Top Deep Learning Interview Questions You Must Know

1.3K Views

Kurt
Last updated on May 22,2019

Deep Learning is one of the Hottest topics of 2018-19 and for a good reason. There have been so many advancements in the Industry wherein the time has come when machines or Computer Programs are actually replacing Humans. Artificial Intelligence is going to create 2.3 million Jobs by 2020 and to crack those job interview I have come up with a set of Deep Learning Interview Questions. I have divided this article into two sections:

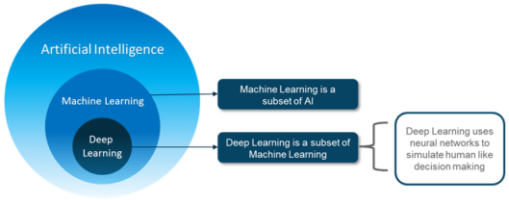
- Basic Deep Learning Interview Questions
- Advance Deep Learning Interview Questions

Basics Deep Learning Interview Questions

Q1. Differentiate between AI, Machine Learning and Deep Learning.

Artificial Intelligence is a technique which enables machines to mimic human behavior.

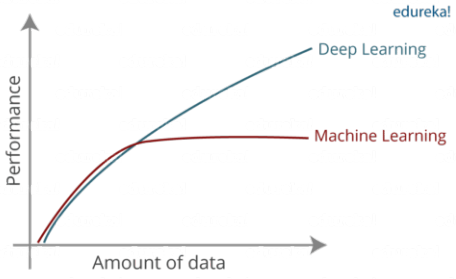
Machine Learning is a subset of AI technique which uses statistical methods to enable machines to improve with experience.



Deep learning is a subset of ML which make the computation of multi-layer neural network feasible. It uses Neural networks to simulate human-like decision making.

Q2. Do you think Deep Learning is Better than Machine Learning? If so, why?

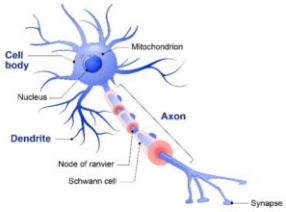
Though traditional ML algorithms solve a lot of our cases, they are not useful while working with high dimensional data, that is where we have a large number of inputs and outputs. For example, in the case of handwriting recognition, we have a large amount of input where we will have a different type of inputs associated with different type of handwriting.



The second major challenge is to tell the computer what are the features it should look for that will play an important role in predicting the outcome as well as to achieve better accuracy while doing so.

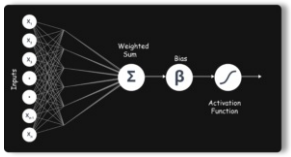
Q3. What is Perceptron? And How does it Work?

If we focus on the structure of a biological neuron, it has dendrites which are used to receive inputs. These inputs are summed in the cell body and using the Axon it is passed on to the next biological neuron as shown below.



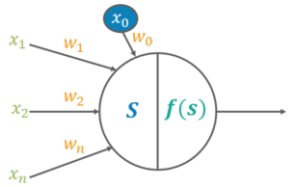
- **Dendrite:** Receives signals from other neurons
- **Cell Body:** Sums all the inputs
- **Axon:** It is used to transmit signals to the other cells

Similarly, a perceptron receives multiple inputs, applies various transformations and functions and provides an output. A Perceptron is a linear model used for binary classification. It models a neuron which has a set of inputs, each of which is given a specific weight. The neuron computes some function on these weighted inputs and gives the output.



Q4. What is the role of weights and bias?

For a perceptron, there can be one more input called **bias**. While the weights determine the **slope** of the classifier line, bias allows us to shift the line towards left or right. Normally bias is treated as another weighted input with the input value x_0.



Q5. What are the activation functions?

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

There can be many Activation functions like:

- Linear or Identity
- Unit or Binary Step
- Sigmoid or Logistic
- Tanh
- ReLU
- Softmax

Q6. Explain Learning of a Perceptron.

1. Initializing the weights and threshold.
2. Provide the input and calculate the output.
3. Update the weights.
4. Repeat Steps 2 and 3

$$Wj\ (t+1) = Wj\ (t) + n\ (d-y)\ x$$

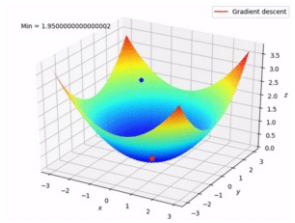
Wj (t+1) – Updated Weight
Wj (t) – Old Weight
d – Desired Output
y – Actual Output
x – Input

Q7. What is the significance of a Cost/Loss function?

A cost function is **a measure of the accuracy** of the neural network with respect to a given training sample and expected output. It provides the performance of a neural network as a whole. In deep learning, the goal is to minimize the cost function. For that, we use the concept of gradient descent.

Q8. What is gradient descent?

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient.



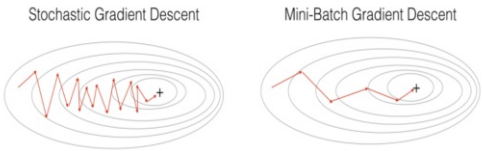
Stochastic Gradient Descent: Uses only a single training example to calculate the gradient and update parameters.

Batch Gradient Descent: Calculate the gradients for the whole dataset and perform just one update at each iteration.

Mini-batch Gradient Descent: Mini-batch gradient is a variation of stochastic gradient descent where instead of single training example, mini-batch of samples is used. It's one of the most popular optimization algorithms.

Q9. What are the benefits of mini-batch gradient descent?

- This is more efficient compared to stochastic gradient descent.
- The generalization by finding the flat minima.
- Mini-batches allows help to approximate the gradient of the entire training set which helps us to avoid local minima.



Q10.What are the steps for using a gradient descent algorithm?

- Initialize random weight and bias.
- Pass an input through the network and get values from the output layer.
- Calculate the error between the actual value and the predicted value.
- Go to each neuron which contributes to the error and then change its respective values to reduce the error.
- Reiterate until you find the best weights of the network.

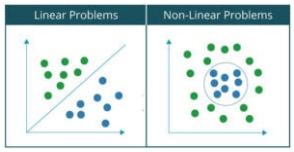
Q11. Create a Gradient Descent in python.

```
params = [weights_hidden, weights_output, bias_hidden, bias_output]

def sgd(cost, params, lr=0.05):

grads = T.grad(cost=cost, wrt=params)
updates = []

for p, g in zip(params, grads):
updates.append([p, p - g * lr])

return updates

updates = sgd(cost, params)
```

Q12. What are the shortcomings of a single layer perceptron?

Well, there are two major problems:

- Single-Layer Perceptrons cannot classify non-linearly separable data points.
- Complex problems, that involve a lot of parameters cannot be solved by Single-Layer Perceptrons



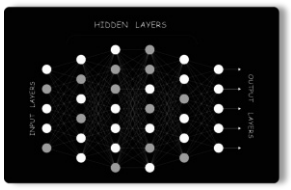
Q13. What is a Multi-Layer-Perceptron

A multilayer perceptron (MLP) is a deep, artificial neural network. It is composed of more than one perceptron. They are composed of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two, an arbitrary number of hidden layers that are the true computational engine of the MLP.

Q14. What are the different parts of a multi-layer perceptron?

Input Nodes: The Input nodes provide information from the outside world to the network and are together referred to as the "Input Layer". No computation is performed in any of the Input nodes – they just pass on the information to the hidden nodes.

Hidden Nodes: The Hidden nodes perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a "Hidden Layer". While a network will only have a single input layer and a single output layer, it can have zero or multiple Hidden Layers.



Output Nodes: The Output nodes are collectively referred to as the "Output Layer" and are responsible for computations and transferring information from the network to the outside world.

Q15. What Is Data Normalization And Why Do We Need It?

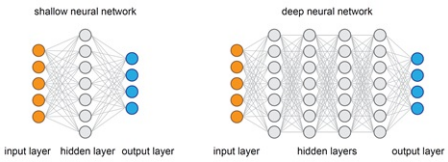
Data normalization is very important preprocessing step, used to rescale values to fit in a specific range to assure better convergence during backpropagation. In general, it boils down to subtracting the mean of each data point and dividing by its standard deviation.

These were some basic Deep Learning Interview Questions. Now, let's move on to some advanced ones.

Advance Interview Questions

Q16. Which is Better Deep Networks or Shallow ones? and Why?

Both the Networks, be it shallow or Deep are capable of approximating any function. But what matters is how precise that network is in terms of getting the results. A shallow network works with only a few features, as it can't extract more. But a deep network goes deep by computing efficiently and working on more features/parameters.



Q17. Why is Weight Initialization important in Neural Networks?

Weight initialization is one of the very important steps. A bad weight initialization can prevent a network from learning but good weight initialization helps in giving a quicker convergence and a better overall error.

Biases can be generally initialized to zero. The rule for setting the weights is to be close to zero without being too small.

Q18. What's the difference between a feed-forward and a backpropagation neural network?

A Feed-Forward Neural Network is a type of Neural Network architecture where the connections are "fed forward", i.e. do not form cycles. The term "Feed-Forward" is also used when you input something at the input layer and it travels from input to hidden and from hidden to the output layer.

Backpropagation is a training algorithm consisting of 2 steps:

- Feed-Forward the values.
- Calculate the error and propagate it back to the earlier layers.

So to be precise, forward-propagation is part of the backpropagation algorithm but comes before back-propagating.

Q19. What are the Hperparameteres? Name a few used in any Neural Network.

Hyperparameters are the variables which determine the network structure(Eg: Number of Hidden Units) and the variables which determine how the network is trained(Eg: Learning Rate). Hyperparameters are set before training.

- Number of Hidden Layers
- Network Weight Initialization
- Activation Function
- Learning Rate
- Momentum
- Number of Epochs
- Batch Size

Q20. Explain the different Hyperparameters related to Network and Training.

Network Hyperparameters

The number of Hidden Layers: Many hidden units within a layer with regularization techniques can increase accuracy. Smaller number of units may cause underfitting.

Network Weight Initialization: Ideally, it may be better to use different weight initialization schemes according to the activation function used on each layer. Mostly uniform distribution is used.

Activation function: Activation functions are used to introduce nonlinearity to models, which allows deep learning models to learn nonlinear prediction boundaries.

Training Hyperparameters

Learning Rate: The learning rate defines how quickly a network updates its parameters. Low learning rate slows down the learning process but converges smoothly. Larger learning rate speeds up the learning but may not converge.

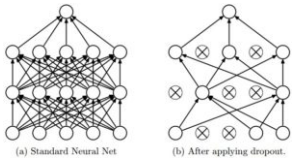
Momentum: Momentum helps to know the direction of the next step with the knowledge of the previous steps. It helps to prevent oscillations. A typical choice of momentum is between 0.5 to 0.9.

The number of epochs: Number of epochs is the number of times the whole training data is shown to the network while training. Increase the number of epochs until the validation accuracy starts decreasing even when training accuracy is increasing(overfitting).

Batch size: Mini batch size is the number of sub-samples given to the network after which parameter update happens. A good default for batch size might be 32. Also try 32, 64, 128, 256, and so on.

Q21. What is Dropout?

Dropout is a regularization technique to avoid overfitting thus increasing the generalizing power. Generally, we should use a small dropout value of 20%-50% of neurons with 20% providing a good starting point. A probability too low has minimal effect and a value too high results in under-learning by the network.


(a) Standard Neural Net (b) After applying dropout.

Use a larger network. You are likely to get better performance when dropout is used on a larger network, giving the model more of an opportunity to learn independent representations.

Q22. In training a neural network, you notice that the loss does not decrease in the few starting epochs. What could be the reason?

The reasons for this could be:

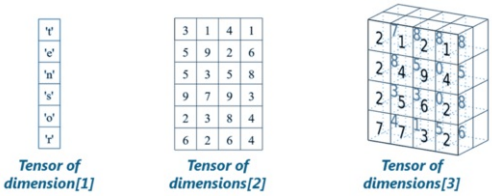
- The learning is rate is low
- Regularization parameter is high
- Stuck at local minima

Q23. Name a few deep learning frameworks

- TensorFlow
- Caffe
- The Microsoft Cognitive Toolkit/CNTK
- Torch/PyTorch
- MXNet
- Chainer
- Keras

Q24. What are Tensors?

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

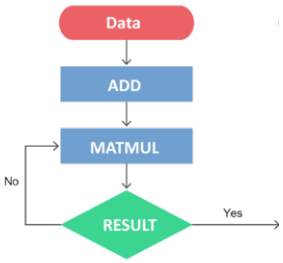


Q25. List a few advantages of TensorFlow?

- It has platform flexibility
- It is easily trainable on CPU as well as GPU for distributed computing.
- TensorFlow has auto differentiation capabilities
- It has advanced support for threads, asynchronous computation, and queue es.
- It is a customizable and open source.

Q26. What is Computational Graph?

A computational graph is a series of TensorFlow operations arranged as nodes in the graph. Each node takes zero or more tensors as input and produces a tensor as output.



Basically, one can think of a Computational Graph as an alternative way of conceptualizing mathematical calculations that takes place in a TensorFlow program. The operations assigned to different nodes of a Computational Graph can be performed in parallel, thus, providing better performance in terms of computations.

Q27. What is a CNN?

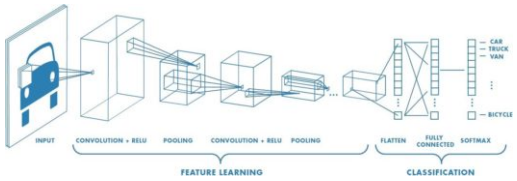
Convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. Unlike neural networks, where the input is a vector, here the input is a multi-channeled image. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.

Q28. Explain the different Layers of CNN.

There are four layered concepts we should understand in Convolutional Neural Networks:

Convolution: The convolution layer comprises of a set of independent filters. All these filters are initialized randomly and become our parameters which will be learned by the network subsequently.

ReLu: This layer is used with the convolutional layer.



Pooling: Its function is to progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network. Pooling layer operates on each feature map independently.

Full Connectedness: Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

Q29. What is an RNN?

Recurrent Networks are a type of artificial neural network designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, numerical times series data. Recurrent Neural Networks use **backpropagation** algorithm for training Because of their **internal memory**, RNN's are able to remember important things about the input they received, which enables them to be very precise in predicting what's coming next.

Q30. What are some issues faced while training an RNN?

Recurrent Neural Networks use backpropagation algorithm for training, but it is applied for every timestamp. It is commonly known as **Back-propagation Through Time** (BTT).

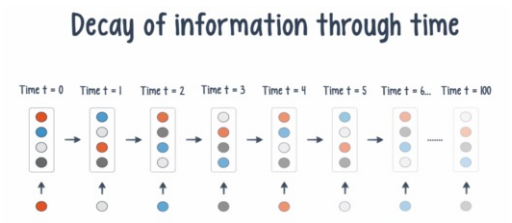
There are some issues with Back-propagation such as:

- Vanishing Gradient
- Exploding Gradient

Q31. What is Vanishing Gradient? And how is this harmful?

When we do Back-propagation, the gradients tend to get smaller and smaller as we keep on moving backward in the Network. This means that the neurons in the Earlier layers learn very slowly as compared to the neurons in the later layers in the Hierarchy.

Earlier layers in the Network are important because they are responsible to learn and detecting the simple patterns and are actually the building blocks of our Network.



Obviously, if they give improper and inaccurate results, then how can we expect the next layers and the complete Network to perform nicely and produce accurate results. The Training process takes too long and the Prediction Accuracy of the Model will decrease.

Q32. What is Exploding Gradient Descent?

Exploding gradients are a problem when large error gradients accumulate and result in very large updates to neural network model weights during training.

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

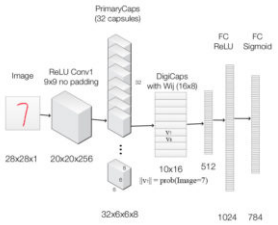
Q33. Explain the importance of LSTM.

Long short-term memory(LSTM) is an artificial recurrent neural network architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections that make it a "general purpose computer". It can not only process single data points, but also entire sequences of data.

They are a special kind of Recurrent Neural Networks which are capable of learning long-term dependencies.

Q34. What are capsules in Capsule Neural Network?

Capsules are a vector specifying the features of the object and its likelihood. These features can be any of the instantiation parameters like position, size, orientation, deformation, velocity, hue, texture and much more.

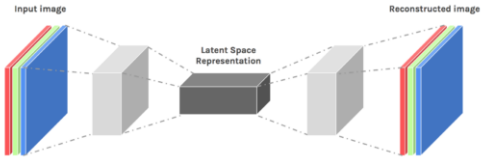


A capsule can also specify its attributes like angle and size so that it can represent the same generic information. Now, just like a neural network has layers of neurons, a capsule network can have layers of capsules.

Now, let's continue this Deep Learning Interview Questions and move to the section of autoencoders and RBMs.

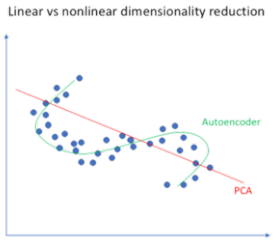
Q35. Explain Autoencoders and it's uses.

An autoencoder neural network is an Unsupervised Machine learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. Autoencoders are used to reduce the size of our inputs into a smaller representation. If anyone needs the original data, they can reconstruct it from the compressed data.



Q36. In terms of Dimensionality Reduction, How does Autoencoder differ from PCAs?

- An autoencoder can learn non-linear transformations with a non-linear activation function and multiple layers.
- It doesn't have to learn dense layers. It can use convolutional layers to learn which is better for video, image and series data.
- It is more efficient to learn several layers with an autoencoder rather than learn one huge transformation with PCA.
- An autoencoder provides a representation of each layer as the output.
- It can make use of pre-trained layers from another model to apply transfer learning to enhance the encoder/decoder.



Q37. Give some real-life examples where autoencoders can be applied.

Image Coloring: Autoencoders are used for converting any black and white picture into a colored image. Depending on what is in the picture, it is possible to tell what the color should be.

Feature variation: It extracts only the required features of an image and generates the output by removing any noise or unnecessary interruption.

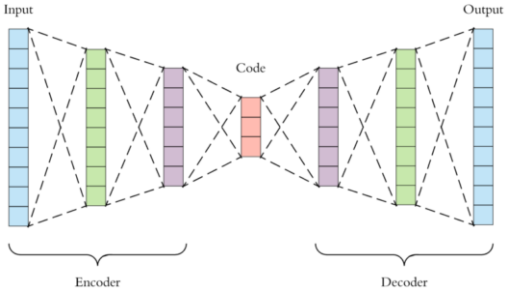
Dimensionality Reduction: The reconstructed image is the same as our input but with reduced dimensions. It helps in providing a similar image with a reduced pixel value.

Denoising Image: The input seen by the autoencoder is not the raw input but a stochastically corrupted version. A denoising autoencoder is thus trained to reconstruct the original input from the noisy version.

Q38. what are the different layers of Autoencoders?

An Autoencoder consist of three layers:

- Encoder
- Code
- Decoder



Q39. Explain the architecture of an Autoencoder.

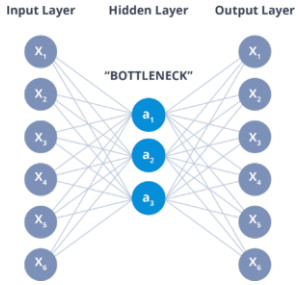
Encoder: This part of the network compresses the input into a latent space representation. The encoder layer encodes the input image as a compressed representation in a reduced dimension. The compressed image is the distorted version of the original image.

Code: This part of the network represents the compressed input which is fed to the decoder.

Decoder: This layer decodes the encoded image back to the original dimension. The decoded image is a lossy reconstruction of the original image and it is reconstructed from the latent space representation.

Q40. What is a Bottleneck in autoencoder and why is it used?

The layer between the encoder and decoder, ie. the code is also known as Bottleneck. This is a well-designed approach to decide which aspects of observed data are relevant information and what aspects can be discarded.



It does this by balancing two criteria:

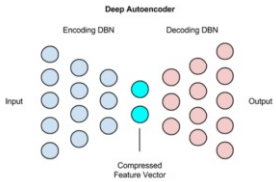
- Compactness of representation, measured as the compressibility.
- It retains some behaviourally relevant variables from the input.

Q41. Is there any variation of Autoencoders?

- Convolution Autoencoders
- Sparse Autoencoders
- Deep Autoencoders
- Contractive Autoencoders

Q42. What are Deep Autoencoders?

The extension of the simple Autoencoder is the Deep Autoencoder. The first layer of the Deep Autoencoder is used for first-order features in the raw input. The second layer is used for second-order features corresponding to patterns in the appearance of first-order features. Deeper layers of the Deep Autoencoder tend to learn even higher-order features.

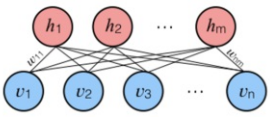


A deep autoencoder is composed of two, symmetrical deep-belief networks:

- First four or five shallow layers representing the encoding half of the net.
- The second set of four or five layers that make up the decoding half.

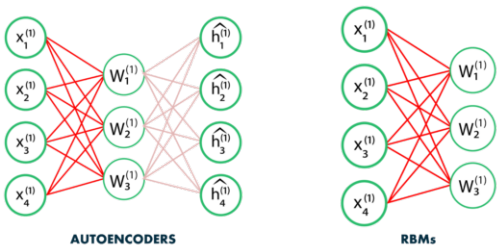
Q43. What is a Restricted Boltzmann Machine?

Restricted Boltzmann Machine is an undirected graphical model that plays a major role in Deep Learning Framework in recent times.
It is an algorithm which is useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning, and topic modeling.



Q44. How Does RBM differ from Autoencoders?

Autoencoder is a simple 3-layer neural network where output units are directly connected back to input units. Typically, the number of hidden units is much less than the number of visible ones. The task of training is to minimize an error or reconstruction, i.e. find the most efficient compact representation for input data.



RBM shares a similar idea, but it uses stochastic units with particular distribution instead of deterministic distribution. The task of training is to find out how these two sets of variables are actually