1. Explain the Activation Functions in your own language

1. Sigmoid - Sigmoid takes a real value as input and outputs another value between 0 and 1. It’s easy to work with and has all the nice properties of activation functions: it’s non-linear, continuously differentiable, monotonic, and has a fixed output range.
2. Tanh - Tanh squashes a real-valued number to the range [-1, 1]. It’s non-linear. But unlike Sigmoid, its output is zero-centered. Therefore, in practice the tanh non-linearity is always preferred to the sigmoid nonlinearity.
3. ReLU - A recent invention which stands for Rectified Linear Units. The formula is deceptively simple: max(0,z). Despite its name and appearance, it’s not linear and provides the same benefits as Sigmoid (i.e. the ability to learn nonlinear functions), but with better performance.
4. ELU - Exponential Linear Unit or its widely known name ELU is a function that tend to converge cost to zero faster and produce more accurate results. Different to other activation functions, ELU has a extra alpha constant which should be positive number. ELU is very similiar to RELU except negative inputs. They are both in identity function form for non-negative inputs. On the other hand, ELU becomes smooth slowly until its output equal to -α whereas RELU sharply smoothes.
5. LeakyReLU - LeakyRelu is a variant of ReLU. Instead of being 0 when z<0, a leaky ReLU allows a small, non-zero, constant gradient α (Normally, α=0.01). However, the consistency of the benefit across tasks is presently unclear. [1]
6. Swish - Mathematical formula: Y = X \* sigmoid(X) Bounded below but Unbounded above: Y approach to constant value at X approaches negative infinity but Y approach to infinity as X approaches infinity. Derivative of Swish, Y' = Y + sigmoid(X) \* (1-Y) Soft curve and non-monotonic function.

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

Ans. A learning rate that is too small may never converge or may get stuck on a suboptimal solution. When the learning rate is too large, gradient descent can inadvertently increase rather than decrease the training error.

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

Ans. each hidden neuron added will increase the number of weights, thus it is recommended to use the least number of hidden neurons that accomplish the task. Using more hidden neurons than required will add more complexity.

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

Ans. large batch size means the model makes very large gradient updates and very small gradient updates. The size of the update depends heavily on which particular samples are drawn from the dataset. On the other hand using small batch size means the model makes updates that are all about the same size.

5. Why we adopt regularization to avoid overfitting?

Ans. By adding regularization term, the value of weights matrices reduces by assuming that a neural network having less weights makes simpler models. And hence, it reduces the overfitting to a certain level.

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

Ans. The terms cost function & loss function are analogous. Loss function: Used when we refer to the error for a single training example. Cost function: Used to refer to an average of the loss functions over an entire training dataset.

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

Ans. Underfitting is a scenario in data science where a data model is unable to capture the relationship between the input and output variables accurately, generating a high error rate on both the training set and unseen data.

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.