1. Explain the Activation Functions in your own language

1. Sigmoid - **Sigmoid Function acts as an activation function in machine learning which is used to add non-linearity in a machine learning model, in simple words it decides which value to pass as output and what not to pass.**
2. Tanh - **The hyperbolic tangent activation function is also referred to simply as the Tanh (also “tanh” and “TanH“) function. It is very similar to the sigmoid activation function and even has the same S-shape. The function takes any real value as input and outputs values in the range -1 to 1**
3. ReLU - **The Rectified Linear Unit is the most commonly used activation function in deep learning models. The function returns 0 if it receives any negative input, but for any positive value x it returns that value back. But the ReLU function works great in most applications, and it is very widely used as a result.**
4. ELU - **Exponential Linear Unit (ELU) is a variation of ReLU with a better output for z < 0. The function is defined as: ELU function (image by author) The hyperparameter α controls the value to which an ELU saturates for negative net inputs.**
5. LeakyReLU - **Leaky ReLU function is an improved version of the ReLU activation function. As for the ReLU activation function, the gradient is 0 for all the values of inputs that are less than zero, which would deactivate the neurons in that region and may cause dying ReLU problem. Leaky ReLU is defined to address this problem.**
6. Swish **- Swish is a smooth, non-monotonic function that consistently matches or outperforms ReLU on deep networks applied to a variety of challenging domains such as Image classification and Machine translation. It is unbounded above and bounded below & it is the non-monotonic attribute that actually creates the difference.**

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

**Ans : Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train.**

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

**Ans : An inordinately large number of neurons in the hidden layers can increase the time it takes to train the network. The amount of training time can increase to the point that it is impossible to adequately train the neural network.**

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

**Ans : higher batch sizes leads to lower asymptotic test accuracy. we can recover the lost test accuracy from a larger batch size by increasing the learning rate. starting with a large batch size doesn't “get the model stuck” in some neighbourhood of bad local optimums.**

5. Why we adopt regularization to avoid overfitting?

**Ans : Regularization is a technique used to reduce the errors by fitting the function appropriately on the given training set and avoid overfitting. If you've built a neural network before, you know how complex they are. This makes them more prone to overfitting. Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.**

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

Ans : A loss function/error function is for a single training example/input. A cost function, on the other hand, is the average loss over the entire training dataset. The optimization strategies aim at “minimizing the cost function”.

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

Ans : A model is said to be underfitting when it's not able to classify the data it was trained on. We can tell that a model is underfitting when the metrics given for the training data are poor, meaning that the training accuracy of the model is low and/or the training loss is high.

8. Why we use Dropout in Neural Networks?

Ans : A Simple Way to Prevent Neural Networks from Overfitting, 2014. Because the outputs of a layer under dropout are randomly subsampled, it has the effect of reducing the capacity or thinning the network during training. As such, a wider network, e.g. more nodes, may be required when using dropout.