Deep Learning Assignment Questionnaire Answers

1. What is Deep learning and how it is different from traditional Machine learning?

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to learn and make predictions or decisions without being explicitly programmed. It involves training algorithms on large datasets to identify patterns and relationships and then using these patterns to make predictions or decisions about new data.

Machine learning is further divided into categories based on the data on which we are training our model.

- Supervised Learning – This method is used when we have Training data along with the labels for the correct answer.
- Unsupervised Learning– In this task our main objective is to find the patterns or groups in the dataset at hand because we don't have any particular labels in this dataset.

Deep learning, on the other hand, is a subset of machine learning that uses neural networks with multiple layers to analyse complex patterns and relationships in data. It is inspired by the structure and function of the human brain and has been successful in a variety of tasks, such as computer vision, natural language processing, and speech recognition.
Deep learning models are trained using large amounts of data and algorithms that are able to learn and improve over time, becoming more accurate as they process more data. This makes them well-suited to complex, real-world problems and enables them to learn and adapt to new situations.

2. Do you think Deep learning is better than Machine learning? If yes state the reason why?

- Deep Learning outperform other techniques if the data size is large. But with small data size, traditional Machine Learning algorithms are preferable.
- Deep Learning techniques need to have high end infrastructure to train in reasonable time.
- When there is lack of domain understanding for feature introspection, Deep Learning techniques outshines others as you have to worry less about feature engineering.
- Deep Learning really shines when it comes to complex problems such as image classification, natural language processing, and speech recognition.

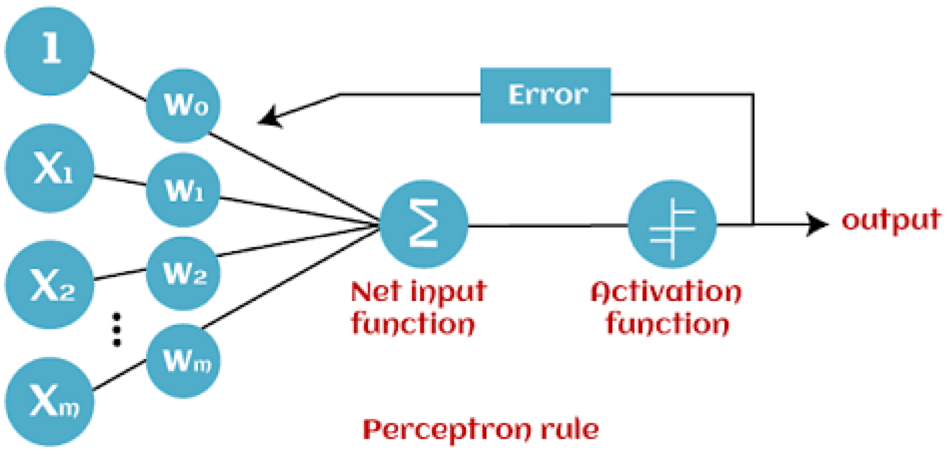
3. Name few Deep Learning frameworks?

- TensorFlow
- 2. TORCH/ PyTorch
- 3. DEEPLEARNING4J
- 4. THE MICROSOFT COGNITIVE TOOLKIT/CNTK
- 5. KERAS
- 6. ONNX
- 7. MXNET

4. What is Perceptron? And how does it work?

A machine-based algorithm used for supervised learning of various binary sorting tasks is called Perceptron. Furthermore, Perceptron also has an essential role as an Artificial Neuron or Neural link in detecting certain input data computations in business intelligence. A perceptron model is also classified as one of the best and most specific types of Artificial Neural networks. Being a supervised learning algorithm of binary classifiers, we can also consider it a single-layer neural network with four main parameters: input values, weights and Bias, net sum, and an activation function.

AS discussed earlier, Perceptron is considered a single-layer neural link with four main parameters. The perceptron model begins with multiplying all input values and their weights, then adds these values to create the weighted sum. Further, this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the step function and is represented by 'f.'



This step function or Activation function is vital in ensuring that output is mapped between (0,1) or (-1,1). Take note that the weight of input indicates a node's strength. Similarly, an input value gives the ability the shift the activation function curve up or down.
Step 1: Multiply all input values with corresponding weight values and then add to calculate the weighted sum. The following is the mathematical expression of it:
∑wi*xi = x1*w1 + x2*w2 + x3*w3+……..x4*w4
Add a term called bias 'b' to this weighted sum to improve the model's performance.
Step 2: An activation function is applied with the above-mentioned weighted sum giving us an output either in binary form or a continuous value as follow: Y=f(∑wi*xi + b)

5. What is neural network? Explain with example and diagram with the elements present in the Neural network?

A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature.

- Neural networks are a series of algorithms that mimic the operations of an animal brain to recognize relationships between vast amounts of data.
- As such, they tend to resemble the connections of neurons and synapses found in the brain.
- They are used in a variety of applications in financial services, from forecasting and marketing research to fraud detection and risk assessment.
- Neural networks with several process layers are known as "deep" networks and are used for deep learning algorithms
- The success of neural networks for stock market price prediction varies.

Types of Neural Networks

Feed-Forward Neural Networks

Feed-forward neural networks are one of the more simple types of neural networks. It conveys information in one direction through input nodes; this information continues to be processed in this single direction until it reaches the output mode. Feed-forward neural networks may have hidden layers for functionality, and this type of most often used for facial recognition technologies.

Recurrent Neural Networks

A more complex type of neural network, recurrent neural networks take the output of a processing node and transmit the information back into the network. This results in theoretical "learning" and improvement of the network. Each node stores historical processes, and these historical processes are reused in the future during processing.

This becomes especially critical for networks in which the prediction is incorrect; the system will attempt to learn why the correct outcome occurred and adjust accordingly. This type of neural network is often used in text-to-speech applications.

Convolutional Neural Networks

Convolutional neural networks, also called ConvNets or CNNs, have several layers in which data is sorted into categories. These networks have an input layer, an output layer, and a hidden multitude of convolutional layers in between. The layers create feature maps that record areas of an image that are broken down further until they generate valuable outputs. These layers can be pooled or entirely connected, and these networks are especially beneficial for image recognition applications.

6. What is the role of weight and bias in neural network?

Weights and biases are crucial concepts to a neural network. The neural network processes the characteristics of a data subject (like an image or audio clip) and produces an identification of the subject.

- Weights set the standards for the neuron’s signal strength. This value will determine the influence input data has on the output product.
- Biases give extra characteristics with a value of 1 that the neural network did not previously have. The neural network needs that extra information to efficiently propagate forward.
- Weights and biases both better distinguish the neurons and their connections to give an accurate output.

7. What are the different types of Architectures present in deep learning and state their applications?

When it comes to deep learning, you have various types of neural networks. And deep learning architectures are based on these networks. Today, we can indicate six of the most common deep learning architectures:

- **RNN**

RNNs are very useful when it comes to fields where the sequence of presented information is key. They are commonly used in NLP (i.a. chatbots), speech synthesis, and machine translations.

- **LSTM**
  Today, LSTMs are commonly used in such fields as text compression, handwriting recognition, speech recognition, gesture recognition, and image captioning [
- **GRU**
  This abbreviation stands for Gated Recurrent Unit. It's a type of LSTM. The major difference is that GRU has fewer parameters than LSTM, as it lacks an output gate. GRUs are used for smaller and less frequent datasets, where they show better performance.
- **CNN**
  This architecture is commonly used for image processing, image recognition, video analysis, and NLP.
- **DBN**
  DBNs can be used i.a. in image recognition and NLP.
- **DSN**

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

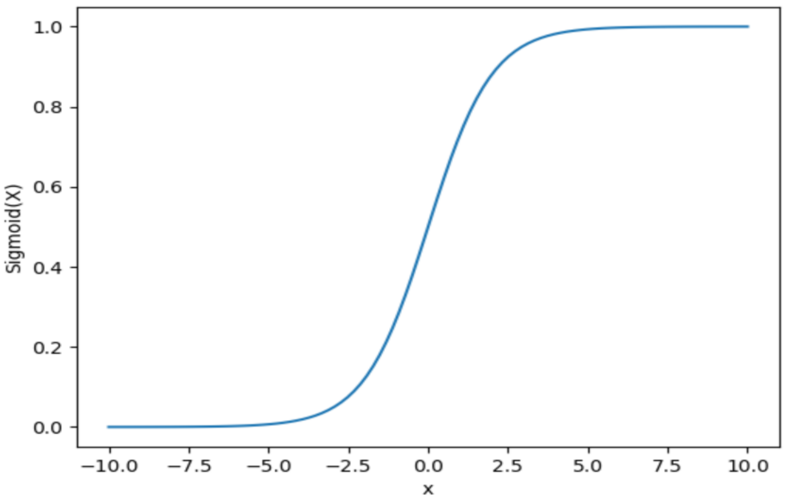
Explanation: We know, the neural network has neurons that work in correspondence with *weight, bias,* and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as ***back-propagation***. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

Linear Function

- **Equation :** Linear function has the equation similar to as of a straight line i.e. **y = x**
- No matter how many layers we have, if all are linear in nature, the final activation function of last layer is nothing but just a linear function of the input of first layer.
- **Range :** -inf to +inf
- **Uses : Linear activation function** is used at just one place i.e. output layer.
- **Issues :** If we will differentiate linear function to bring non-linearity, result will no more depend on *input "x"* and function will become constant, it won't introduce any ground-breaking behavior to our algorithm.

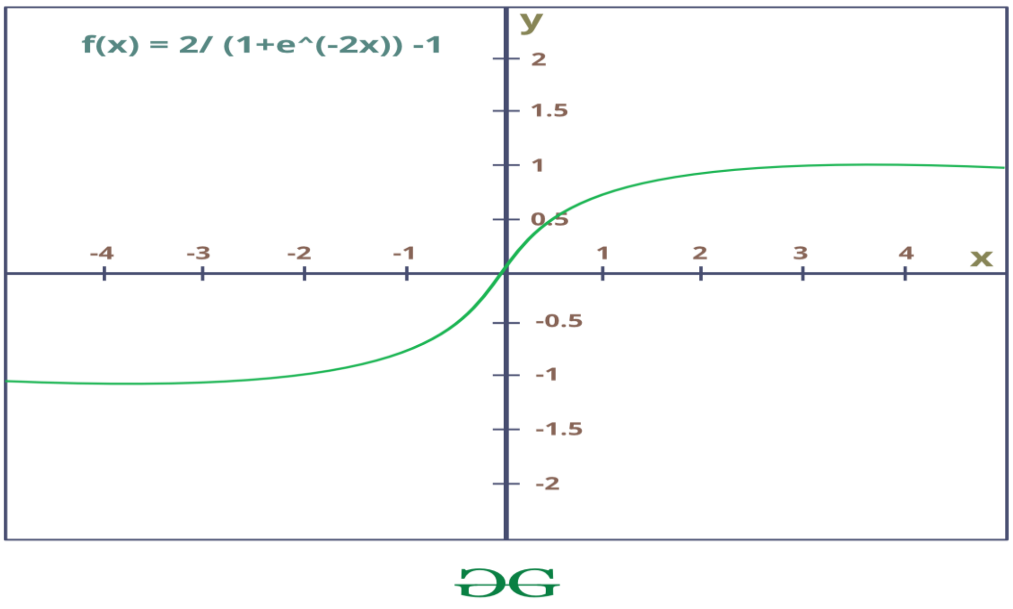
For example : Calculation of price of a house is a regression problem. House price may have any big/small value, so we can apply linear activation at output layer. Even in this case neural net must have any non-linear function at hidden layers.

Sigmoid Function



- It is a function which is plotted as **'S'** shaped graph.
- **Equation :** A = 1/(1 + e^{-x})
- **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
- **Value Range :** 0 to 1
- **Uses :** Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be **1** if value is greater than **0.5** and **0** otherwise.
-

Tanh



Function

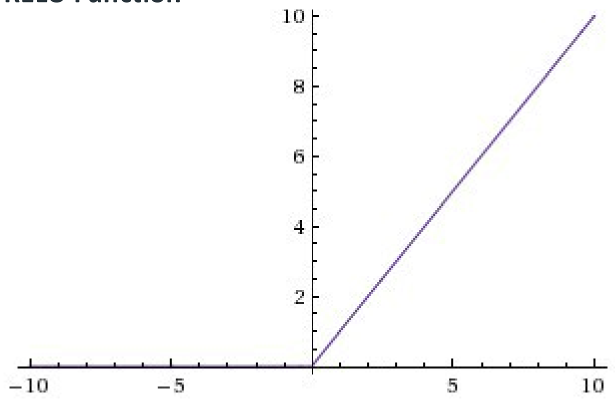
- The activation that works almost always better than sigmoid function is Tanh function also known as **Tangent Hyperbolic function**. It's actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.

- **Equation:-**

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

- **Value Range :-** -1 to +1
- **Nature :-** non-linear
- **Uses :-** Usually used in hidden layers of a neural network as it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.

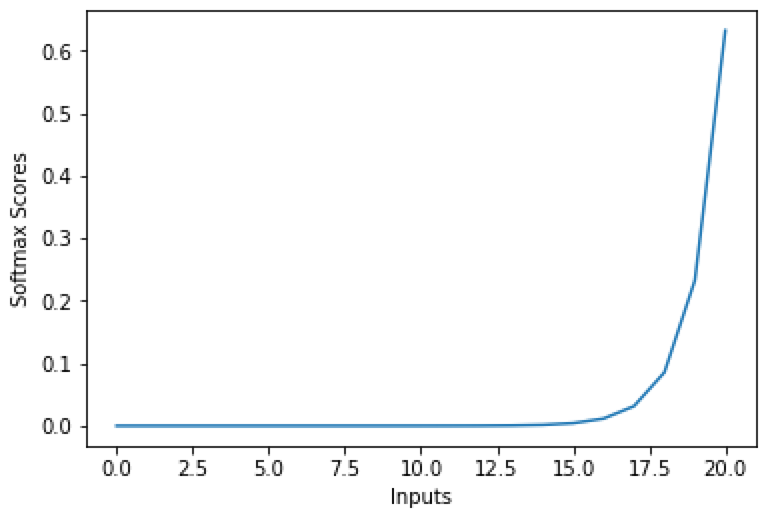
RELU Function



- It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.
- **Equation :-** ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
- **Value Range :-** [0, inf)
- **Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.
- **Uses :-** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

In simple words, RELU learns *much faster* than sigmoid and Tanh function.

Softmax Function



The softmax function is also a type of sigmoid function but is handy when we are trying to handle multi- class classification problems.

- **Nature :-** non-linear
- **Uses :-** Usually used when trying to handle multiple classes. the SoftMax function was commonly found in the output layer of image classification problems. The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.
- **Output:-** The SoftMax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.
- The basic rule of thumb is if you really don't know what activation function to use, then simply use *RELU* as it is a general activation function in hidden layers and is used in most cases these days.
- If your output is for binary classification then, *sigmoid function* is very natural choice for output layer.
- If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each class.

9. What if we do not use any activation function in neural network?

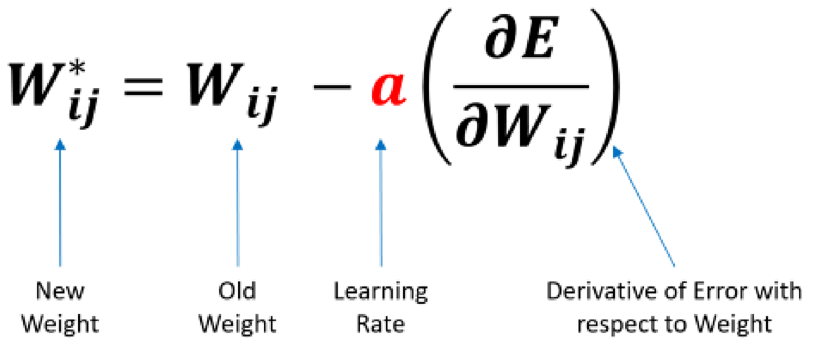
We understand that using an activation function introduces an additional step at each layer during the forward propagation. Now the question is – if the activation function increases the complexity so much, can we do without an activation function?
Imagine a neural network without the activation functions. In that case, every neuron will only be performing a linear transformation on the inputs using the weights and biases. Although linear transformations make the neural network simpler, but this network would be less powerful and will not be able to learn the complex patterns from the data.
A neural network without an activation function is essentially just a linear regression model.
Thus we use a non linear transformation to the inputs of the neuron and this non-linearity in the network is introduced by an activation function.

11. What is dying Relu problem?

The dying ReLU problem occurs when several neurons only output a value of zero. This happens primarily when the input is negative. This offers an advantage of network sparsity to ReLU, but it creates a major problem when most of the inputs to the neurons are negative. It basically leads to a worst-case scenario when the entire network dies and only a constant function remains.
When most of the neurons output zero, the gradient fails to flow and the weights stop getting updated. Thus, the network stops learning. As the slope of ReLU activation function is zero in the negative input range, once it becomes dead, it is impossible to recover the network to learn.
This dying ReLU problem does not occur quite often because optimizers feed a variety of inputs to the network where in, not all the inputs are in the negative range. As long as the inputs have some positive values, some neurons are active and keep learning as the gradients keeps flowing through the network.

Causes of the Dying ReLU Problem:

1. *High Learning Rate* - Let us have a look at the weight updating equation first:


$W^*_{ij} = W_{ij} - a\left(\frac{\partial E}{\partial W_{ij}}\right)$
New Weight
Old Weight
Learning Rate
Derivative of Error with respect to Weight

The input to activation function is: *(W*x) + b.* If the learning rate, alpha value is quite high, it leads to a high probability of the new weights being negative as quite large negative value gets subtracted from our old weights. This further leads to negative inputs to the ReLU function and ultimately causes dying ReLU problem!

2. *Higher Negative Bias* - While we have mostly taked about negative inputs and higher learning rate, we must not forget that the bias value also gets fed as input when added to the products of inputs and weights. This larger negative bias value might also lead to negative inputs to ReLU function and thus cause dying ReLU problem.

12. What is the significance of loss function and cost function? Bifurcate function on the basis of regression and classification problems?

Loss Function	**Cost Function**
Measures the error between predicted and actual values in a machine learning model.	Quantifies the overall cost or error of the model on the entire training set.
Used to optimize the model during training.	Used to guide the optimization process by minimizing the cost or error.
Can be specific to individual samples.	Aggregates the loss values over the entire training set.
Examples include mean squared error (MSE), mean absolute error (MAE), and binary cross-entropy.	Often the average or sum of individual loss values in the training set.
Used to evaluate model performance.	Used to determine the direction and magnitude of parameter updates during optimization.

Loss Function	Cost Function
Different loss functions can be used for different tasks or problem domains.	Typically derived from the loss function, but can include additional regularization terms or other considerations.

1. Regression
   - MSE(Mean Squared Error)
   - MAE(Mean Absolute Error)
   - Hubber loss
2. Classification
   - Binary cross-entropy
   - Categorical cross-entropy
3. Autoencoder
   - KL Divergence
4. GAN
   - Discriminator loss
   - Minmax GAN loss
5. Object detection
   - Focal loss
6. Word embeddings
   - Triplet loss

13. What do you understand by forward propagation and backward propagation?

In order to be trained, a neural network relies on both forward and backward propagation. Backpropagation is used in machine learning and data mining to improve prediction accuracy through backward propagation calculated derivatives. Backward Propagation is the process of moving from right (output layer) to left (input layer). Forward propagation is the way data moves from left (input layer) to right (output layer) in the neural network.
A neural network can be understood by a collection of connected input/output nodes. The accuracy of a node is expressed as a loss function or error rate. Backpropagation calculates the slope of a loss function of other weights in the neural network.

14. What are optimizers and its function? State different types of optimizers and their core fundamentals?

https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0#:~:text=Optimizers%20are%20algorithms%20or%20methods,learnable%20parameters%20i.e%20Weights%20%26%20Biases.

https://www.upgrad.com/blog/types-of-optimizers-in-deep-learning/

15. Explain the Adam Optimizer algorithm with EMA concept and dynamic learning rate?

https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c

16. What are vanishing and exploding gradients and how these problems can be fixed?

https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/

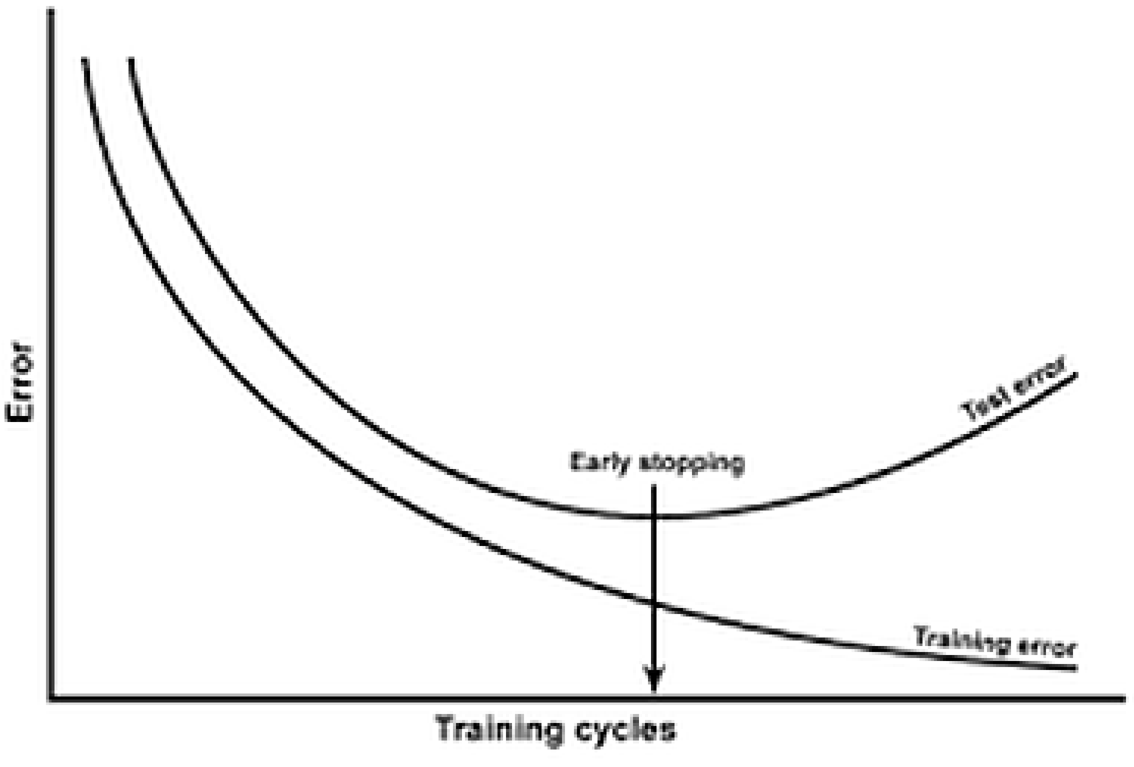
17. Can neural network get overfitted? What is Dropout in neural network? And how it avoids overfitting problem?

1. Simplifying the Model

The first step when dealing with overfitting is to decrease the complexity of the model. To decrease the complexity, we can simply remove layers or reduce the number of neurons to make the network smaller. While doing this, it is important to calculate the input and output dimensions of the various layers involved in the neural network. There is no general rule on how much to remove or how large your network should be. But, if your neural network is overfitting, try making it smaller.

2. Early Stopping

Early stopping is a form of regularization while training a model with an iterative method, such as gradient descent. Since all the neural networks learn exclusively by using gradient descent, early stopping is a technique applicable to all the problems. This method update the model so as to make it better fit the training data with each iteration. Up to a point, this improves the model's performance on data on the test set. Past that point however, improving the model's fit to the training data leads to increased generalization error. Early stopping rules provide guidance as to how many iterations can be run before the model begins to overfit.

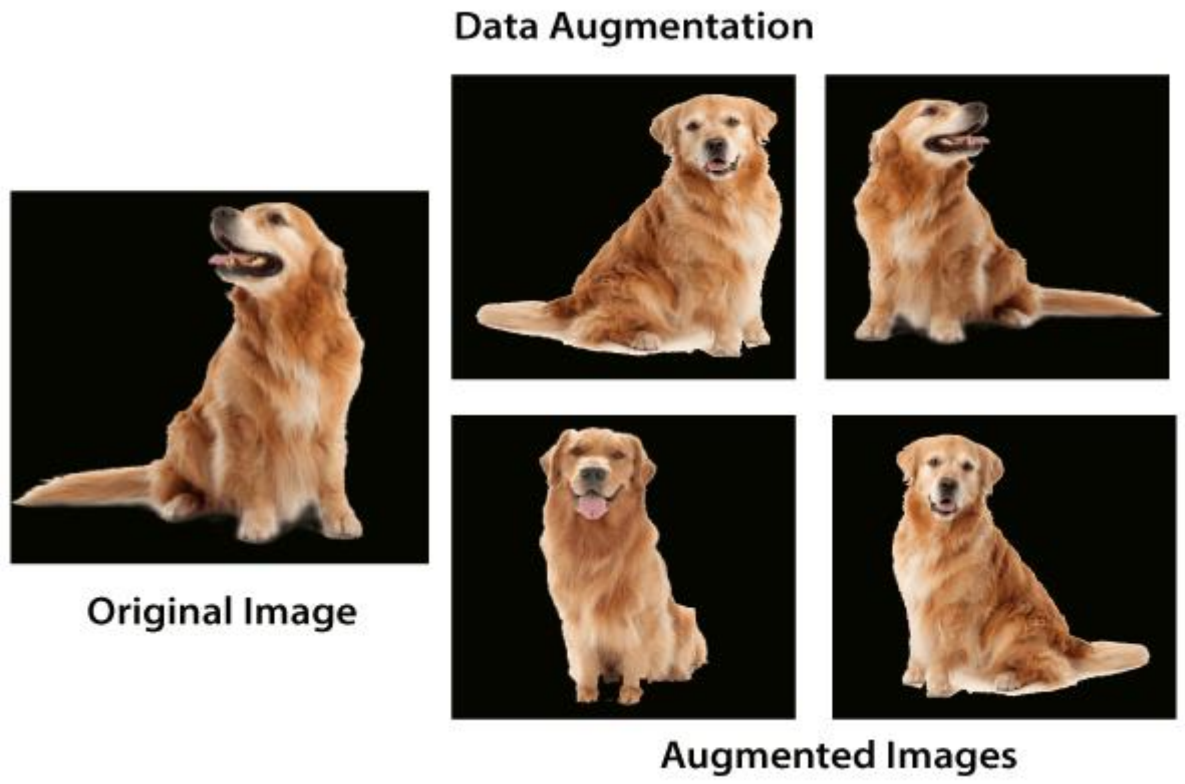


Early Stopping

This technique is shown in the above diagram. As we can see, after some iterations, test error has started to increase while the training error is still decreasing. Hence the model is overfitting. So to combat this, we stop the model at the point when this starts to happen.

3. Use Data Augmentation

In the case of neural networks, data augmentation simply means increasing size of the data that is increasing the number of images present in the dataset. Some of the popular image augmentation techniques are flipping, translation, rotation, scaling, changing brightness, adding noise etcetera. For a more complete reference, feel free to checkout albumentations and imgaug.



Data Augmentation

This technique is shown in the above diagram. As we can see, using data augmentation a lot of similar images can be generated. This helps in increasing the dataset size and thus reduce overfitting. The reason is that, as we add more data, the model is unable to overfit all the samples, and is forced to generalize.

4. Use Regularization

Regularization is a technique to reduce the complexity of the model. It does so by adding a penalty term to the loss function. The most common techniques are known as L1 and L2 regularization:

- The L1 penalty aims to minimize the absolute value of the weights. This is mathematically shown in the below formula.

$$L(x,y) \equiv \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \boxed{\lambda \sum_{i=1}^{n}|\theta_i|}$$

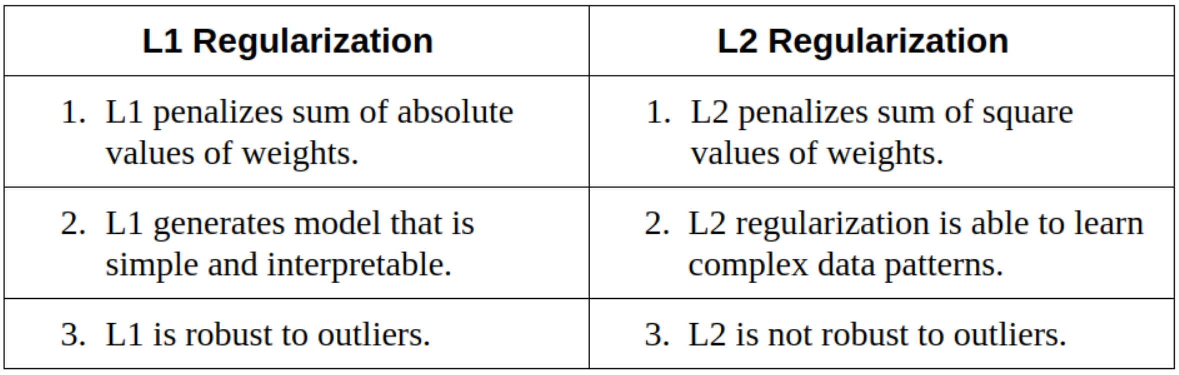
L1 Regularization

- The L2 penalty aims to minimize the squared magnitude of the weights. This is mathematically shown in the below formula.

$$L(x,y) \equiv \sum_{i=1}^{n}(y_i - h_\theta(x_i))^2 + \boxed{\lambda \sum_{i=1}^{n} \theta_i^2}$$

L2 Regularization

The below table compares both the regularization techniques.



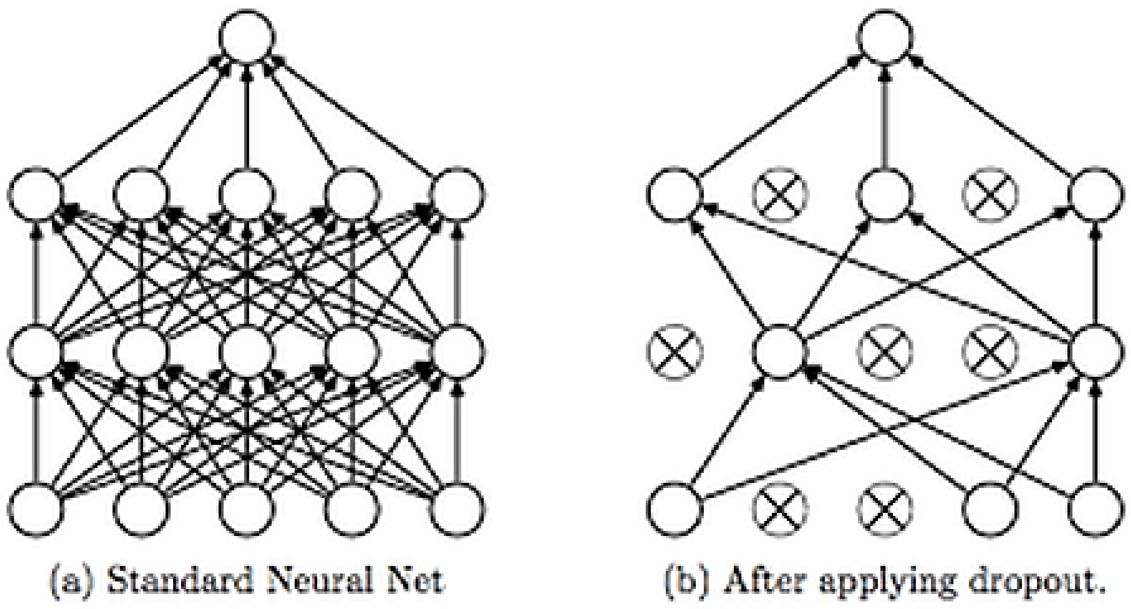
L1 Regularization	L2 Regularization
1. L1 penalizes sum of absolute values of weights.	1. L2 penalizes sum of square values of weights.
2. L1 generates model that is simple and interpretable.	2. L2 regularization is able to learn complex data patterns.
3. L1 is robust to outliers.	3. L2 is not robust to outliers.

L1 vs L2 Regularization

So which technique is better at avoiding overfitting? The answer is — it depends. If the data is too complex to be modelled accurately then L2 is a better choice as it is able to learn inherent patterns present in the data. While L1 is better if the data is simple enough to be modelled accurately. For most of the computer vision problems that I have encountered, L2 regularization almost always gives better results. However, L1 has an added advantage of being robust to outliers. So the correct choice of regularization depends on the problem that we are trying to solve.

5. Use Dropouts

Dropout is a regularization technique that prevents neural networks from overfitting. Regularization methods like L1 and L2 reduce overfitting by modifying the cost function. Dropout on the other hand, modify the network itself. It randomly drops neurons from the neural network during training in each iteration. When we drop different sets of neurons, it's equivalent to training different neural networks. The different networks will overfit in different ways, so the net effect of dropout will be to reduce overfitting.



Using Dropouts

This technique is shown in the above diagram. As we can see, dropouts are used to randomly remove neurons while training of the neural network. This technique has proven to reduce overfitting to a variety of problems involving image classification, image segmentation, word embeddings, semantic matching etcetera.

18. What is weight initialization and types of weight initialization?

1. Zero Initialization
As the name suggests, all the weights are assigned zero as the initial value is zero initialization. This kind of initialization is highly ineffective as neurons learn the same feature during each iteration. Rather, during any kind of constant initialization, the same issue happens to occur. Thus, constant initializations are not preferred.

2. Random Initialization
In an attempt to overcome the shortcomings of Zero or Constant Initialization, random initialization assigns random values except for zeros as weights to neuron paths. However, assigning values randomly to the weights, problems such as Overfitting, Vanishing Gradient Problem, Exploding Gradient Problem might occur.
Random Initialization can be of two kinds:

- Random Normal
- Random Uniform

a) Random Normal: The weights are initialized from values in a normal distribution.
b) Random Uniform: The weights are initialized from values in a uniform distribution.

3.Xavier/Glorot Initialization
In Xavier/Glorot weight initialization, the weights are assigned from values of a uniform distribution as follows: Xavier/Glorot Initialization often termed as Xavier Uniform Initialization, is suitable for layers where the activation function used is **Sigmoid.**
4. Normalized Xavier/Glorot Initialization
In Normalized Xavier/Glorot weight initialization, the weights are assigned from values of a normal distribution as follows: Here, is given by:Xavier/Glorot Initialization, too, is suitable for layers where the activation function used is Sigmoid.

6. He Normal Initialization
In He Normal weight initialization, the weights are assigned from values of a normal distribution as follows: Here, \sigma is given by:He Uniform Initialization, too, is suitable for layers where **ReLU** activation function is used

https://medium.com/guidona-softpedia/weight-initialization-methods-in-neural-networks-a3e7a793cee5

19. What techniques should be avoided during weight and bias initialization?

. Zero Initialization
As the name suggests, all the weights are assigned zero as the initial value is zero initialization. This kind of initialization is highly ineffective as neurons learn the same feature during each iteration. Rather, during any kind of constant initialization, the same issue happens to occur. Thus, constant initializations are not preferred.

2. Random Initialization
In an attempt to overcome the shortcomings of Zero or Constant Initialization, random initialization assigns random values except for zeros as weights to neuron paths. However, assigning values randomly to the weights, problems such as Overfitting, Vanishing Gradient Problem, Exploding Gradient Problem might occur.
Random Initialization can be of two kinds:

- Random Normal
- Random Uniform

a) Random Normal: The weights are initialized from values in a normal distribution.
b) Random Uniform: The weights are initialized from values in a uniform distribution.

20. What is Early stopping?

Early stopping is a method that allows you to specify an arbitrarily large number of training epochs and stop training once the model performance stops improving on the validation dataset.
https://towardsdatascience.com/early-stopping-a-cool-strategy-to-regularize-neural-networks-bfdeca6d722e

21. What is Data normalisation? And different types of Data Normalisation?

Normalizing a set of data transforms the set of data to be on a similar scale. For machine learning models, our goal is usually to recentre and rescale our data such that is between 0 and 1 or -1 and 1, depending on the data itself. One common way to accomplish this is to calculate the mean and the standard deviation on the set of data and transform each sample by subtracting the mean and dividing by the standard deviation, which is good if we assume that the data follows a normal distribution as this method helps us standardize the data and achieve a standard normal distribution. Normalization can help training of our neural networks as the different features are on a similar scale, which helps to stabilize the gradient descent step, allowing us to use larger learning rates or help models converge faster for a given learning rate.
https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/

22. Why is Deep neural network better than a shallow neural network?

Universal approximation theorem says, "Neural network with a hidden layer can represent any functions" (with enough and proper parameters). However, we don't have enough data for our problems most of the time. If we do, we may solve those problems with a look-up table. Therefore, we need machine learning or deep learning to learn the approximate functions to solve the problem. It's been shown that deep networks outperform shallow networks given the same number of parameters and training data; deep networks need exponentially fewer parameters and sample complexity to achieve similar performance [2]. But why?
One hidden layer is applying the non-linear activation function to the linear combination of the input. Each hidden layer can be viewed as a modular function. A deep network of many hidden layers is like a stack of multiple functions, which can achieve more complex functions with the same number of parameters compared to a shallow network. In other words, deep networks utilize parameters more efficiently. To be noted, if the training data are excessively enough or the problem is easy enough for shallow networks to learn, then you may see similar performance between deep and shallow networks.
For the example of image classification and convolutional neural networks, initial layers learn the low-level features like edges and curves and later layers learn high-level features that reuse low-level features to build up the shape of the object [3]. A similar phenomenon was found in speech recognition [4]. Initial layers learn the manner of articulation and later layers learn different phonemes based on the combinations of different low-level features. Therefore, deep networks with many hidden layers make it easier to learn such high-level features than shallow networks.
Summary
In summary, deep networks outperform shallow networks (with the same amount of parameters and data) because multiple hidden layers lead to reusable modular functions, which enable better efficiency in utilizing parameters.

23. What is CNN and explain different layers in CNN?

It has three layers namely, convolutional, pooling, and a fully connected layer. It is a class of neural networks and processes data having a grid-like topology. The convolution layer is the building block of CNN carrying the main responsibility for computation. Pooling reduces the spatial size of the representation and lessens the number of computations required. Whereas, the Fully Connected Layer is connected to both the layers, prior and the recent one.

24. List few applications of CNN?

Convolutional Neural Network is a type of deep learning neural network that is artificial. It is employed in computer vision and image recognition. This procedure includes the following steps:

- OCR and image recognition
- Detecting objects in self-driving cars
- Social media face recognition
- Image analysis in medicine

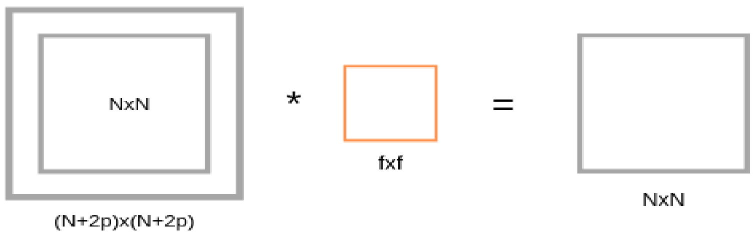
The term "convolutional" refers to a mathematical function that is created by integrating two different functions. It usually involves multiplying various elements to combine them into a coherent whole. Convolution describes how the shape of one function is influenced by another function. In other words, it is all about the relationships between elements and how they work together.

25. Explain filters, padding and strides in detail?

https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529#:~:text=3.3%20Stride%20and%20Padding,By%20default%20it%20is%20one.&text=We%20can%20observe%20that%20the,to%20the%20input%20matrix%20symmetrically.

26. Explain in detail about the tern valid padding and same padding?

Valid Padding: This type is used when there is no requirement for Padding. The output matrix after convolution will have the dimension of (n – f + 1) X (n – f + 1).
Same Padding: Here, we added the Padding elements all around the output matrix. After this type of padding, we will get the dimensions of the input matrix the same as that of the convolved matrix.



After Same padding, if we apply a filter of dimension f x f to (n+2p) x (n+2p) input matrix, then we will get output matrix dimension **(n+2p-f+1) x (n+2p-f+1)**. As we know that after applying Padding we will get the same dimension as the original input dimension (n x n). Hence we have,
(n+2p-f+1)x(n+2p-f+1) equivalent to nxn
n+2p-f+1 = n
p = (f-1)/2
So, by using Padding in this way we don't lose a lot of information and the image also does not shrink.

27. What is pooling and different types of pooling?

Pooling is performed in neural networks to reduce variance and computation complexity. Many a times, beginners blindly use a pooling method without knowing the reason for using it. Here is a comparison of three basic pooling methods that are widely used.
The three types of pooling operations are:

1. Max pooling: The maximum pixel value of the batch is selected.
2. Min pooling: The minimum pixel value of the batch is selected.
3. Average pooling: The average value of all the pixels in the batch is selected.

https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

28. Explain the role of flattening and fully connected layer in CNN?

Flattening is converting the data into a 1-dimensional array for inputting it to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. And it is connected to the final classification model, which is called a fully-connected layer.

https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480#:~:text=Flattening%20is%20converting%20the%20data,called%20a%20fully%2Dconnected%20layer.

29. What is Data Augmentation?

Data Augmentation is a set of techniques that enable AI teams to artificially generate new data points from the data that already exists. This practice includes making small changes to the data (which could either be a text, audio, or visual), generating diverse instances, and expanding the data set to have improved the performance and outcome of the deep learning model.
For example, Data augmentation methods reduce data overfitting which significantly improves the accuracy rate and generalizes the output of a model.
Data overfitting happens when a model is trained too well for a set of data. In this case, the model learns the noise and detail of the data to an extent that it starts to impact the performance of the model on new data. This means that the noise and fluctuations in the training data are learned as concepts by the model.
When a new set of data is added to the model, the learned concepts do not apply to it, which negatively impacts the ability of a model to generalize. This is generally the problem with small data sets. The smaller the data sets, the more is the control of networks over it. But when the size of a data increases through augmentation, the network doesn’t overfit the training data set and is thus forced to generalize.
Data augmentation technique is adopted in almost every deep learning application such as image classification,, natural language processing, image recognition, semantic segmentation, etc.

https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/

30. What are the Sequential API and Functional API?

https://intuitivetutorial.com/2023/05/18/sequential-api-and-functional-api-in-keras/