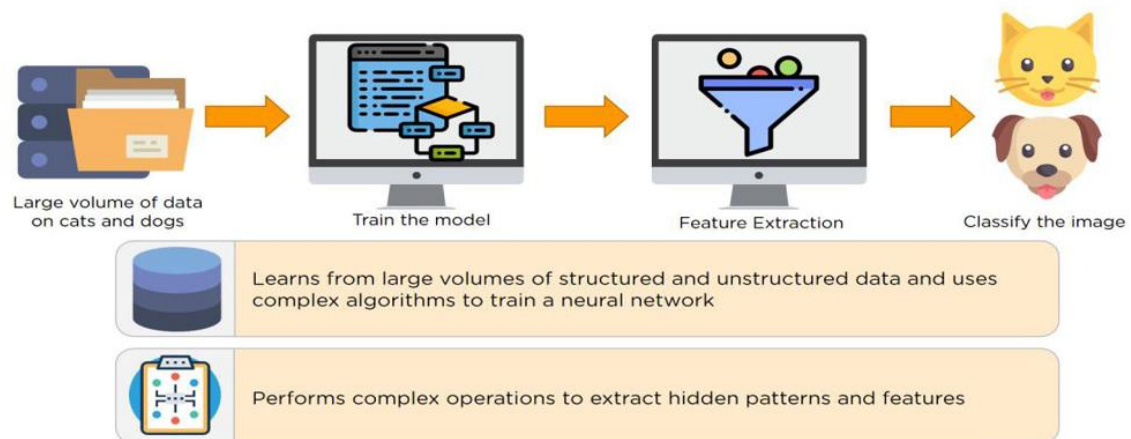


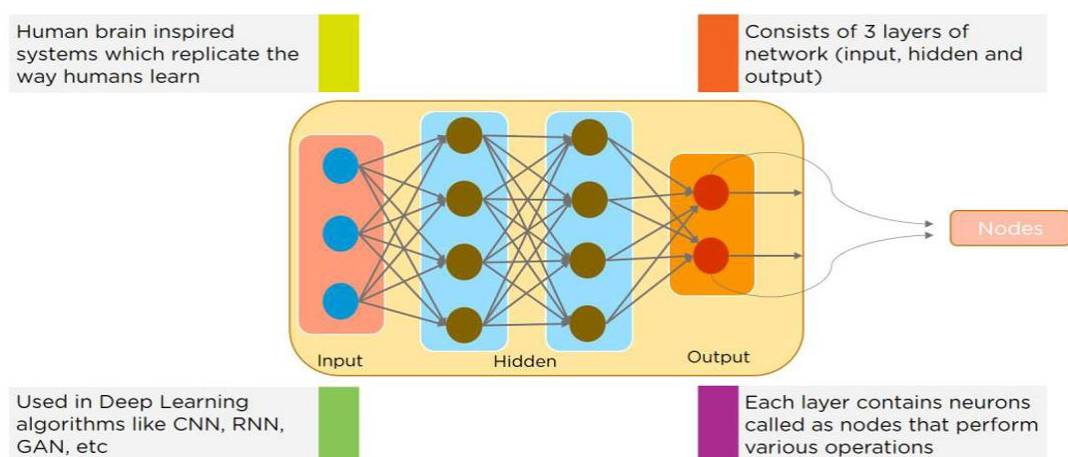
1. What is Deep Learning?

[Deep Learning](#) involves taking large volumes of structured or unstructured data and using complex algorithms to train neural networks. It performs complex operations to extract hidden patterns and features (for instance, distinguishing the image of a cat from that of a dog).



2. What is a Neural Network?

[Neural Networks](#) replicate the way humans learn, inspired by how the neurons in our brains fire, only much simpler.



The most common Neural Networks consist of three network layers:

1. An input layer

2. A hidden layer (this is the most important layer where feature extraction takes place, and adjustments are made to train faster and function better)
3. An output layer

Each sheet contains neurons called “nodes,” performing various operations. Neural Networks are used in [deep learning algorithms](#) like CNN, RNN, GAN, etc.

3. What Is a Multi-layer Perceptron(MLP)?

As in Neural Networks, MLPs have an input layer, a hidden layer, and an output layer. It has the same structure as a single layer perceptron with one or more hidden layers. A single layer perceptron can classify only linear separable classes with binary output (0,1), but MLP can classify nonlinear classes.

Except for the input layer, each node in the other layers uses a nonlinear activation function. This means the input layers, the data coming in, and the activation function is based upon all nodes and weights being added together, producing the output. MLP uses a supervised learning method called “backpropagation.” In backpropagation, the neural network calculates the error with the help of cost function. It propagates this error backward from where it came (adjusts the weights to train the model more accurately).

4. What Is Data Normalization, and Why Do We Need It?

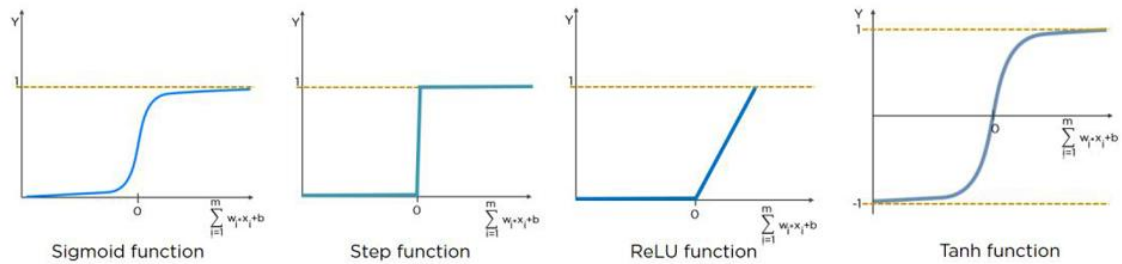
The process of standardizing and reforming data is called “Data Normalization.” It’s a pre-processing step to eliminate data redundancy. Often, data comes in, and you get the same information in different formats. In these cases, you should rescale values to fit into a particular range, achieving better convergence.

5. What is the Boltzmann Machine?

One of the most basic Deep Learning models is a Boltzmann Machine, resembling a simplified version of the Multi-Layer Perceptron. This model features a visible input layer and a hidden layer – just a two-layer neural net that makes stochastic decisions as to whether a neuron should be on or off. Nodes are connected across layers, but no two nodes of the same layer are connected.

6. What Is the Role of Activation Functions in a Neural Network?

At the most basic level, an activation function decides whether a neuron should be fired or not. It accepts the weighted sum of the inputs and bias as input to any activation function. Step function, Sigmoid, ReLU, Tanh, and Softmax are examples of activation functions.



7. What Is the Cost Function?

Also referred to as “loss” or “error,” cost function is a measure to evaluate how good your model’s performance is. It’s used to compute the error of the output layer during backpropagation. We push that error backward through the neural network and use that during the different training functions.

Cost Function:

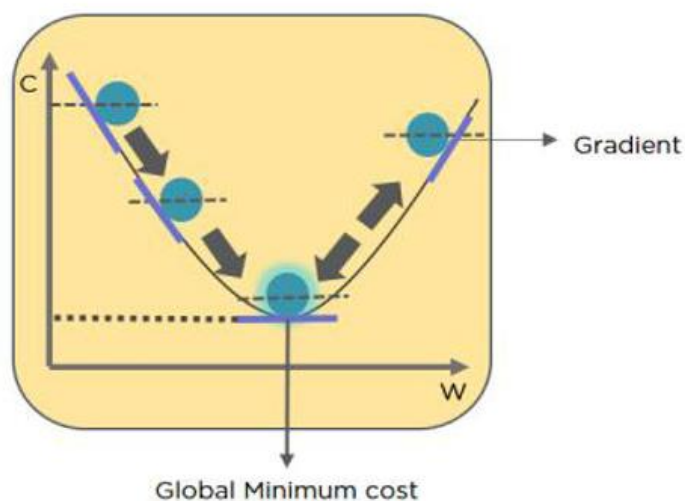
$$C = \frac{1}{2} (Y - \hat{Y})^2$$

Y -----> Original Output

\hat{Y} -----> Predicted Output

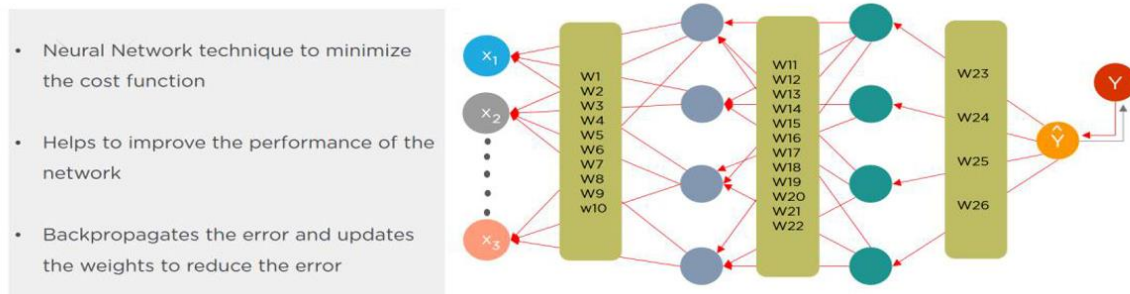
8. What Is Gradient Descent?

Gradient Descent is an optimal algorithm to minimize the cost function or to minimize an error. The aim is to find the local-global minima of a function. This determines the direction the model should take to reduce the error.



9. What Do You Understand by Backpropagation?

Backpropagation is a technique to improve the performance of the network. It backpropagates the error and updates the weights to reduce the error.



10. What Is the Difference Between a Feedforward Neural Network and Recurrent Neural Network?

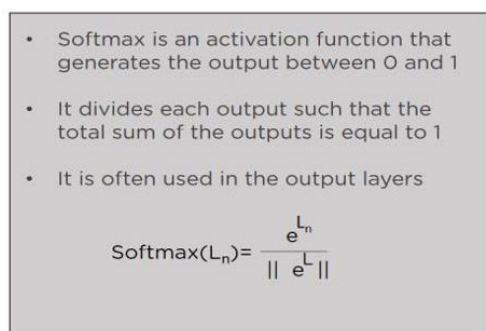
A Feedforward Neural Network signals travel in one direction from input to output. There are no feedback loops; the network considers only the current input. It cannot memorize previous inputs (e.g., [CNN](#)).

11. What Are the Applications of a Recurrent Neural Network (RNN)?

The [RNN](#) can be used for sentiment analysis, text mining, and image captioning. Recurrent Neural Networks can also address time series problems such as predicting the prices of stocks in a month or quarter.

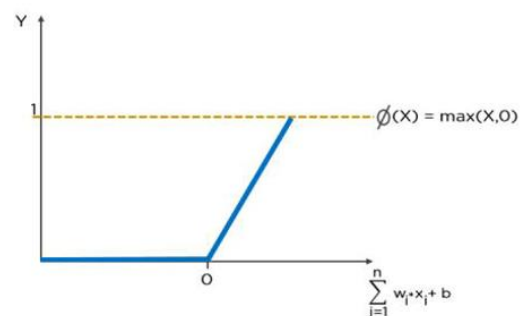
12. What Are the Softmax and ReLU Functions?

Softmax is an activation function that generates the output between zero and one. It divides each output, such that the total sum of the outputs is equal to one. Softmax is often used for output layers.



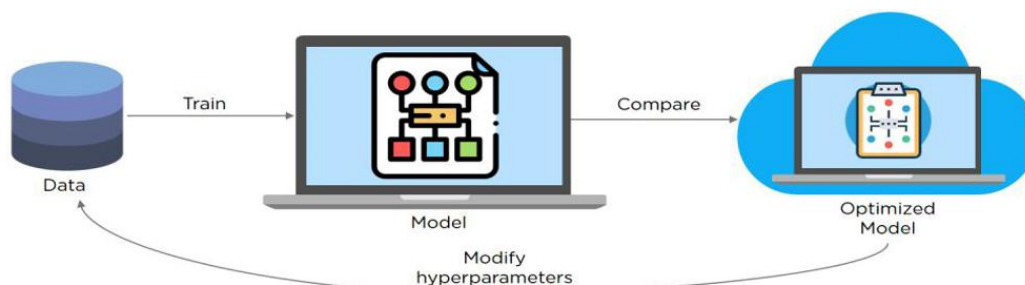
ReLU (or Rectified Linear Unit) is the most widely used activation function. It gives an output of X if X is positive and zeros otherwise. ReLU is often used for hidden layers.

- ReLU stands for Rectified Linear Unit and is the most widely used activation function
- It gives an output of X if X is positive and 0 otherwise
- It is often used in the hidden layers



13. What Are Hyperparameters?

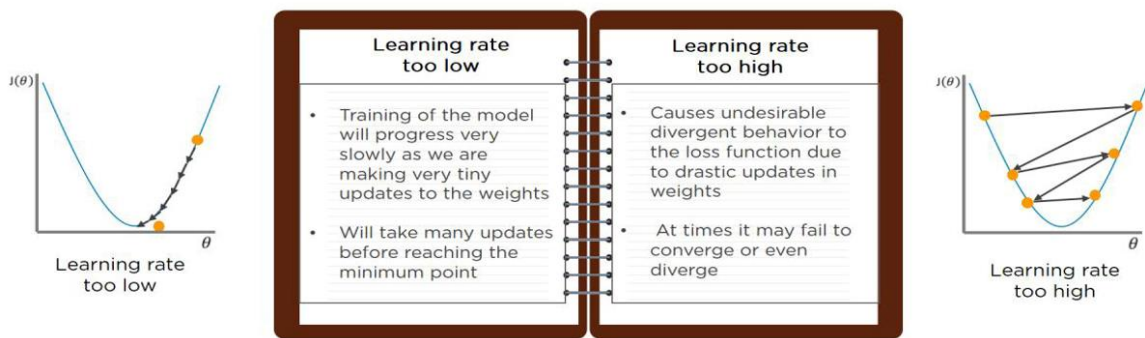
With neural networks, you're usually working with hyperparameters once the data is formatted correctly. A hyperparameter is a parameter whose value is set before the learning process begins. It determines how a network is trained and the structure of the network (such as the number of hidden units, the learning rate, epochs, etc.).



14. What Will Happen If the Learning Rate Is Set Too Low or Too High?

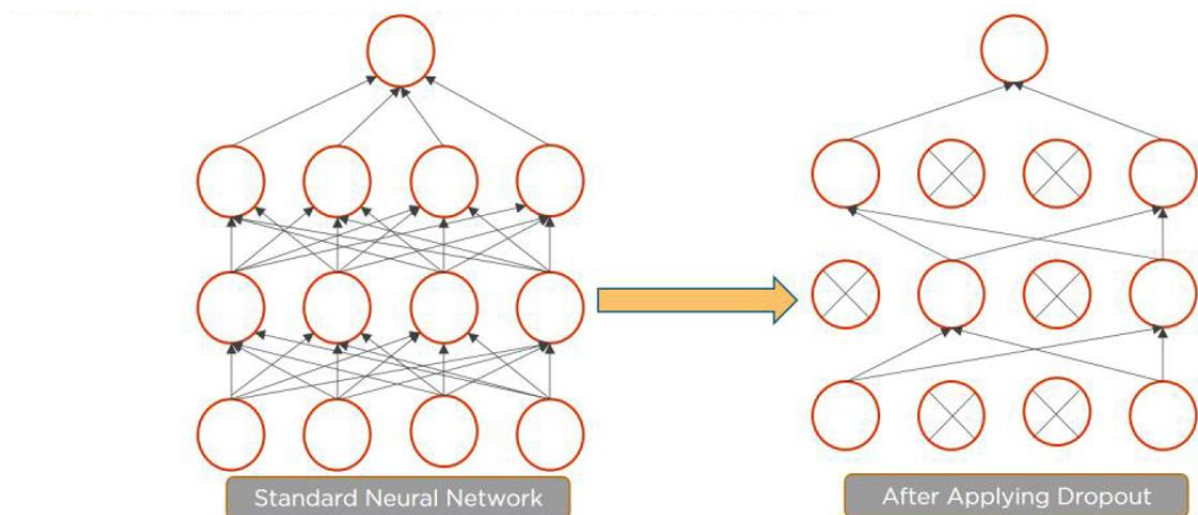
When your learning rate is too low, training of the model will progress very slowly as we are making minimal updates to the weights. It will take many updates before reaching the minimum point.

If the learning rate is set too high, this causes undesirable divergent behavior to the loss function due to drastic updates in weights. It may fail to converge (model can give a good output) or even diverge (data is too chaotic for the network to train).



15. What Is Dropout and Batch Normalization?

Dropout is a technique of dropping out hidden and visible units of a network randomly to prevent overfitting of data (typically dropping 20 percent of the nodes). It doubles the number of iterations needed to converge the network.



Batch normalization is the technique to improve the performance and stability of neural networks by normalizing the inputs in every layer so that they have mean output activation of zero and standard deviation of one.

16. What Is the Difference Between Batch Gradient Descent and Stochastic Gradient Descent?

| | |
|------------------------|-----------------------------|
| Batch Gradient Descent | Stochastic Gradient Descent |
|------------------------|-----------------------------|

| | |
|---|--|
| <p>The batch gradient computes the gradient using the entire dataset.</p> <p>It takes time to converge because the volume of data is huge, and weights update slowly.</p> | <p>The stochastic gradient computes the gradient using a single sample.</p> <p>It converges much faster than the batch gradient because it updates weight more frequently.</p> |
|---|--|

17. What is Overfitting and Underfitting, and How to Combat Them?

Overfitting occurs when the model learns the details and noise in the training data to the degree that it adversely impacts the execution of the model on new information. It is more likely to occur with nonlinear models that have more flexibility when learning a target function. An example would be if a model is looking at cars and trucks, but only recognizes trucks that have a specific box shape. It might not be able to notice a flatbed truck because there's only a particular kind of truck it saw in training. The model performs well on training data, but not in the real world.

Underfitting alludes to a model that is neither well-trained on data nor can generalize to new information. This usually happens when there is less and incorrect data to train a model. Underfitting has both poor performance and accuracy.

To combat overfitting and underfitting, you can resample the data to estimate the model accuracy (k-fold cross-validation) and by having a validation dataset to evaluate the model.

18. How Are Weights Initialized in a Network?

There are two methods here: we can either initialize the weights to zero or assign them randomly.

Initializing all weights to 0: This makes your model similar to a linear model. All the neurons and every layer perform the same operation, giving the same output and making the deep net useless.

Initializing all weights randomly: Here, the weights are assigned randomly by initializing them very close to 0. It gives better accuracy to the model since every neuron performs different computations. This is the most commonly used method.

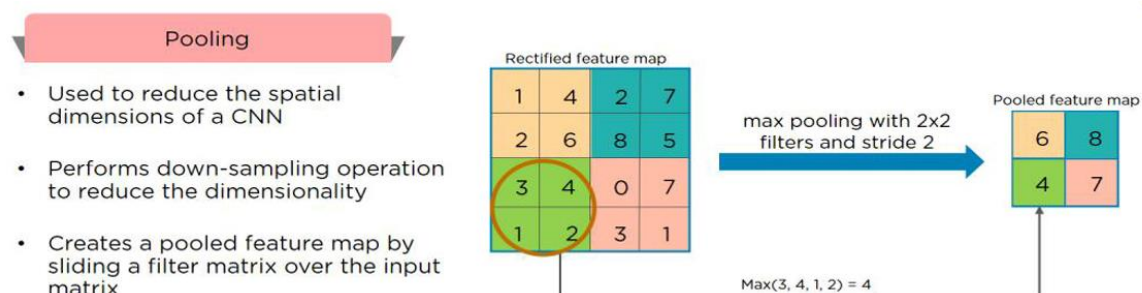
19. What Are the Different Layers on CNN?

There are four layers in CNN:

1. Convolutional Layer - the layer that performs a convolutional operation, creating several smaller picture windows to go over the data.
2. ReLU Layer - it brings non-linearity to the network and converts all the negative pixels to zero. The output is a rectified feature map.
3. Pooling Layer - pooling is a down-sampling operation that reduces the dimensionality of the feature map.
4. Fully Connected Layer - this layer recognizes and classifies the objects in the image.

20. What is Pooling on CNN, and How Does It Work?

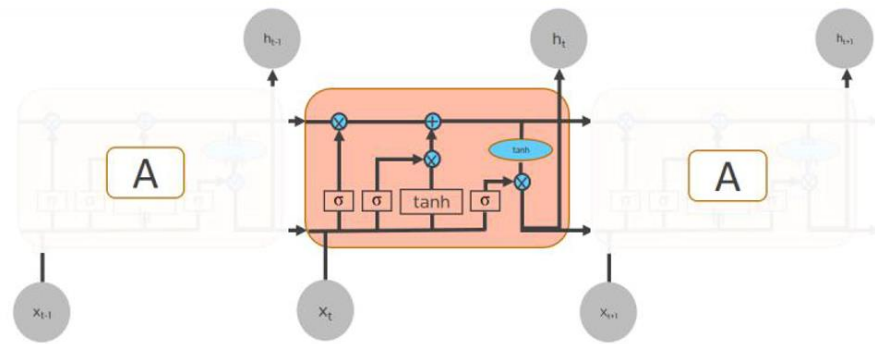
Pooling is used to reduce the spatial dimensions of a CNN. It performs down-sampling operations to reduce the dimensionality and creates a pooled feature map by sliding a filter matrix over the input matrix.



21. How Does an LSTM Network Work?

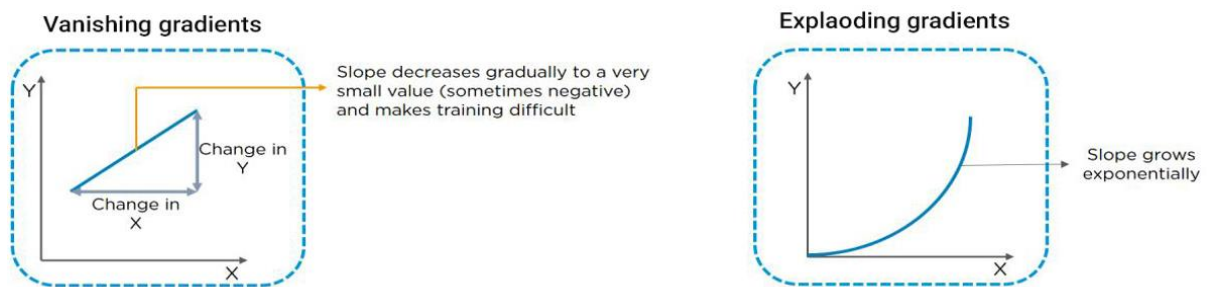
Long-Short-Term Memory (LSTM) is a special kind of recurrent neural network capable of learning long-term dependencies, remembering information for long periods as its default behavior. There are three steps in an LSTM network:

- Step 1: The network decides what to forget and what to remember.
- Step 2: It selectively updates cell state values.
- Step 3: The network decides what part of the current state makes it to the output.



22. What Are Vanishing and Exploding Gradients?

While training an RNN, your slope can become either too small or too large; this makes the training difficult. When the slope is too small, the problem is known as a “Vanishing Gradient.” When the slope tends to grow exponentially instead of decaying, it’s referred to as an “Exploding Gradient.” Gradient problems lead to long training times, poor performance, and low accuracy.



23. What Is the Difference Between Epoch, Batch, and Iteration in Deep Learning?

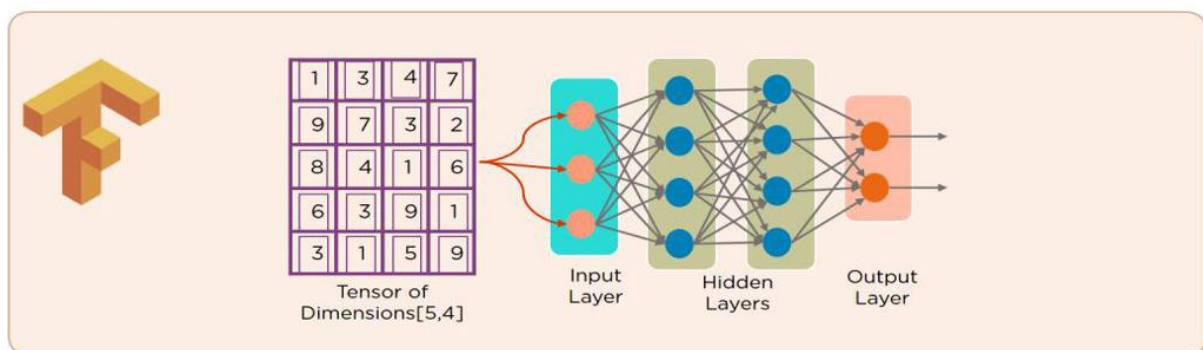
- Epoch - Represents one iteration over the entire dataset (everything put into the training model).
- Batch - Refers to when we cannot pass the entire dataset into the neural network at once, so we divide the dataset into several batches.
- Iteration - if we have 10,000 images as data and a batch size of 200. then an epoch should run 50 iterations (10,000 divided by 50).

24. Why is Tensorflow the Most Preferred Library in Deep Learning?

[Tensorflow](#) provides both C++ and Python APIs, making it easier to work on and has a faster compilation time compared to other Deep Learning libraries like [Keras](#) and [Torch](#). Tensorflow supports both CPU and GPU computing devices.

25. What Do You Mean by Tensor in Tensorflow?

A tensor is a mathematical object represented as arrays of higher dimensions. These arrays of data with different dimensions and ranks fed as input to the neural network are called "Tensors."



26. What Are the Programming Elements in Tensorflow?

Constants - Constants are parameters whose value does not change. To define a constant we use `tf.constant()` command. For example:

```
a = tf.constant(2.0,tf.float32)
```

```
b = tf.constant(3.0)
```

```
Print(a, b)
```

Variables - Variables allow us to add new trainable parameters to graph. To define a variable, we use the `tf.Variable()` command and initialize them before running the graph in a session. An example:

```
W = tf.Variable([.3].dtype=tf.float32)
```

```
b = tf.Variable([- .3].dtype=tf.float32)
```

Placeholders - these allow us to feed data to a tensorflow model from outside a model. It permits a value to be assigned later. To define a placeholder, we use the `tf.placeholder()` command. An example:

```
a = tf.placeholder (tf.float32)

b = a*2

with tf.Session() as sess:

result = sess.run(b,feed_dict={a:3.0})

print result
```

Sessions - a session is run to evaluate the nodes. This is called the "Tensorflow runtime." For example:

```
a = tf.constant(2.0)

b = tf.constant(4.0)

c = a+b

# Launch Session

Sess = tf.Session()

# Evaluate the tensor c

print(sess.run(c))
```

27. Explain a Computational Graph.

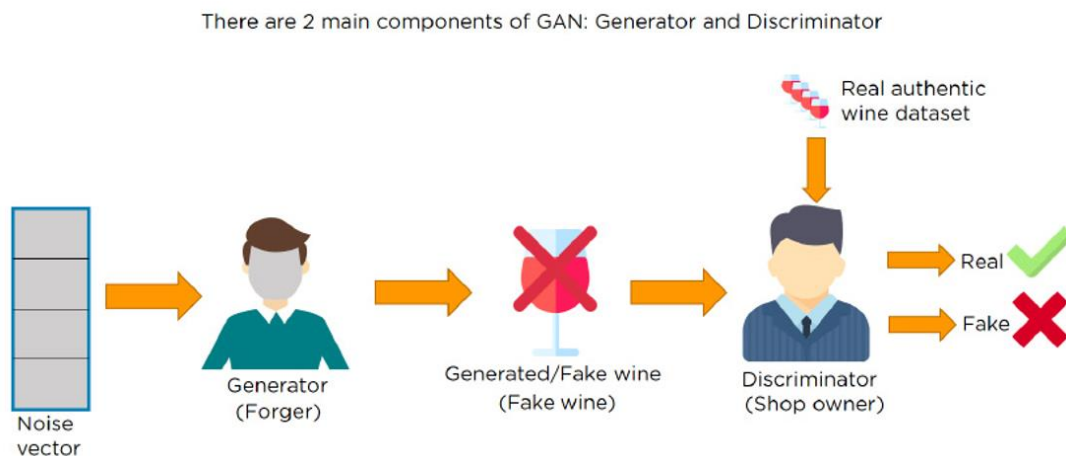
Everything in a tensorflow is based on creating a computational graph. It has a network of nodes where each node operates, Nodes represent mathematical operations, and edges represent tensors. Since data flows in the form of a graph, it is also called a "DataFlow Graph."

28. Explain Generative Adversarial Network.

Suppose there is a wine shop purchasing wine from dealers, which they resell later. But some dealers sell fake wine. In this case, the shop owner should be able to distinguish between fake and authentic wine.

The forger will try different techniques to sell fake wine and make sure specific techniques go past the shop owner's check. The shop owner would probably get some feedback from wine experts that some of the wine is not original. The owner would have to improve how he determines whether a wine is fake or authentic.

The forger's goal is to create wines that are indistinguishable from the authentic ones while the shop owner intends to tell if the wine is real or not accurately.



Let us understand this example with the help of an image shown above.

There is a noise vector coming into the forger who is generating fake wine.

Here the forger acts as a Generator.

The shop owner acts as a Discriminator.

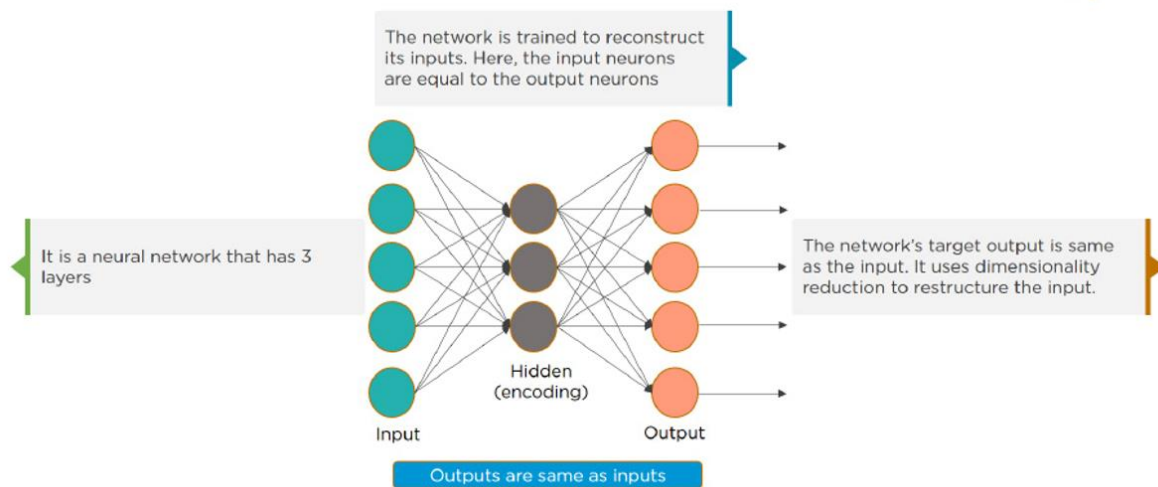
The Discriminator gets two inputs; one is the fake wine, while the other is the real authentic wine. The shop owner has to figure out whether it is real or fake.

So, there are two primary components of Generative Adversarial Network (GAN) named:

1. Generator
2. Discriminator

The generator is a CNN that keeps producing images and is closer in appearance to the real images while the discriminator tries to determine the difference between real and fake images. The ultimate aim is to make the discriminator learn to identify real and fake images.

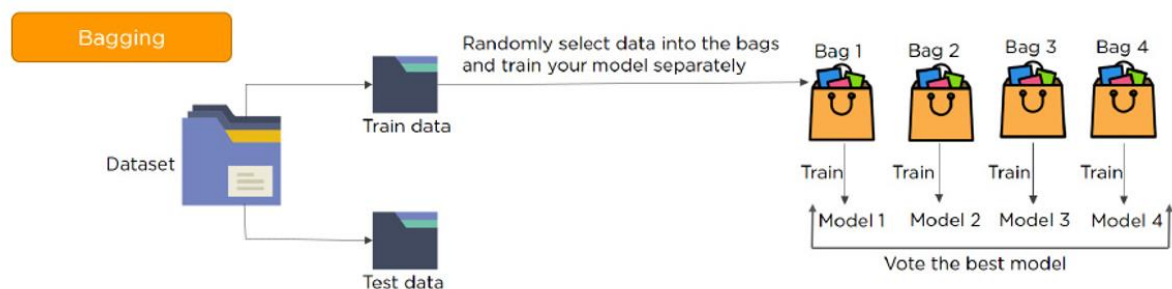
29. What Is an Auto-encoder?



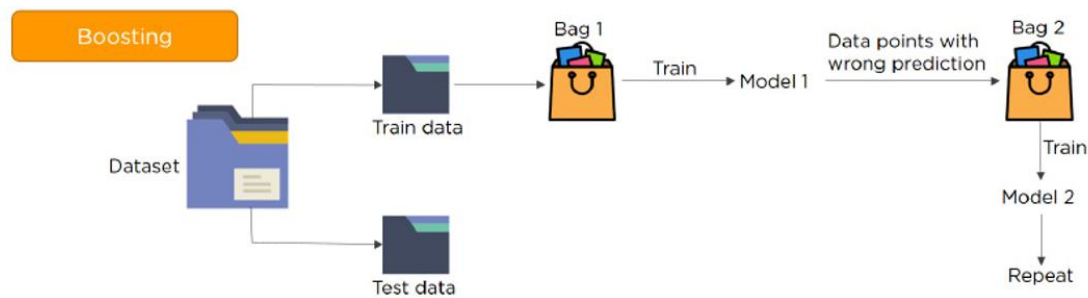
This Neural Network has three layers in which the input neurons are equal to the output neurons. The network's target outside is the same as the input. It uses dimensionality reduction to restructure the input. It works by compressing the image input to a latent space representation then reconstructing the output from this representation.

30. What Is Bagging and Boosting?

Bagging and Boosting are ensemble techniques to train multiple models using the same learning algorithm and then taking a call.



With Bagging, we take a dataset and split it into training data and test data. Then we randomly select data to place into the bags and train the model separately.



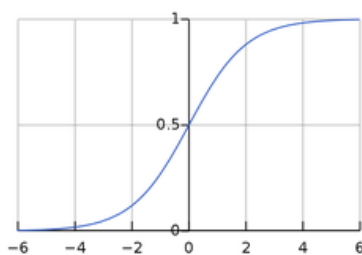
With Boosting, the emphasis is on selecting data points which give wrong output to improve the accuracy.

31. What is the difference between a Perceptron and Logistic Regression?

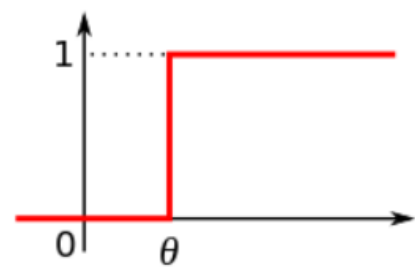
A Multi-Layer Perceptron (MLP) is one of the most basic [neural networks](#) that we use for classification. For a binary classification problem, we know that the output can be either 0 or 1. This is just like our simple logistic regression, where we use a logit function to generate a probability between 0 and 1.

So, what's the difference between the two?

Simply put, it is just the difference in the threshold function! When we restrict the logistic regression model to give us either exactly 1 or exactly 0, we get a Perceptron model:



Logistic regression - Logit function



Perceptron step function

32. Can we have the same bias for all neurons of a hidden layer?

Essentially, you can have a different bias value at each layer or at each neuron as well. However, it is best if we have a bias matrix for all the neurons in the hidden layers as well.

A point to note is that both these strategies would give you very different results.

33. What if we do not use any activation function(s) in a neural network?

The main aim of this question is to understand why we need [activation functions](#) in a neural network. You can start off by giving a simple explanation of how neural networks are built:

Step 1: Calculate the sum of all the inputs (X) according to their weights and include the bias term:

$$Z = (\text{weights} * X) + \text{bias}$$

Step 2: Apply an activation function to calculate the expected output:

$$Y = \text{Activation}(Z)$$

Steps 1 and 2 are performed at each layer. If you recollect, this is nothing but forward propagation! Now, what if there is no activation function?

Our equation for Y essentially becomes:

$$Y = Z = (\text{weights} * X) + \text{bias}$$

Wait – isn't this just a simple linear equation? Yes – and that is why we need activation functions. A linear equation will not be able to capture the complex patterns in the data – this is even more evident in the case of deep learning problems.

In order to capture non-linear relationships, we use activation functions, and that is why a neural network without an activation function is just a linear regression model.

34. In a neural network, what if all the weights are initialized with the same value?

In simplest terms, if all the neurons have the same value of weights, each hidden unit will get exactly the same signal. While this might work during forward propagation, the derivative of the cost function during backward propagation would be the same every time.

In short, there is no learning happening by the network! What do you call the phenomenon of the model being unable to learn any patterns from the data? Yes, [underfitting](#).

Therefore, if all weights have the same initial value, this would lead to underfitting.

Note: This question might further lead to questions on exploding and vanishing gradients, which I have covered below.

35. List the supervised and unsupervised tasks in Deep Learning.

Now, this can be one tricky question. There might be a misconception that deep learning can only solve unsupervised learning problems. This is not the case. Some example of Supervised Learning and Deep learning include:

- Image classification
- Text classification
- Sequence tagging

On the other hand, there are some unsupervised deep learning techniques as well:

- Word embeddings (like Skip-gram and Continuous Bag of Words): [Understanding Word Embeddings: From Word2Vec to Count Vectors](#)
- Autoencoders: [Learn How to Enhance a Blurred Image using an Autoencoder!](#)

Here is a great article on applications of Deep Learning for unsupervised tasks:

- [Essentials of Deep Learning: Introduction to Unsupervised Deep Learning \(with Python codes\)](#)

36. What is the role of weights and bias in a neural network?

This is a question best explained with a real-life example. Consider that you want to go out today to play a cricket match with your friends. Now, a number of factors can affect your decision-making, like:

- How many of your friends can make it to the game?
- How much equipment can all of you bring?
- What is the temperature outside?

And so on. These factors can change your decision greatly or not too much. For example, if it is raining outside, then you cannot go out to play at all. Or if you have only one bat, you can share it while playing as well. The magnitude by which these factors can affect the game is called the weight of that factor.

Factors like the weather or temperature might have a higher weight, and other factors like equipment would have a lower weight.

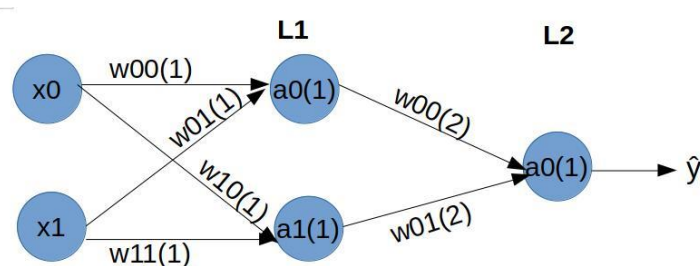
However, does this mean that we can play a cricket match with only one bat? No – we would need 1 ball and 6 wickets as well. This is where bias comes into the picture. Bias lets you assign some threshold which helps you activate a decision-point (or a neuron) only when that threshold is crossed.

37. How does forward propagation and backpropagation work in deep learning?

Now, this can be answered in two ways. If you are on a phone interview, you cannot perform all the calculus in writing and show the interviewer. In such cases, it best to explain it as such:

- **Forward propagation:** The inputs are provided with weights to the hidden layer. At each hidden layer, we calculate the output of the activation at each node and this further propagates to the next layer till the final output layer is reached. Since we start from the inputs to the final output layer, we move forward and it is called forward propagation
- **Backpropagation:** We minimize the cost function by its understanding of how it changes with changing the weights and biases in a neural network. This change is obtained by calculating the gradient at each hidden layer (and using the chain rule). Since we start from the final cost function and go back each hidden layer, we move backward and thus it is called backward propagation

For an in-person interview, it is best to take up the marker, create a simple neural network with 2 inputs, a hidden layer, and an output layer, and explain it.



Forward propagation:

Forward Propagation

Assuming Activation function = Sigmoid(σ)

At Layer L1,

$$z_0^{(1)} = [w_{00}^{(1)} \cdot x_0 + b_{00}^{(1)}] + [w_{01}^{(1)} \cdot x_1 + b_{01}^{(1)}] \quad \text{and}$$

$$z_1^{(1)} = [w_{10}^{(1)} \cdot x_0 + b_{10}^{(1)}] + [w_{11}^{(1)} \cdot x_1 + b_{11}^{(1)}]$$

After applying Activation function at L1,

$$a_0^{(1)} = \sigma(z_0^{(1)}) \quad \quad \quad a_1^{(1)} = \sigma(z_1^{(1)}) \quad |$$

At Layer L2,

$$z_0^{(2)} = [w_{00}^{(2)} \cdot a_0^{(1)} + b_{00}^{(2)}] + [w_{01}^{(2)} \cdot a_1^{(1)} + b_{01}^{(2)}]$$

Final Output Layer,

$$\hat{y} = \sigma(z_0^{(2)}) = a_0^{(2)}$$

Backpropagation:

At layer L2, for all weights:

At Layer L2,

$$\frac{\partial C}{\partial w_{00}^{(2)}} = \frac{\partial C}{\partial a_0^{(2)}} \cdot \frac{\partial a_0^{(2)}}{\partial z_0^{(2)}} \cdot \frac{\partial z_0^{(2)}}{\partial w_{00}^{(2)}} \quad |$$
$$\frac{\partial C}{\partial w_{01}^{(2)}} = \frac{\partial C}{\partial a_0^{(2)}} \cdot \frac{\partial a_0^{(2)}}{\partial z_0^{(2)}} \cdot \frac{\partial z_0^{(2)}}{\partial w_{01}^{(2)}}$$

At layer L1, for all weights:

At Layer L1,

1.

$$\frac{\delta C}{\delta w_{00}^{(1)}} = \frac{\delta C}{\delta a_0^{(1)}} \cdot \frac{\delta a_0^{(1)}}{\delta z_0^{(1)}} \cdot \frac{\delta z_0^{(1)}}{\delta w_{00}^{(1)}} = \left[\frac{\delta C}{\delta a_0^{(2)}} \cdot \frac{\delta a_0^{(2)}}{\delta z_0^{(2)}} \cdot \frac{\delta z_0^{(2)}}{\delta a_0^{(1)}} \right] \cdot \frac{\delta a_0^{(1)}}{\delta z_0^{(1)}} \cdot \frac{\delta z_0^{(1)}}{\delta w_{00}^{(1)}}$$

2.

$$\frac{\delta C}{\delta w_{01}^{(1)}} = \frac{\delta C}{\delta a_0^{(1)}} \cdot \frac{\delta a_0^{(1)}}{\delta z_0^{(1)}} \cdot \frac{\delta z_0^{(1)}}{\delta w_{01}^{(1)}} = \left[\frac{\delta C}{\delta a_0^{(2)}} \cdot \frac{\delta a_0^{(2)}}{\delta z_0^{(2)}} \cdot \frac{\delta z_0^{(2)}}{\delta a_0^{(1)}} \right] \cdot \frac{\delta a_0^{(1)}}{\delta z_0^{(1)}} \cdot \frac{\delta z_0^{(1)}}{\delta w_{01}^{(1)}}$$

3.

$$\frac{\delta C}{\delta w_{10}^{(1)}} = \frac{\delta C}{\delta a_1^{(1)}} \cdot \frac{\delta a_1^{(1)}}{\delta z_1^{(1)}} \cdot \frac{\delta z_1^{(1)}}{\delta w_{10}^{(1)}} = \left[\frac{\delta C}{\delta a_0^{(2)}} \cdot \frac{\delta a_0^{(2)}}{\delta z_0^{(2)}} \cdot \frac{\delta z_0^{(2)}}{\delta a_1^{(1)}} \right] \cdot \frac{\delta a_0^{(1)}}{\delta z_0^{(1)}} \cdot \frac{\delta z_0^{(1)}}{\delta w_{10}^{(1)}}$$

4.

$$\frac{\delta C}{\delta w_{11}^{(1)}} = \frac{\delta C}{\delta a_1^{(1)}} \cdot \frac{\delta a_1^{(1)}}{\delta z_1^{(1)}} \cdot \frac{\delta z_1^{(1)}}{\delta w_{11}^{(1)}} = \left[\frac{\delta C}{\delta a_0^{(2)}} \cdot \frac{\delta a_0^{(2)}}{\delta z_0^{(2)}} \cdot \frac{\delta z_0^{(2)}}{\delta a_1^{(1)}} \right] \cdot \frac{\delta a_0^{(1)}}{\delta z_0^{(1)}} \cdot \frac{\delta z_0^{(1)}}{\delta w_{11}^{(1)}}$$

You need not explain with respect to the bias term as well, though you might need to expand the above equations substituting the actual derivatives.

38. What are the common data structures used in Deep Learning?

Deep Learning goes right from the simplest data structures like lists to complicated ones like computation graphs.

Here are the most common ones:

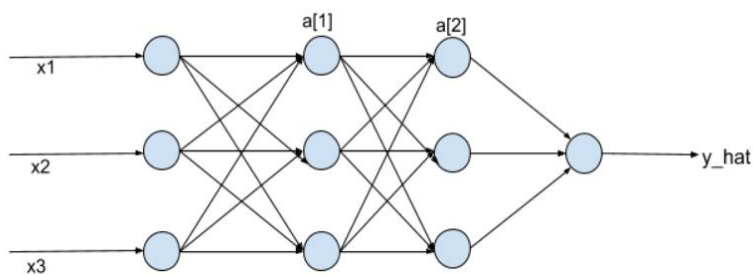
- **List:** An ordered sequence of elements (You can also mention NumPy ndarrays here)
- **Matrix:** An ordered sequence of elements with rows and columns
- **Dataframe:** A dataframe is just like a matrix, but it holds actual data with the column names and rows denoting each datapoint in your dataset. If marks of 100 students, their grades, and their details are stored in a dataframe, their details are stored as columns. Each row will represent the data of each of the 100 students
- **Tensors:** You will work with them on a daily basis if you have ventured into deep learning. Used both in PyTorch and TensorFlow, tensors are like the basic programming unit of deep learning. Just like multidimensional arrays, we can perform numerous mathematical operations on them. Read more about tensors [here](#)
- **Computation Graphs:** Since deep learning involves multiple layers and often hundreds, if not thousands of parameters, it is important to understand the flow of computation. A computation graph is just that. A computation graph gives us

the sequence of operations performed with each node denoting an operation or a component in the neural network

39. Why should we use Batch Normalization?

Once the interviewer has asked you about the fundamentals of deep learning architectures, they would move on to the key topic of improving your deep learning model's performance.

Batch Normalization is one of the techniques used for reducing the training time of our deep learning algorithm. Just like normalizing our input helps improve our logistic regression model, we can normalize the activations of the hidden layers in our deep learning model as well:



We basically normalize $a[1]$ and $a[2]$ here. This means we normalize the inputs to the layer, and then apply the activation functions to the normalized inputs.

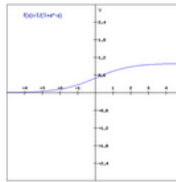
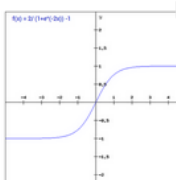
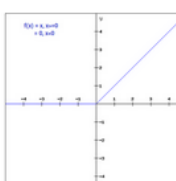
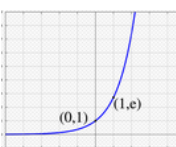
Here is an article that explains Batch Normalization and other techniques for improving Neural Networks: [Neural Networks – Hyperparameter Tuning, Regularization & Optimization](#).

40. List the activation functions you have used so far in your projects and how you would choose one.

The most common activation functions are:

- Sigmoid
- Tanh
- ReLU
- Softmax

While it is not important to know all the activation functions, you can always score points by knowing the range of these functions and how they are used. Here is a handy table for you to follow:

| Function | Mathematical Expression | Range | Plot |
|----------|---|----------|--|
| Sigmoid | $\frac{1}{1 + e^{-x}}$ | (0, 1) |  |
| tanh | $2 * \text{sigmoid}(2x) - 1$ | (-1, 1) |  |
| ReLU | $\max(0, x)$ | [0, inf) |  |
| Softmax | $s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$ | [0, 1] |  |

Here is a great guide on how to use these and other activations functions: [Fundamentals of Deep Learning – Activation Functions and When to Use Them?](#)

41. Why does a Convolutional Neural Network (CNN) work better with image data?

The key to this question lies in the Convolution operation. Unlike humans, the machine sees the image as a matrix of pixel values. Instead of interpreting a shape like a petal or an ear, it just identifies curves and edges.

Thus, instead of looking at the entire image, it helps to just read the image in parts. Doing this for a 300 x 300 pixel image would mean dividing the matrix into smaller 3 x 3 matrices and dealing with them one by one. This is convolution.

Mathematically, we just perform a small operation on the matrix to help us detect features in the image – like boundaries, colors, etc.

$$Z = X * f$$

Here, we are convolving (* operation – not multiplication) the input matrix X with another small matrix f, called the kernel/filter to create a new matrix Z. This matrix is then passed on to the other layers.

If you have a board/screen in front of you, you can always illustrate this with a simple example:

| | | |
|----|----|---|
| 3 | 9 | 4 |
| 11 | 1 | 8 |
| 2 | 13 | 7 |

| | |
|---|---|
| 0 | 0 |
| 1 | 1 |

Thus, the filter 'f' considers 2 X 2 subparts of the X matrix at the time and performs the convolution operation

- $(3 \times 0) + (9 \times 0) + (11 \times 1) + (1 \times 1) = 12$
- $(9 \times 0) + (4 \times 0) + (1 \times 1) + (8 \times 1) = 9$
- $(11 \times 0) + (1 \times 0) + (2 \times 1) + (13 \times 1) = 15$
- $(1 \times 0) + (8 \times 0) + (13 \times 1) + (7 \times 1) = 20$

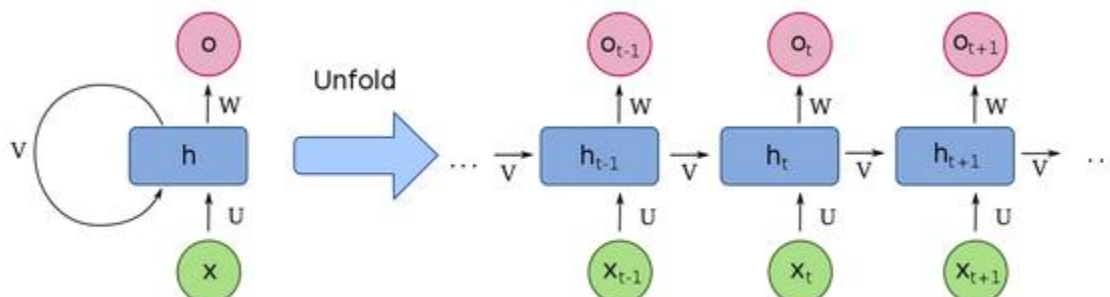
Thus, Z =

| | |
|----|----|
| 12 | 9 |
| 15 | 20 |

Learning more about how CNNs work [here](#).

42. Why do RNNs work better with text data?

The main component that differentiates Recurrent Neural Networks (RNN) from the other models is the addition of a loop at each node. This loop brings the **recurrence** mechanism in RNNs. In a basic Artificial Neural Network (ANN), each input is given the same weight and fed to the network at the same time. So, for a sentence like "I saw the movie and hated it", it would be difficult to capture the information which associates "it" with the "movie".



The addition of a loop is to denote preserving the previous node's information for the next node, and so on. This is why RNNs are much better for sequential data, and since text data also is sequential in nature, they are an improvement over ANNs.

43. In a CNN, if the input size 5 X 5 and the filter size is 7 X 7, then what would be the size of the output?

This is a pretty intuitive answer. As we saw above, we perform the convolution on 'x' one step at a time, to the right, and in the end, we got Z with dimensions 2 X 2, for X with dimensions 3 X 3.

Thus, to make the input size similar to the filter size, we make use of padding – adding 0s to the input matrix such that its new size becomes at least 7 X 7. Thus, the output size would be using the formula:

$$\text{Dimension of image} = (n, n) = 5 \times 5$$

$$\text{Dimension of filter} = (f, f) = 7 \times 7$$

$$\text{Padding} = 1 \text{ (adding 1 pixel with value 0 all around the edges)}$$

$$\text{Dimension of output will be } (n+2p-f+1) \times (n+2p-f+1) = 1 \times 1$$

44. What's the difference between valid and same padding in a CNN?

This question has more chances of being a follow-up question to the previous one. Or if you have explained how you used CNNs in a computer vision task, the interviewer might ask this question along with the details of the padding parameters.

- Valid Padding: When we do not use any padding. The resultant matrix after convolution will have dimensions $(n - f + 1) \times (n - f + 1)$
- Same padding: Adding padded elements all around the edges such that the output matrix will have the same dimensions as that of the input matrix

45. What do you mean by exploding and vanishing gradients?

The key here is to make the explanation as simple as possible. As we know, the [gradient descent algorithm](#) tries to minimize the error by taking small steps towards the minimum value. These steps are used to update the weights and biases in a neural network.

However, at times, the steps become too large and this results in larger updates to weights and bias terms – so much so as to cause an overflow (or a NaN) value in the weights. This leads to an unstable algorithm and is called an exploding gradient.

On the other hand, the steps are too small and this leads to minimal changes in the weights and bias terms – even negligible changes at times. We thus might end up training a deep learning model with almost the same weights and biases each time and never reach the minimum error function. This is called the vanishing gradient.

A point to note is that both these issues are specifically evident in Recurrent Neural Networks – so be prepared for follow-up questions on RNN!

46. What are the applications of transfer learning in Deep Learning?

I am sure you would have a doubt as to why a relatively simple question was included in the Intermediate Level. The reason is the sheer volume of subsequent questions it can generate!

The use of [transfer learning](#) has been one of the key milestones in deep learning. Training a large model on a huge dataset, and then using the final parameters on smaller simpler datasets has led to defining breakthroughs in the form of Pretrained Models. Be it Computer Vision or NLP, pretrained models have become the norm in research and in the industry.

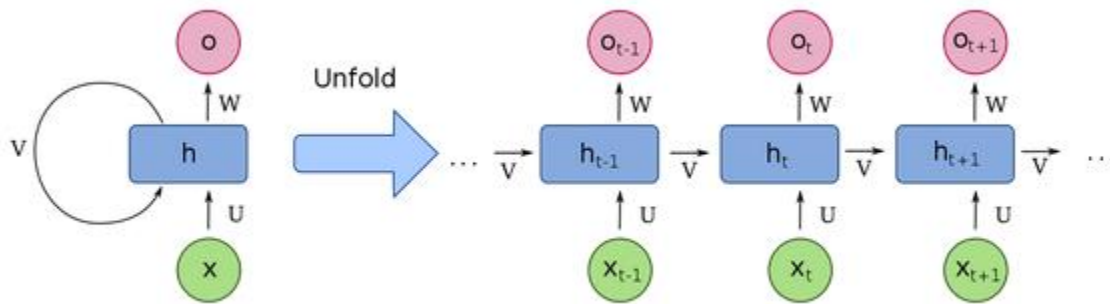
Some popular examples include BERT, ResNet, GPT-2, VGG-16, etc and many more.

It is here that you can earn brownie points by pointing out specific examples/projects where you used these models and how you used them as well.

It is not possible to discuss all of them, so here are a few resources to get started:

47. How backpropagation is different in RNN compared to ANN?

In Recurrent Neural Networks, we have an additional loop at each node:



This loop essentially includes a time component into the network as well. This helps in capturing sequential information from the data, which could not be possible in a generic artificial neural network.

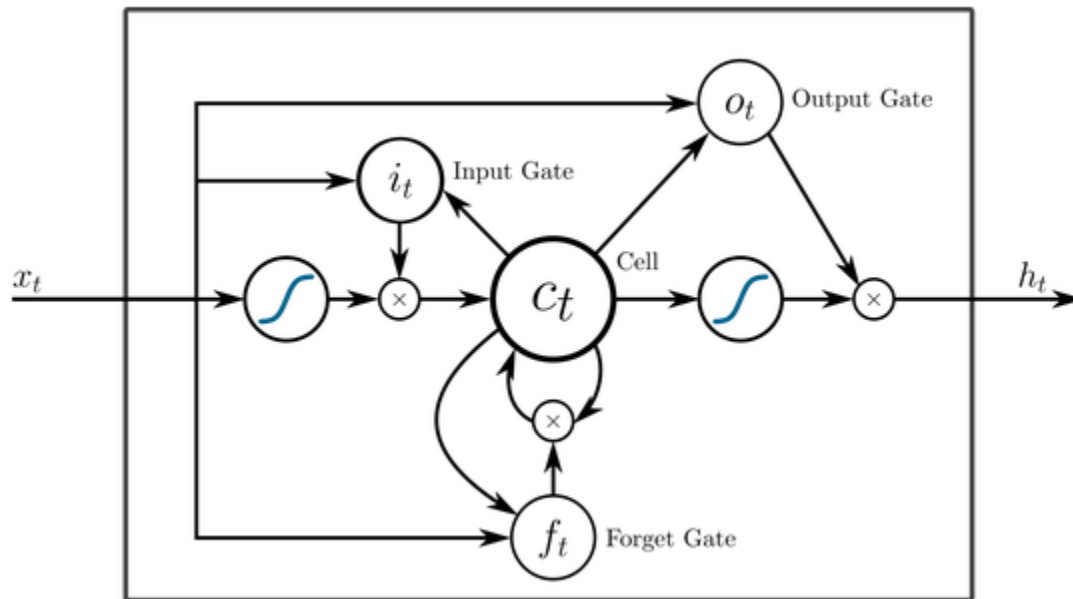
This is why the backpropagation in RNN is called Backpropagation through Time, as in backpropagation at each time step.

You can find a detailed explanation of RNNs here: [Fundamentals of Deep Learning – Introduction to Recurrent Neural Networks](#).

48. How does LSTM solve the vanishing gradient challenge?

The [LSTM model](#) is considered a special case of RNNs. The problems of vanishing gradients and exploding gradients we saw earlier are a disadvantage while using the plain RNN model.

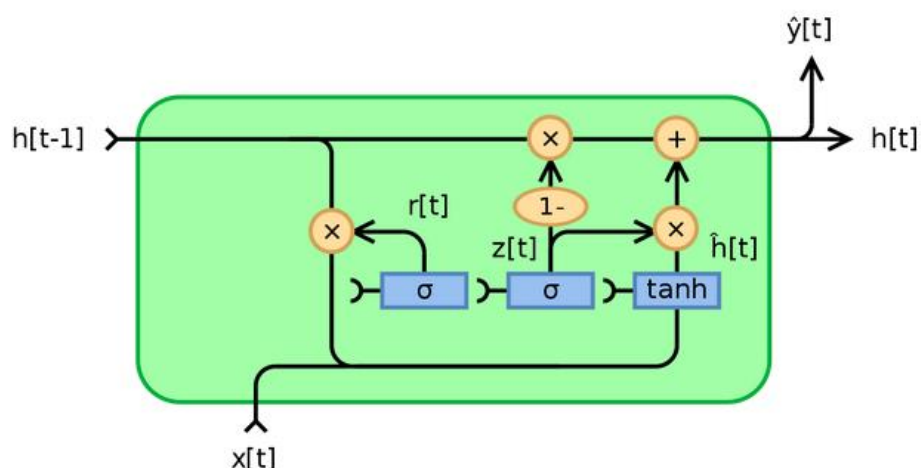
In LSTMs, we add a forget gate, which is basically a memory unit that retains information that is retained across timesteps and discards the other information that is not needed. This also necessitates the need for input and output gates to include the results of the forget gate as well.



49. Why is GRU faster as compared to LSTM?

As you can see, the LSTM model can become quite complex. In order to still retain the functionality of retaining information across time and yet not make a too complex model, we need GRUs.

Basically, in GRUs, instead of having an additional Forget gate, we combine the input and Forget gates into a single Update Gate:



It is this reduction in the number of gates that makes GRU less complex and faster than LSTM. You can learn about GRUs, LSTMs and other sequence models in detail here: [Must-Read Tutorial to Learn Sequence Modeling & Attention Models](#).

50. How is the transformer architecture better than RNN?

Advancements in deep learning have made it possible to solve many tasks in Natural Language Processing. Networks/Sequence models like RNNs, LSTMs, etc. are specifically used for this purpose – so as to capture all possible information from a given sentence, or a paragraph. However, sequential processing comes with its caveats:

- It requires high processing power
- It is difficult to execute in parallel because of its sequential nature

This gave rise to the Transformer architecture. Transformers use what is called the attention mechanism. This basically means mapping dependencies between all the parts of a sentence.

Here is an excellent article explaining transformers: [How do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models.](#)

51. Describe a project you worked on and the tools/frameworks you used?

Now, this is one question that is sure to be asked even if none of the above ones is asked in your deep learning interview. I have included it in the advanced section since you might be grilled on each and every part of the code you have written.

Before the interview, make sure to:

- have your GitHub code updated with the latest code changes you have made
- be ready to give in-depth explanations on at least 2-3 projects where you used deep learning

When you are asked such a question, it is best to give a small 30-second pitch on what was the:

- problem statement
- data you used and the framework (like PyTorch or TensorFlow)
- any pretrained model you used or just the name of the basic model you built upon
- the value of the evaluation metric you achieved

After this, you can start going into detail about the model architecture, what preprocessing steps you had to take, and how that changed the data.

An important point to be noted is that the project need not be a very complicated or sophisticated one. A well-explained object detection project would earn you more points

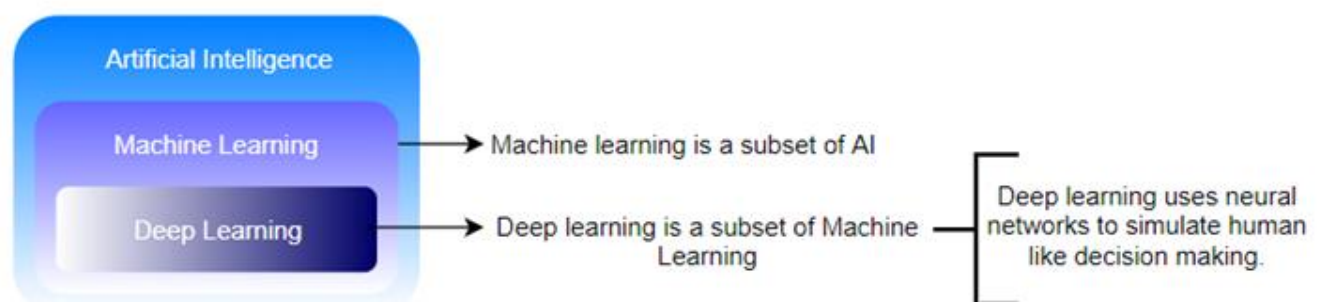
than a poorly-explained video classification project. Towards this end, I recommend having a README file in the above format for every project that you have implemented.

52) What is deep learning?

Deep learning is a part of machine learning with an algorithm inspired by the structure and function of the brain, which is called an **artificial neural network**. In the mid-1960s, **Alexey Grigorevich Ivakhnenko** published the first general, while working on deep learning network. Deep learning is suited over a range of fields such as computer vision, speech recognition, natural language processing, etc.

53) What are the main differences between AI, Machine Learning, and Deep Learning?

- AI stands for Artificial Intelligence. It is a technique which enables machines to mimic human behavior.
- Machine Learning is a subset of AI which uses statistical methods to enable machines to improve with experiences.



- Deep learning is a part of Machine learning, which makes the computation of multi-layer neural networks feasible. It takes advantage of neural networks to simulate human-like decision making.
-

54) Differentiate supervised and unsupervised deep learning procedures.

- Supervised learning is a system in which both input and desired output data are provided. Input and output data are labeled to provide a learning basis for future data processing.

- Unsupervised procedure does not need labeling information explicitly, and the operations can be carried out without the same. The common unsupervised learning method is **cluster analysis**. It is used for exploratory data analysis to find hidden patterns or grouping in data.
-

55) What are the applications of deep learning?

There are various applications of deep learning:

- Computer vision
 - Natural language processing and pattern recognition
 - Image recognition and processing
 - Machine translation
 - Sentiment analysis
 - Question Answering system
 - Object Classification and Detection
 - Automatic Handwriting Generation
 - Automatic Text Generation.
-

56) Do you think that deep network is better than a shallow one?

Both shallow and deep networks are good enough and capable of approximating any function. But for the same level of accuracy, deeper networks can be much more efficient in terms of computation and number of parameters. Deeper networks can create deep representations. At every layer, the network learns a new, more abstract representation of the input.

57) What do you mean by "overfitting"?

Overfitting is the most common issue which occurs in deep learning. It usually occurs when a deep learning algorithm apprehends the sound of specific data. It also appears when the particular algorithm is well suitable for the data and shows up when the algorithm or model represents high variance and low bias.

58) What is Backpropagation?

Backpropagation is a training algorithm which is used for multilayer neural networks. It transfers the error information from the end of the network to all the weights inside the network. It allows the efficient computation of the gradient.

Backpropagation can be divided into the following steps:

- It can forward propagation of training data through the network to generate output.
 - It uses target value and output value to compute error derivative concerning output activations.
 - It can backpropagate to compute the derivative of the error concerning output activations in the previous layer and continue for all hidden layers.
 - It uses the previously calculated derivatives for output and all hidden layers to calculate the error derivative concerning weights.
 - It updates the weights.
-

59) What is the function of the Fourier Transform in Deep Learning?

Fourier transform package is highly efficient for analyzing, maintaining, and managing a large databases. The software is created with a high-quality feature known as the **special portrayal**. One can effectively utilize it to generate real-time array data, which is extremely helpful for processing all categories of signals.

60) Describe the theory of autonomous form of deep learning in a few words.

There are several forms and categories available for the particular subject, but the autonomous pattern represents independent or unspecified mathematical bases which are free from any specific categorizer or formula.

62) What is the use of Deep learning in today's age, and how is it adding data scientists?

Deep learning has brought significant changes or revolution in the field of machine learning and data science. The concept of a **complex neural network** (CNN) is the main center of attention for data scientists. It is widely taken because of its

advantages in performing next-level machine learning operations. The advantages of deep learning also include the process of clarifying and simplifying issues based on an algorithm due to its utmost flexible and adaptable nature. It is one of the rare procedures which allow the movement of data in independent pathways. Most of the data scientists are viewing this particular medium as an advanced additive and extended way to the existing process of machine learning and utilizing the same for solving complex day to day issues.

63) What are the deep learning frameworks or tools?

Deep learning frameworks or tools are:

Tensorflow, Keras, Chainer, Pytorch, Theano & Ecosystem, Caffe2, CNTK, DyNetGensim, DSSTNE, Gluon, Paddle, Mxnet, BigDL

64) What are the disadvantages of deep learning?

There are some disadvantages of deep learning, which are:

- Deep learning model takes longer time to execute the model. In some cases, it even takes several days to execute a single model depends on complexity.
 - The deep learning model is not good for small data sets, and it fails here.
-

65) What is the meaning of term weight initialization in neural networks?

In neural networking, weight initialization is one of the essential factors. A bad weight initialization prevents a network from learning. On the other side, a good weight initialization helps in giving a quicker convergence and a better overall error. Biases can be initialized to zero. The standard rule for setting the weights is to be close to zero without being too small.

66) Explain Data Normalization.

Data normalization is an essential preprocessing step, which is used to rescale values to fit in a specific range. It assures better convergence during backpropagation. In general, data normalization boils down to subtracting the mean of each data point and dividing by its standard deviation.

67) Why is zero initialization not a good weight initialization process?

If the set of weights in the network is put to a zero, then all the neurons at each layer will start producing the same output and the same gradients during backpropagation.

As a result, the network cannot learn at all because there is no source of asymmetry between neurons. That is the reason why we need to add randomness to the weight initialization process.

68) What are the prerequisites for starting in Deep Learning?

There are some basic requirements for starting in Deep Learning, which are:

- Machine Learning
 - Mathematics
 - Python Programming
-

69) What are the supervised learning algorithms in Deep learning?

- Artificial neural network
 - Convolution neural network
 - Recurrent neural network
-

70) What are the unsupervised learning algorithms in Deep learning?

- Self Organizing Maps
 - Deep belief networks (Boltzmann Machine)
 - Auto Encoders
-

71) How many layers in the neural network?

- **Input Layer**

The input layer contains input neurons which send information to the hidden layer.

- **Hidden Layer**

The hidden layer is used to send data to the output layer.

- **Output Layer**

The data is made available at the output layer.

72) What is the use of the Activation function?

The activation function is used to introduce nonlinearity into the neural network so that it can learn more complex function. Without the Activation function, the neural network would be only able to learn function, which is a linear combination of its input data.

Activation function translates the inputs into outputs. The activation function is responsible for deciding whether a neuron should be activated or not. It makes the decision by calculating the weighted sum and further adding bias with it. The basic purpose of the activation function is to introduce non-linearity into the output of a neuron.

73) How many types of activation function are available?

- Binary Step
 - Sigmoid
 - Tanh
 - ReLU
 - Leaky ReLU
 - Softmax
 - Swish
-

74) What is a binary step function?

The binary step function is an activation function, which is usually based on a threshold. If the input value is above or below a particular threshold limit, the neuron is activated, then it sends the same signal to the next layer. This function does not allow multi-value outputs.

75) What is the sigmoid function?

The sigmoid activation function is also called the logistic function. It is traditionally a trendy activation function for neural networks. The input data to the function is transformed into a value between **0.0** and **1.0**. Input values that are much larger than 1.0 are transformed to the value 1.0. Similarly, values that are much smaller than 0.0 are transformed into 0.0. The shape of the function for all possible inputs is an S-shape from zero up through 0.5 to 1.0. It was the default activation used on neural networks, in the early 1990s.

76) What is Tanh function?

The hyperbolic tangent function, also known as tanh for short, is a similar shaped nonlinear activation function. It provides output values between **-1.0** and **1.0**. Later in the 1990s and through the 2000s, this function was preferred over the sigmoid activation function as models. It was easier to train and often had better predictive performance.

77) What is ReLU function?

A node or unit which implements the activation function is referred to as a **rectified linear activation unit** or ReLU for short. Generally, networks that use the rectifier function for the hidden layers are referred to as **rectified networks**.

Adoption of ReLU may easily be considered one of the few milestones in the deep learning revolution.

78) What is the use of leaky ReLU function?

The Leaky ReLU (LReLU or LReL) manages the function to allow small negative values when the input is less than zero.

79) What is the softmax function?

The softmax function is used to calculate the probability distribution of the event over 'n' different events. One of the main advantages of using softmax is the output probabilities range. The range will be between 0 to 1, and the sum of all the probabilities will be equal to one. When the softmax function is used for multi-classification model, it returns the probabilities of each class, and the target class will have a high probability.

80) What is a Swish function?

Swish is a new, self-gated activation function. Researchers at Google discovered the Swish function. According to their paper, it performs better than ReLU with a similar level of computational efficiency.

81) What is the most used activation function?

Relu function is the most used activation function. It helps us to solve vanishing gradient problems.

82) Can Relu function be used in output layer?

No, Relu function has to be used in hidden layers.

83) In which layer softmax activation function used?

Softmax activation function has to be used in the output layer.

84) What do you understand by Autoencoder?

Autoencoder is an artificial neural network. It can learn representation for a set of data without any supervision. The network automatically learns by copying its input to the output; typically, internal representation consists of smaller dimensions than the input vector. As a result, they can learn efficient ways of representing the data. Autoencoder consists of two parts; an encoder tries to fit the inputs to the internal representation, and a decoder converts the internal state to the outputs.

85) What do you mean by Dropout?

Dropout is a cheap regulation technique used for reducing overfitting in neural networks. We randomly drop out a set of nodes at each training step. As a result, we create a different model for each training case, and all of these models share weights. It's a form of model averaging.

86) What do you understand by Tensors?

Tensors are nothing but a de facto for representing the data in deep learning. They are just multidimensional arrays, which allows us to represent the data having higher dimensions. In general, we deal with high dimensional data sets where dimensions refer to different features present in the data set.

87) What do you understand by Boltzmann Machine?

A Boltzmann machine (also known as stochastic Hopfield network with hidden units) is a type of recurrent neural network. In a Boltzmann machine, nodes make binary decisions with some bias. Boltzmann machines can be strung together to create more sophisticated systems such as deep belief networks. Boltzmann Machines can be used to optimize the solution to a problem.

Some important points about Boltzmann Machine-

- It uses a recurrent structure.
 - It consists of stochastic neurons, which include one of the two possible states, either 1 or 0.
 - The neurons present in this are either in an adaptive state (free state) or clamped state (frozen state).
 - If we apply simulated annealing or discrete Hopfield network, then it would become a Boltzmann Machine.
-

88) What is Model Capacity?

The capacity of a deep learning neural network controls the scope of the types of mapping functions that it can learn. Model capacity can approximate any given function. When there is a higher model capacity, it means that the larger amount of information can be stored in the network.

89) What is the cost function?

A cost function describes us how well the neural network is performing with respect to its given training sample and the expected output. It may depend on variables such as weights and biases. It provides the performance of a neural network as a whole. In deep learning, our priority is to minimize the cost function. That's why we prefer to use the concept of gradient descent.

90) Explain gradient descent?

An optimization algorithm that is used to minimize some function by repeatedly moving in the direction of steepest descent as specified by the negative of the gradient is known as gradient descent. It's an iteration algorithm, in every iteration algorithm, we compute the gradient of a cost function, concerning each parameter and update the parameter of the function via the following formula:

$$\Theta := \Theta - \alpha \frac{d}{d\Theta} J(\Theta)$$

Where,

Θ - is the parameter vector,

α - learning rate,

$J(\Theta)$ - is a cost function

In machine learning, it is used to update the parameters of our model. Parameters represent the coefficients in linear regression and weights in neural networks.

91) Explain the following variant of Gradient Descent: Stochastic, Batch, and Mini-batch?

- **Stochastic Gradient Descent**

Stochastic gradient descent is used to calculate the gradient and update the parameters by using only a single training example.

- **Batch Gradient Descent**

Batch gradient descent is used to calculate the gradients for the whole dataset and perform just one update at each iteration.

- **Mini-batch Gradient Descent**

Mini-batch gradient descent is a variation of stochastic gradient descent. Instead of a single training example, mini-batch of samples is used. Mini-batch gradient descent is one of the most popular optimization algorithms.

92) What are the main benefits of Mini-batch Gradient Descent?

- It is computationally efficient compared to stochastic gradient descent.
 - It improves generalization by finding flat minima.
 - It improves convergence by using mini-batches. We can approximate the gradient of the entire training set, which might help to avoid local minima.
-

93) What is matrix element-wise multiplication? Explain with an example.

Element-wise matrix multiplication is used to take two matrices of the same dimensions. It further produces another combined matrix with the elements that are a product of corresponding elements of matrix a and b.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \circ \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} =$$

$$\begin{pmatrix} a_{11}b_{11} & a_{12}b_{12} & a_{13}b_{13} \\ a_{21}b_{21} & a_{22}b_{22} & a_{23}b_{23} \\ a_{31}b_{31} & a_{32}b_{32} & a_{33}b_{33} \end{pmatrix}$$

94) What do you understand by a convolutional neural network?

A convolutional neural network, often called CNN, is a feedforward neural network. It uses convolution in at least one of its layers. The convolutional layer contains a set of filter (kernels). This filter is sliding across the entire input image, computing the dot product between the weights of the filter and the input image. As a result of training, the network automatically learns filters that can detect specific features.

95) Explain the different layers of CNN.

There are four layered concepts that we should understand in CNN (Convolutional Neural Network):

- **Convolution**
This layer comprises of a set of independent filters. All these filters are

initialized randomly. These filters then become our parameters which will be learned by the network subsequently.

- **ReLU**

The ReLU layer is used with the convolutional layer.

- **Pooling**

It reduces the spatial size of the representation to lower the number of parameters and computation in the network. This layer operates on each feature map independently.

- **Full Connectedness**

Neurons in a completely connected layer have complete connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can be easily computed with a matrix multiplication followed by a bias offset.

96) What is an RNN?

RNN stands for Recurrent Neural Networks. These are the artificial neural networks which are designed to recognize patterns in sequences of data such as handwriting, text, the spoken word, genomes, and numerical time series data. RNN use backpropagation algorithm for training because of their internal memory. RNN can remember important things about the input they received, which enables them to be very precise in predicting what's coming next.

97) What are the issues faced while training in Recurrent Networks?

Recurrent Neural Network uses backpropagation algorithm for training, but it is applied on every timestamp. It is usually known **as Back-propagation Through Time (BTT)**.

There are two significant issues with Back-propagation, such as:

- **Vanishing Gradient**

When we perform Back-propagation, the gradients tend to get smaller and smaller because we keep on moving backward in the Network. As a result, the neurons in the earlier layer learn very slowly if we compare it with the neurons in the later layers. Earlier layers are more valuable because they are responsible for learning and detecting simple patterns. They are the building

blocks of the network.

If they provide improper or inaccurate results, then how can we expect the next layers and complete network to perform nicely and provide accurate results. The training procedure takes long, and the prediction accuracy of the model decreases.

- **Exploding Gradient**

Exploding gradients are the main problem when large error gradients accumulate. They provide result in very large updates to neural network model weights during training.

Gradient Descent process works best when updates are small and controlled. When the magnitudes of the gradient accumulate, an unstable network is likely to occur. It can cause poor prediction of results or even a model that reports nothing useful.

98) Explain the importance of LSTM.

LSTM stands for **Long short-term memory**. It is an artificial RNN (Recurrent Neural Network) architecture, which is used in the field of deep learning. LSTM has feedback connections which makes it a "general purpose computer." It can process not only a single data point but also entire sequences of data.

They are a special kind of RNN which are capable of learning long-term dependencies.

99) What are the different layers of Autoencoders? Explain briefly.

An autoencoder contains three layers:

- **Encoder**

The encoder is used to compress the input into a latent space representation. It encodes the input images as a compressed representation in a reduced dimension. The compressed images are the distorted version of the original image.

- **Code**

The code layer is used to represent the compressed input which is fed to the decoder.

- **Decoder**

The decoder layer decodes the encoded image back to its original dimension. The decoded image is a reduced reconstruction of the original image. It is automatically reconstructed from the latent space representation.

100) What do you understand by Deep Autoencoders?

Deep Autoencoder is the extension of the simple Autoencoder. The first layer present in DeepAutoencoder is responsible for first-order functions in the raw input. The second layer is responsible for second-order functions corresponding to patterns in the appearance of first-order functions. Deeper layers which are available in the Deep Autoencoder tend to learn even high-order features.

A deep autoencoder is the combination of two, symmetrical deep-belief networks:

- First four or five shallow layers represent the encoding half.
 - The other combination of four or five layers makes up the decoding half.
-

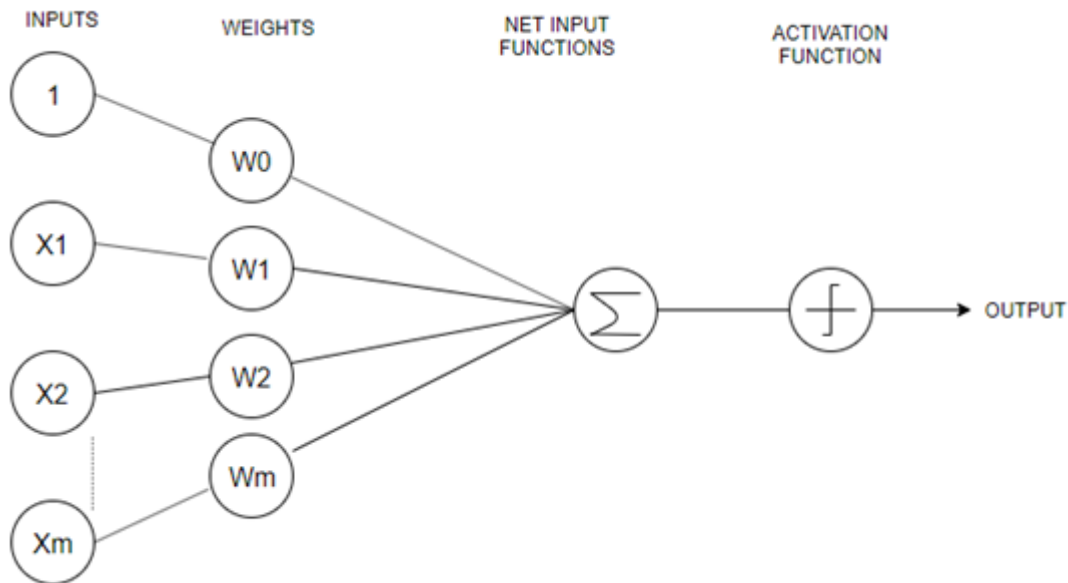
102) What are the three steps to developing the necessary assumption structure in Deep learning?

The procedure of developing an assumption structure involves three specific actions.

- The first step contains algorithm development. This particular process is lengthy.
 - The second step contains algorithm analyzing, which represents the in-process methodology.
 - The third step is about implementing the general algorithm in the final procedure. The entire framework is interlinked and required for throughout the process.
-

103) What do you understand by Perceptron? Also, explain its type.

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features. It is an algorithm for supervised learning of binary classifiers. This algorithm is used to enable neurons to learn and processes elements in the training set one at a time.



There are two types of perceptrons:

- **Single-Layer Perceptron**
Single layer perceptrons can learn only linearly separable patterns.
- **Multilayer Perceptrons**
Multilayer perceptrons or feedforward neural networks with two or more layers have the higher processing power.

105. Why is it necessary to introduce non-linearities in a neural network?

Solution: otherwise, we would have a composition of linear functions, which is also a linear function, giving a linear model. A linear model has a much smaller number of parameters, and is therefore limited in the complexity it can model.

106. Describe two ways of dealing with the vanishing gradient problem in a neural network.

Solution:

- Using ReLU activation instead of sigmoid.
- Using Xavier initialization.

107. What are some advantages in using a CNN (convolutional neural network) rather than a DNN (dense neural network) in an image classification task?

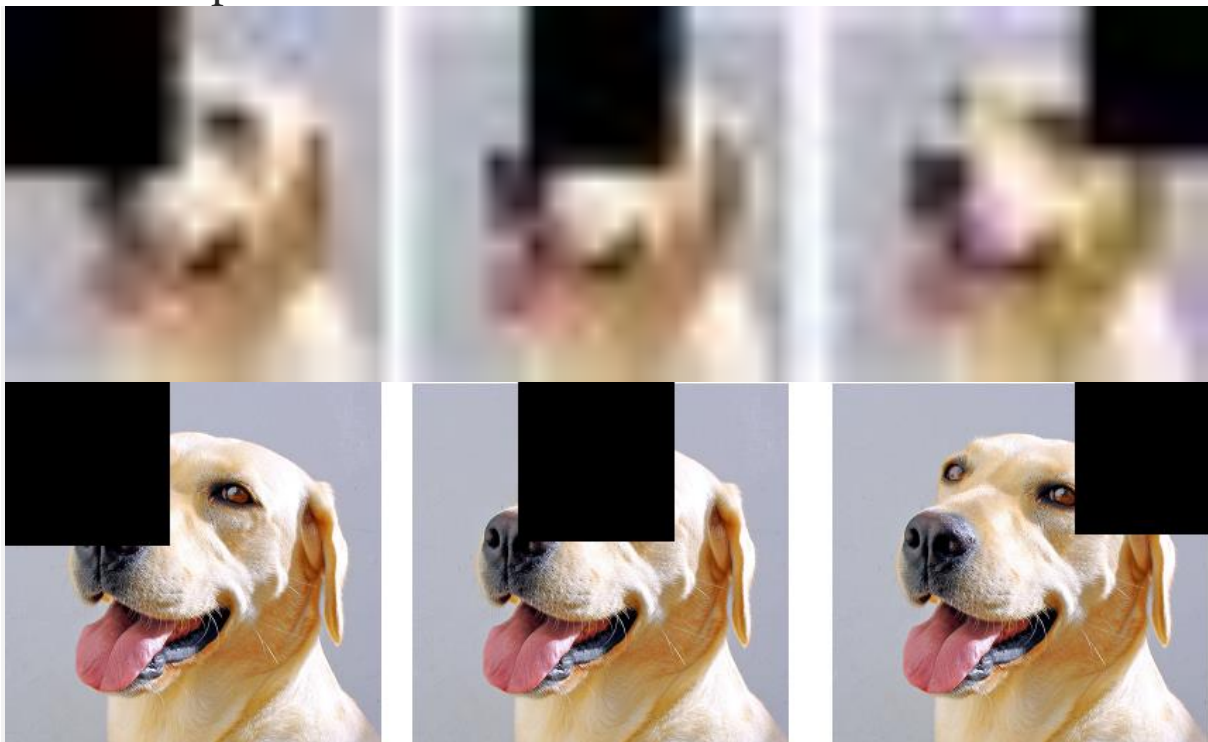
Solution: while both models can capture the relationship between close pixels, CNNs have the following properties:

- It is translation invariant — the exact location of the pixel is irrelevant for the filter.
- It is less likely to overfit — the typical number of parameters in a CNN is much smaller than that of a DNN.
- Gives us a better understanding of the model — we can look at the filters' weights and visualize what the network “learned”.
- Hierarchical nature — learns patterns in by describing complex patterns using simpler ones.

108. Describe two ways to visualize features of a CNN in an image classification task.

Solution:

- Input occlusion — cover a part of the input image and see which part affect the classification the most. For instance, given a trained image classification model, give the images below as input. If, for instance, we see that the 3rd image is classified with 98% probability as a dog, while the 2nd image only with 65% accuracy, it means that the part covered in the 2nd image is more important.



- Activation Maximization — the idea is to create an artificial input image that maximize the target response (gradient ascent).

109. Is trying the following learning rates: 0.1,0.2,...,0.5 a good strategy to optimize the learning rate?

Solution: No, it is recommended to try a logarithmic scale to optimize the learning rate.

110. Suppose you have a NN with 3 layers and ReLU activations. What will happen if we initialize all the weights with the same value? what if we only had 1 layer (i.e linear/logistic regression?)

Solution: If we initialize all the weights to be the same we would not be able to break the symmetry; i.e, all gradients will be updated the same and the network will not be able to learn. In the 1-layers scenario, however, the cost function is convex (linear/sigmoid) and thus the weights will always converge to the optimal point, regardless of the initial value (convergence may be slower).

111. Explain the idea behind the Adam optimizer.

Solution: Adam, or adaptive momentum, combines two ideas to improve convergence: per-parameter updates which give faster convergence, and momentum which helps to avoid getting stuck in saddle point.

112. Compare batch, mini-batch and stochastic gradient descent.

Solution: batch refers to estimating the data by taking the entire data, mini-batch by sampling a few datapoints, and SGD refers to update the gradient one datapoint at each epoch. The tradeoff

here is between how precise the calculation of the gradient is versus what size of batch we can keep in memory. Moreover, taking mini-batch rather than the entire batch has a regularizing effect by adding random noise at each epoch.

113. What is data augmentation? Give examples.

Solution: Data augmentation is a technique to increase the input data by performing manipulations on the original data. For instance in images, one can: rotate the image, reflect (flip) the image, add Gaussian blur.

114. What is the idea behind GANs?

Solution: GANs, or generative adversarial networks, consist of two networks (D,G) where D is the “discriminator” network and G is the “generative” network. The goal is to create data — images, for instance, which are undistinguishable from real images. Suppose we want to create an adversarial example of a cat. The network G will generate images. The network D will classify images according to whether they are a cat or not. The cost function of G will be constructed such that it tries to “fool” D — to classify its output always as cat.

115. What are the advantages of using Batchnorm?

Solution: Batchnorm accelerates the training process. It also (as a byproduct of including some noise) has a regularizing effect.

116. What is multi-task learning? When should it be used?

Solution: Multi-tasking is useful when we have a small amount of data for some task, and we would benefit from training a model on a large dataset of another task. Parameters of the models are shared — either in a “hard” way (i.e the same parameters) or a “soft” way (i.e regularization/penalty to the cost function).

117. What is end-to-end learning? Give a few of its advantages.

Solution: End-to-end learning is usually a model which gets the raw data and outputs directly the desired outcome, with no intermediate tasks or feature engineering. It has several advantages, among which: there is no need to handcraft features, and it generally leads to lower bias.

118. What happens if we use a ReLU activation and then a sigmoid as the final layer?

Solution: Since ReLU always outputs a non-negative result, the network will constantly predict one class for all the inputs!

119. How to solve the exploding gradient problem?

Solution: A simple solution to the exploding gradient problem is gradient clipping — taking the gradient to be $\pm M$ when its absolute value is bigger than M , where M is some large number.

120. Is it necessary to shuffle the training data when using batch gradient descent?

Solution: No, because the gradient is calculated at each epoch using the entire training data, so shuffling does not make a difference.

121. When using mini batch gradient descent, why is it important to shuffle the data?

Solution: otherwise, suppose we train a NN classifier and have two classes — A and B, and that all samples of one class come before the other class. Not shuffling the data will make the weights converge to a wrong value.

122. Describe some hyperparameters for transfer learning.

Solution: How many layers to keep, how many layers to add, how many to freeze.

123. Is dropout used on the test set?

Solution: No! only in the train set. Dropout is a regularization technique that is applied in the training process.

124. Explain why dropout in a neural network acts as a regularizer.

Solution: There are several (related) explanations to why dropout works. It can be seen as a form of model averaging — at each step we “turn off” a part of the model and average the models we get. It also adds noise, which naturally has a regularizing effect. It also leads to more sparsity of the weights and essentially prevents co-adaptation of neurons in the network.

125. Give examples in which a many-to-one RNN architecture is appropriate.

Solution: A few examples are: sentiment analysis, gender recognition from speech, .

126. When can't we use BiLSTM? Explain what assumption has to be made.

Solution: in any bi-directional model, we assume that we have access to the next elements of the sequence in a given “time”. This is the case for text data (i.e sentiment analysis, translation etc.), but not the case for time-series data.

127. True/false: adding L2 regularization to a RNN can help with the vanishing gradient problem.

Solution: false! Adding L2 regularization will shrink the weights towards zero, which can actually make the vanishing gradients worse in some cases.

128. Suppose the training error/cost is high and that the validation cost/error is almost equal to it. What does it mean? What should be done?

Solution: this indicates underfitting. One can add more parameters, increase the complexity of the model, or lower the regularization.

129. Describe how L2 regularization can be explained as a sort of a weight decay.

Solution: Suppose our cost function is $C(w)$, and that we add a penalization $c|w|^2$. When using gradient descent, the iterations will look like

$$w = w - \text{grad}(C)(w) - 2cw = (1-2c)w - \text{grad}(C)(w)$$

In this equation, the weight is multiplied by a factor < 1 .