Generally there will be a visible layer ‘X’ (features as input), every feature has a weight ‘w’ associated and the dot product of the weight and input features will be fed to the Hidden layer h1. So h1 will have the dot product of the weight and input adding extra bias b.

For the Hidden layer h1 all inputs are fully connected. And hidden layer will have many nodes. Each node will have input as XW and at the output it will have an activation function.(RELU,TANH,SOFTMAX).

So like this there can be many number of hidden layers with the same like inputs and outputs. All outputs from h1 will be input to the next hidden layer h2.

At the end of the network we will have Output layer which will have all h2 outputs as inputs. The nodes in the output layer is directly depends on the number of different classes in the target variable.

Once the output is derived from the output layer, then there is a process to check the actual output y and the difference between actual output and expected output y hat. So that is [y true – y pred].

We will have a threshold to decide the output. If the output difference is too large, then network will back propagate and change weight values. Again with the changed weights the all above process continues.

While back propagating the system learns, and tries to converge at global minima. We need to find the best check point at which the network is converging, here for each step, there will be learning rate, and Learning rate is the hyper parameter that we need to choose very carefully otherwise there are many chances that it overshoots the local minima. So depends on the learning rate it converges at the best possible point.

While feeding in the hidden layer to output layer, we wantedly drop 10% of the input nodes, this we do to reduce the overfitting problem that may occur in future. So dropout is also one of the hyper parameters.

While defining the cost function which measures the difference between actual and expected, we use a entropy function for classification we use classification cross entropy and for regression we use binary cross entropy. And the Final objective of the neural net is to optimize the loss/cost function. We use either Adam Optimizer or Gradient Descent Optimizer,

We add dropouts, early stopping for overfitting problems,

We also do Regularization L2 regularization for better scaled results.

Some common questions and answers:

1. How many Hidden layers?

Ans: To solve a non-linear problem we need a hidden layers.

1. How many Nodes/Neurons?

Ans: For a three layer network with n input and m output neurons, the hidden layer would have square root of n∗m neurons.

n = 5 inputs

m = 10 outputs

Hidden layer will have square root of 5\*10 = square root of 50 neurons.

How many Output Layers?

What is batch Size?

1. How many Epochs?

Ans: If the network is trained on all training samples once it is called one Epoch. Number of epochs (n\_iter). Epoch is defined as one forward pass and one backward pass of all training data.

What is default Learning Rate?

Learning rate “η” (eta). Default is 10 power -3. It determines how fast we want to update the weights during optimization.  if learning rate is too small, gradient descent can be slow to find the minimum and if it’s too large gradient descent may not converge(it can overshoot the minima). It’s considered to be the most important hyper parameter.

Default Dropout Rate?

Dropout is a regularization technique for reducing overfitting in neural networks. At each training step we randomly drop out (set to zero) set of nodes, thus we create a different model for each training case, all of these models share weights. It’s a form of model averaging.

Default rate is 10% or 0.1.

When to perform early stopping?

A major challenge in training neural networks is how long to train them.

Too little training will mean that the model will under fit the train and the test sets. Too much training will mean that the model will over fit the training dataset and have poor performance on the test set.

A compromise is to train on the training dataset but to stop training at the point when performance on a validation dataset starts to degrade. This simple, effective, and widely used approach to training neural networks is called early stopping.

Trigger for Early Stopping:

In the simplest case, training is stopped as soon as the performance on the validation dataset decreases as compared to the performance on the validation dataset at the prior training epoch (e.g. an increase in loss).

Some more elaborate triggers may include:

* No change in metric over a given number of epochs.
* An absolute change in a metric.
* A decrease in performance observed over a given number of epochs.
* Average change in metric over a given number of epochs.

Some delay or “*patience*” in stopping is almost always a good idea.

### Plot Learning Curves to Select a Trigger

### Loss is an easy metric to monitor during training and to trigger early stopping.

When to use Regularization? What all Regularizations are there?

When the model failed to generalize on test data, then it means the model is over fitted. To avoid over fitting we generally introduce small error into the model in the form of Regularization. We add regularization to the end of the cost function. We add Regularization Function (Lambda).

What are different loss/cost functions? J (W)

Ans: Cost function tells us how well the neural network is performing. Cost function is called the Cross-entropy cost function.

Back Propagation:

Backpropagation is essentially a computationally efficient approach to compute the partial derivatives of a complex cost function in multilayer neural networks.

Difference between Activation and cost functions?

Evaluation Metrics after we built the Model?

How to save and Restore the model?

What is placeholder and what is a variable?

How to measure Training Accuracy and Validation Accuracy?

In Industries, Actually Data Scientists will set a goal to Validation loss.

How many maximum dimensions tensor flow will support?

It depends it has no limit. Not necessarily.

What is Overfitting and under fitting? Techniques to remove them?

Ans: Model memories when the data is small size, and fails to generalize on unseen data. This is called Overfitting. When we see no or less error during Training Process and see more error during Testing on unseen data, this problem is called overfitting. Overfitting is a scenario where the performance (error) is less on trained data and high on test data.

Overfitting can be handled by:

Increase the dataset size will decrease overfitting.

Use Regularization methods e.g. L2. In Regularization techniques, we introduce a small error to the cost function, so that the model will not over fit. We actually add the Regularization function to the end of the cost function. We use lambda symbol to represent the regularization function.

What is Variance and Bias Tradeoff?

The prediction error for any machine learning algorithm can be broken down into three parts:

* Bias Error
* Variance Error
* Irreducible Error

**Bias Error**

Bias are the simplifying assumptions made by a model to make the target function easier to learn.

Examples of low-bias machine learning algorithms include: Decision Trees, k-Nearest Neighbors and Support Vector Machines.

Examples of high-bias machine learning algorithms include: Linear Regression, Linear Discriminant Analysis and Logistic Regression.

## Variance Error

Variance is the amount that the estimate of the target function will change if different training data was used.

The goal of any supervised machine learning algorithm is to achieve low bias and low variance.

* The k-nearest neighbors’ algorithm has low bias and high variance, but the trade-off can be changed by increasing the value of k which increases the number of neighbors that contribute t the prediction and in turn increases the bias of the model.
* The support vector machine algorithm has low bias and high variance, but the trade-off can be changed by increasing the C parameter that influences the number of violations of the margin allowed in the training data which increases the bias but decreases the variance.

There is no escaping the relationship between bias and variance in machine learning.

* Increasing the bias will decrease the variance.
* Increasing the variance will decrease the bias.

What is Vanishing Gradient Problem / Exploding Gradient ?

Deeper layers can lead to vanishing gradient problem. Decay of information through time. **Vanishing Gradient** Problem occurs when we try to train a Neural Network model using **Gradient based optimization techniques**.

When there are many hidden layers, during back propagation it will be very difficult to compute the gradient loss ( Weights) and to update. This makes the training process takes long time.

Generally Sigmoid and Tanh will produce Vanshing Gradient problem. Now a days we use RELU activation functions which will not produce Vanishing Gradients problem.

1. What are weights and how to change weights?

Ans: Weights are also called parameters or coefficients. Parameters are in network control.

Weights are updated by the following formula.

W=W-ng.

n = eta (Learning rate will be between 0 and 1).

g = gradient.

1. Notion of all Neural Networks?

Ans:

<https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/>

**Best link**

What is transfer Learning?

This we can start with a nice Teacher, Student analogy. A teacher will have many year’s experience in a specific field of interest. In the lectuers, the teacher can teach brief introduction and the subject to the teacher using his vast knowledge and experience.

If someone has already solved a similar problem which we want to solve, we can use that pre trained model and tailor according to our requirement.

Approach : we can remove the output layer, and can still use it for feature extractions.

Eg., the human language has been evolved for generations. Each generation transfers the knowledge they have to another generation. This is Transferred learning.

<https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/>

Different Activation Functions:

As a Thumb rule starting with RELU function at hidden layers and

Soft max at output layer for classification

Sigmoid only for binary classification output layer

Binary Step Function f(x) = 1, x>=0 = 0, x<0,

Sigmoid Activation Function f(x)=1/(1+e^-x). Ranges from 0 to 1 σ(x)=1/(1+e−x)

Tanh Activation Function (Scaled Version of Sigmoid) tanh(x)=2/(1+e^(-2x)) -1

Tanh ranges from -1 to 1 tanh(x)=2σ(2x)−1 .

ReLu Activation Function. f(x)=max(0,x). Relu should be used only inside the Hidden layers

Leaky Relu: To account smaller negative values as smaller portion of x. <https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/>

## Choosing the right Activation Function

* Sigmoid functions and their combinations generally work better in the case of classifiers
* Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
* ReLU function is a general activation function and is used in most cases these days
* If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice
* Always keep in mind that ReLU function should only be used in the hidden layers
* As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn’t provide with optimum results