is a simple but effective activation function. It returns the input for positive values and zero for negative values. It helps with the vanishing gradient problem and speeds up training. It's widely used in hidden layers.
10. d) **ELU (Exponential Linear Unit)**: ELU is an activation function that is similar to ReLU but has negative values for certain inputs. This helps prevent the "dying ReLU" problem. ELU is differentiable for all inputs and can lead to faster convergence.
11. e) **LeakyReLU**: LeakyReLU is similar to ReLU but allows a small gradient for negative values. This prevents neurons from becoming completely inactive and helps with the dying ReLU problem.
12. f) **Swish**: Swish is a relatively recent activation function that combines the advantages of ReLU and Sigmoid. It is similar to ReLU for positive values and has smooth gradients for optimization.

2. What happens when you increase or decrease the optimizer learning rate?

Optimizer Learning Rate:

* Increasing the learning rate may lead to faster convergence, but it can also make the training process unstable, with overshooting and divergence.
* Decreasing the learning rate can lead to more stable training and fine-tuning but may slow down convergence or lead to getting stuck in local minima. The choice of learning rate depends on the specific problem and network architecture.

3. What happens when you increase the number of internal hidden neurons?

Number of Hidden Neurons:

* Increasing the number of hidden neurons generally increases the capacity of the model to learn complex patterns in the data.
* However, too many neurons can lead to overfitting, where the model performs well on training data but poorly on unseen data. It also increases computational cost.

4. What happens when you increase the size of batch computation?

Batch Size:

* Increasing the batch size in batch computations can lead to faster training because it processes more data points in parallel.
* Larger batch sizes can provide more stable gradient estimates, which can help the training process converge faster.
* However, very large batch sizes can require more memory, and there may be diminishing returns in terms of speed and convergence.

5. Why we adopt regularization to avoid overfitting?

Regularization for Overfitting:

* Regularization techniques, such as L1 and L2 regularization, dropout, and early stopping, are adopted to avoid overfitting.
* Overfitting occurs when a model learns to fit the training data too closely, capturing noise in the data rather than the underlying patterns.
* Regularization helps constrain the model's capacity and encourages it to generalize better to unseen data.

6. What are loss and cost functions in deep learning?

Loss and Cost Functions:

* Loss and cost functions quantify the error between the predicted values and the actual target values.
* In deep learning, the loss function measures how well the model is performing on the training data, while the cost function is an aggregate measure of performance over the entire dataset.

7. What do ou mean by underfitting in neural networks?

Underfitting:

* Underfitting in neural networks occurs when the model is too simple to capture the underlying patterns in the data.
* It leads to poor performance on both training and validation data, indicating that the model lacks the capacity to learn the task.

8. Why we use Dropout in Neural Networks?

Dropout in Neural Networks:

* Dropout is used to prevent overfitting in neural networks.
* It randomly deactivates a subset of neurons during training, making the model more robust and preventing it from relying too heavily on specific neurons.
* Dropout effectively regularizes the model and improves its generalization to unseen data.