DATA SCIENCE INTERVIEW PREPARATION (30 Days of Interview Preparation)

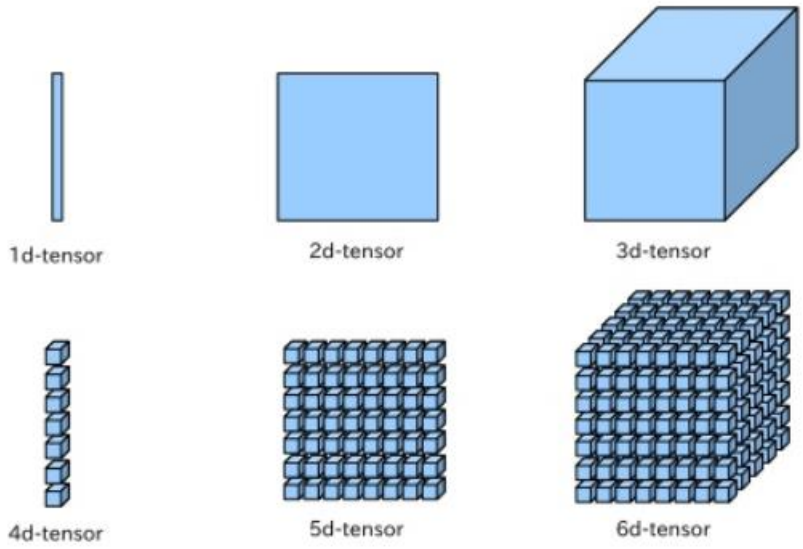
DAY 11

Q1. What are tensors?

Answer:

The tensors are no more than a method of presenting the data in deep learning. If put in the simple term, tensors are just multidimensional arrays that allow developers to represent the data in a layer, which means deep learning you are using contains high-level data sets where each dimension represents a different feature.

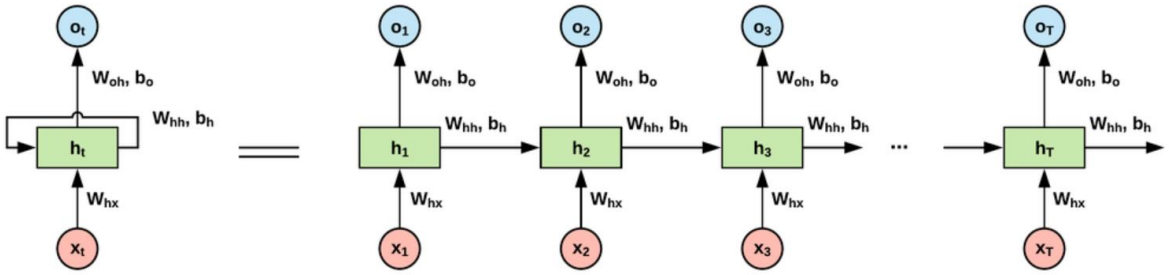
The foremost benefit of using tensors is it provides the much-needed platform-flexibility and is easy to trainable on CPU. Apart from this, tensors have the auto differentiation capabilities, advanced support system for queues, threads, and asynchronous computation. All these features also make it customizable.



Q2. Define the concept of RNN?

Answer:

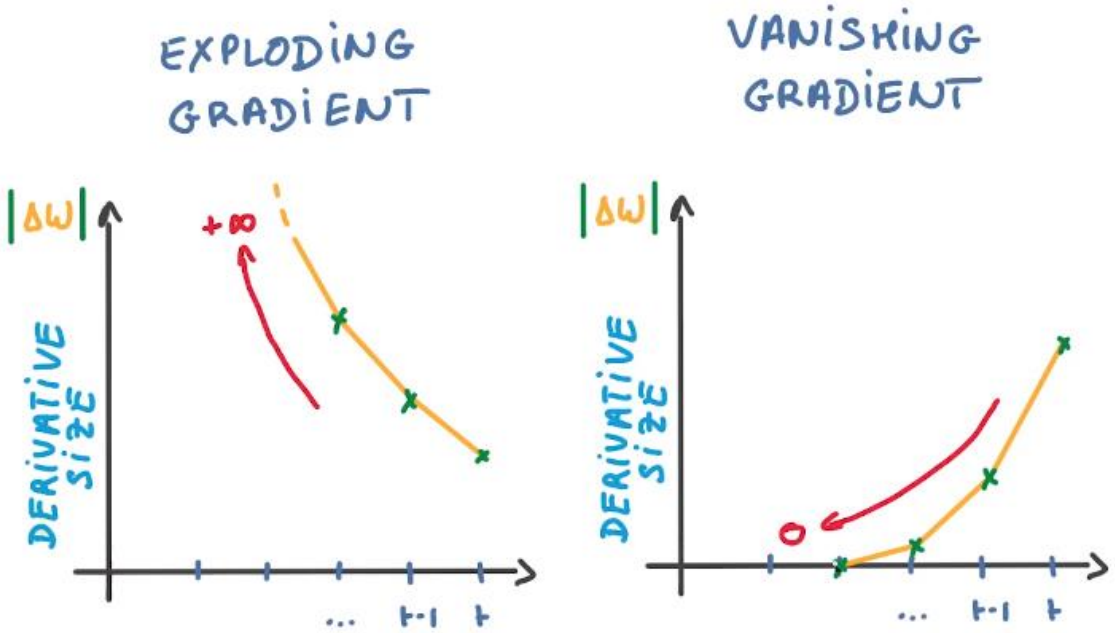
RNN is the artificial neutral which were created to analyze and recognize the patterns in the sequences of the data. Due to their internal memory, RNN can certainly remember the things about the inputs they receive.



Most common issues faced with RNN

Although RNN is around for a while and uses backpropagation, there are some common issues faced by developers who work it. Out of all, some of the most common issues are:

- Exploding gradients
- Vanishing gradients



Q3. What is a ResNet, and where would you use it? Is it efficient?

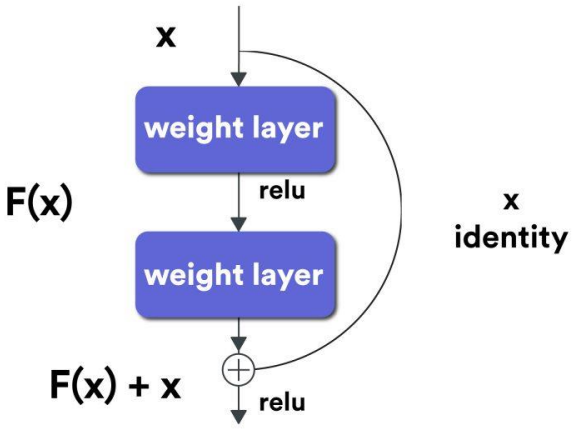
Answer:

Among the various neural networks that are used for computer vision, ResNet (Residual Neural Networks), is one of the most popular ones. It allows us to train extremely deep neural networks, which is the prime reason for its huge usage and popularity. Before the invention of this network, training extremely deep neural networks was almost impossible.

To understand why we must look at the vanishing gradient problem which is an issue that arises when the gradient is backpropagated to all the layers. As a large number of multiplications are performed, the size of the network keeps decreasing till it becomes extremely small, and thus, the network starts performing badly. ResNet helps to counter the vanishing gradient problem.

The efficiency of this network is highly dependent on the concept of skip connections. Skip connections are a method of allowing a shortcut path through which the gradient can flow, which in effect helps counter the vanishing gradient problem.

An example of a skip connection is shown below:



In general, a skip connection allows us to skip the training of a few layers. Skip connections are also called identity shortcut connections as they allow us to directly compute an identity function by just relying on these connections and not having to look at the whole network.

The skipping of these layers makes ResNet an extremely efficient network.

Q4. Transfer learning is one of the most useful concepts today. Where can it be used?

Answer:

Pre-trained models are probably one of the most common use cases for transfer learning.

For anyone who does not have access to huge computational power, training complex models is always a challenge. Transfer learning aims to help by both improving the performance and speeding up your network.

In layman terms, transfer learning is a technique in which a model that has already been trained to do one task is used for another without much change. This type of learning is also called multi-task learning.

Many models that are pre-trained are available online. Any of these models can be used as a starting point in the creation of the new model required. After just using the weights, the model must be refined and adapted on the required data by tuning the parameters of the model.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes

The general idea behind transfer learning is to transfer knowledge not data. For humans, this task is easy – we can generalize models that we have mentally created a long time ago for a different purpose. One or two samples is almost always enough. However, in the case of neural networks, a huge amount of data and computational power are required.

Transfer learning should generally be used when we don't have a lot of labeled training data, or if there already exists a network for the task you are trying to achieve, probably trained on a much more massive dataset. Note, however, that the input of the model must have the same size during training. Also, this works only if the tasks are fairly similar to each other, and the features learned can be generalized. For example, something like learning how to recognize vehicles can probably be extended to learn how to recognize airplanes and helicopters.

Q5. What does tuning of hyperparameters signify? Explain with examples.

Answer:

A hyperparameter is just a variable that defines the structure of the network. Let's go through some hyperparameters and see the effect of tuning them.

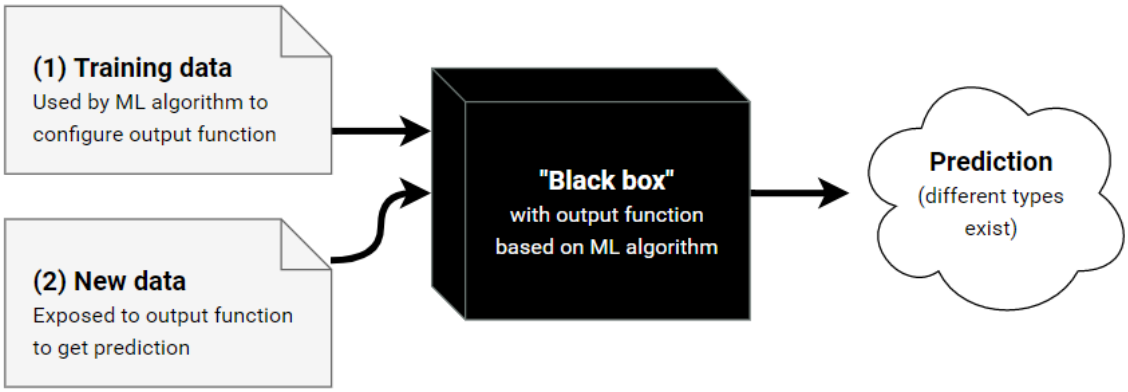
1. A number of hidden layers – Most times, the presence or absence of a large number of hidden layers may determine the output, accuracy and training time of the neural network. Having a large number of these layers may sometimes cause an increase in accuracy.
2. Learning rate – This is simply a measure of how fast the neural network will change its parameters. A large learning rate may lead to the network not being able to converge, but might also speed up learning. On the other hand, a smaller value for the learning rate will probably slow down the network but might lead to the network being able to converge.
3. Number of epochs – This is the number of times the entire training data is run through the network. Increasing the number of epochs leads to better accuracy.
4. Momentum – Momentum is a measure of how and where the network will go while taking into account all of its past actions. A proper measure of momentum can lead to a better network.
5. Batch Size – Batch size determines the number of subsamples that are inputs to the network before every parameter update.

Q6. Why are deep learning models referred as black boxes?

Answer:

Lately, the concept of deep learning being a black box has been floating around. A black box is a system whose functioning cannot be properly grasped, but the output produced can be understood and utilized.

Now, since most models are mathematically sound and are created based on legit equations, how is it possible that we do not know how the system works?



First, it is almost impossible to visualize the functions that are generated by a system. Most machine learning models end up with such complex output that a human can't make sense of it.

Second, there are networks with millions of hyperparameters. As a human, we can grasp around 10 to 15 parameters. But analysing a million of them seems out of the question.

Third and most important, it becomes very hard, if not impossible, to trace back why the system made the decisions it did. This may not sound like a huge problem to worry about but consider the case of a self driving car. If the car hits someone on the road, we need to understand why that happened and prevent it. But this isn’t possible if we do not understand how the system works.

To make a deep learning model not be a black box, a new field called Explainable Artificial Intelligence or simply, Explainable AI is emerging. This field aims to be able to create intermediate results and trace back the decision-making process of a system.

Q7. Why do we have gates in neural networks?

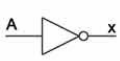
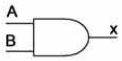
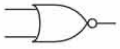
Answer:

To understand gates, we must first understand recurrent neural networks.

Recurrent neural networks allow information to be stored as a memory using loops. Thus, the output of a recurrent neural network is not only based on the current input but also the past inputs which are stored in the memory of the network. Backpropagation is done through time, but in general, the truncated version of this is used for longer sequences.

Gates are generally used in networks that are dependent on time. In effect, any network which would require memory, so to speak, would benefit from the use of gates. These gates are generally used to keep track of any information that is required by the network without leading to a state of either vanishing or exploding gradients. Such a network can also preserve the error through time. Since a sense of constant error is maintained, the network can learn better.

Logic Gates

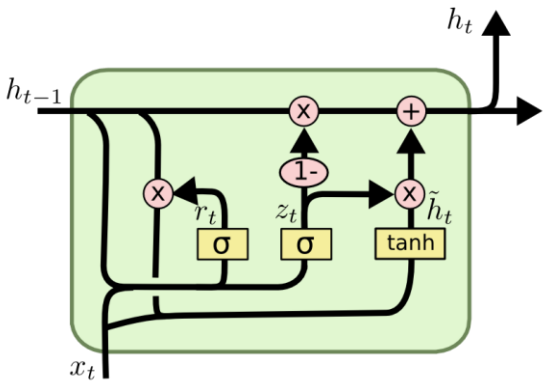
Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

These gated units can be considered as units with recurrent connections. They also contain additional neurons, which are gates. If you relate this process to a signal processing system, the gate is used to

regulate which part of the signal passes through. A sigmoid activation function is used which means that the values taken are from 0 to 1.

An advantage of using gates is that it enables the network to either forget information that it has already learned or to selectively ignore information either based on the state of the network or the input the gate receives.

Gates are extensively used in recurrent neural networks, especially in Long Short-Term Memory (LSTM) networks. A general LSTM network will have 3 to 5 gates, typically an input gate, output gate, hidden gate, and activation gate.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Q8. What is a Sobel filter?

Answer:

The Sobel filter performs a two-dimensional spatial gradient measurement on a given image, which then emphasizes regions that have a high spatial frequency. In effect, this means finding edges.

In most cases, Sobel filters are used to find the approximate absolute gradient magnitude for every point in a grayscale image. The operator consists of a pair of 3×3 convolution kernels. One of these kernels is rotated by 90 degrees.

-1	0	+1
-2	0	+2
-1	0	+1

x filter

+1	+2	+1
0	0	0
-1	-2	-1

y filter

These kernels respond to edges that run horizontal or vertical with respect to the pixel grid, one kernel for each orientation. A point to note is that these kernels can be applied either separately or can be combined to find the absolute magnitude of the gradient at every point.

The Sobel operator has a large convolution kernel, which ends up smoothing the image to a greater extent, and thus, the operator becomes less sensitive to noise. It also produces higher output values for similar edges compared to other methods.

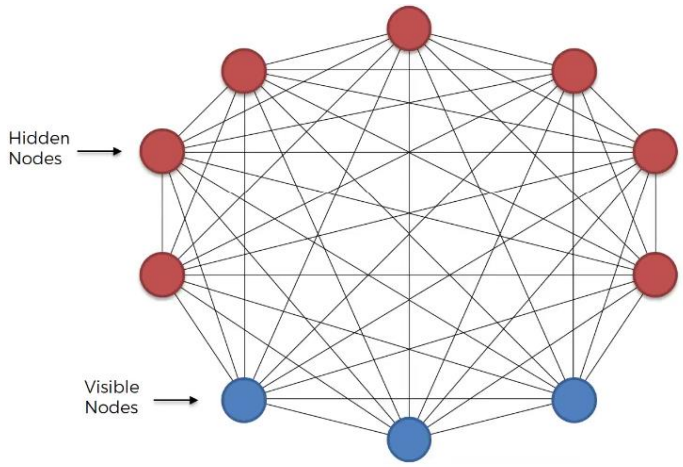
To overcome the problem of output values from the operator overflowing the maximum allowed pixel value per image type, avoid using image types that support pixel values.

Q9. What is the purpose of a Boltzmann Machine?

Answer:

Boltzmann machines are algorithms that are based on physics, specifically thermal equilibrium. A special and more well-known case of Boltzmann machines is the Restricted Boltzmann machine, which is a type of Boltzmann machine where there are no connections between hidden layers of the network.

The concept was coined by Geoff Hinton, who most recently won the Turing award. In general, the algorithm uses the laws of thermodynamics and tries to optimize a global distribution of energy in the system.



In discrete mathematical terms, a restricted Boltzmann machine can be called a symmetric bipartite graph, i.e. two symmetric layers. These machines are a form of unsupervised learning, which means that there are no labels provided with data. It uses stochastic binary units to reach this state.

Boltzmann machines are derived from Markov state machines. A Markov State Machine is a model that can be used to represent almost any computable function. The restricted Boltzmann machine can be regarded as an undirected graphical model. It is used in dimensionality reduction, collaborative filtering, learning features as well as modeling. It can also be used for classification and regression. In general, restricted Boltzmann machines are composed of a two-layer network, which can then be extended further.

Note that these models are probabilistic since each of the nodes present in the system learns low-level features from items in the dataset. For example, if we take a grayscale image, each node that is responsible for the visible layer will take just one-pixel value from the image.

A part of the process of creating such a machine is a feature hierarchy where sequences of activations are grouped in terms of features. In thermodynamics principles, simulated annealing is a process that the machine follows to separate signal and noise.

Q10. What are the types of weight initialization?

Answer:

There are two major types of weight initialization:- zero initialization and random initialization.

Zero initialization: In this process, biases and weights are initialised to 0. If the weights are set to 0, all derivatives with respect to the loss functions in the weight matrix become equal. Hence, none of the weights change during subsequent iterations. Setting the bias to 0 cancels out any effect it may have.

All hidden units become symmetric due to zero initialization. In general, zero initialization is not very useful or accurate for classification and thus must be avoided when any classification task is required.

Random initialization: As compared to 0 initialization, this involves setting random values for the weights. The only disadvantage is that set very high values will increase the learning time as the sigmoid activation function maps close to 1. Likewise, if low values are set, the learning time increases as the activation function is mapped close to 0.

Setting too high or too low values thus generally leads to the exploding or vanishing gradient problem.

New types of weight initialization like **"He initialization"** and **"Xavier initialization"** have also emerged. These are based on specific equations and are not mentioned here due to their sheer complexity.