Practical issues in neural Network Training

- While training a neural network, Gradient descent algorithm is used to train the model
- The aim is to minimize the cost function to achieve the optimal values for the model parameters ie, to reach a global minima.
- But there is a global optimum but at the same time, there are multiple local optima.
- Depending on the type of Neural Network, there are a handful of hyperparameters you need to tune to reach the global optimum.

Weight Symmetry

- When all the weights in Neural Network have identical values for successive steps of backpropagation
- This phenomenon occurs when we initialize all the weights & biases with a common value
- Thus, all the hidden units will have identical activation.
- This in turn will result in identical derivatives during backpropagation & hence identical weights in the next step.
- no learning will happen.

- The problem can be solved by random initialization of these parameters.

Slow Progress

- Occurs when small learning rate is selected
- the gradient descent makes slow progress towards the global optimum thus increasing the computational cost.
- The solution is to increase the learning rate.

Instability and Oscillations

- If the learning rate is too high, the gradient descent will overshoot
- Sometimes, the overshoot gets larger with each step & quickly blows up.
- This phenomenon is known as Instability
- When the learning rate is large, yet not large enough to cause Instability, the large weight updates occurs.
- This, in turn, will cause the weights to diverge & will result in Oscillations over subsequent training epochs.

- Solution
  - tune the learning rate by gradually decreasing it.

Local Optima

- While minimizing the cost function, sometimes get stuck in a local optimum.
- The set of weights that lead to a given local optimum are known as a Basin of Attraction.
- **Solution: Random Restarts**.
- initialize the training from several random values, train the model with each set of values, & pick whichever gives the lowest cost.

The Problem of Overfitting

- the model tries to learn too many details in the training data along with the noise from the training data.
- . As a result, the model performance is very poor
- Thus the network fails to generalize the features or patterns present in the training dataset.

- Overfitting during training can be spotted when the error on training data decreases to a very small value but the error on the new data or test data increases to a large value.

Reasons for Overfitting

- **The size of the training dataset is small**
- **The model tries to make predictions on Noisy Data**

- Underfitting happens when the network can neither model the training or test data which results in overall bad performance.

- Deep neural networks are prone to overfitting because they learn millions or billions of parameters while building the model.
- A model having this many parameters can overfit the training data
- The basic idea to deal with the problem of overfitting is to decrease the complexity of the model
- Make the network smaller by removing the layers or reducing the number of neurons, etc.

- Another technique for reducing overfitting is to increase the size of the training dataset.
- when the size of the training data is small, then the network tends to have greater control over the training data.
- To increase the size of the training data, use data augmentation, which is the easiest way to diversify our data and make the training data larger.
- Eg: image augmentation techniques are flipping, translation, rotation, scaling, changing brightness, adding noise

Weight Regularization

- Aims to stabilize an overfitted network by adding a weight penalty term, which penalizes the large value of weights in the network.
- An overfitted model has problems with a large value of weights as a small change in the input can lead to large changes in the output.
- Weight regularization penalizes the network's large weights & forces the optimization algorithm to reduce the larger weight values to smaller weights, and this leads to stability of the network & presents good performance.
- In this technique, the network configuration remains unchanged since it only modifies the value of the weights.

Vanishing and exploding gradient problems

- As the backpropagation algorithm advances downwards(or backward) from the output layer towards the input layer, the gradients often get smaller and smaller and approach zero which eventually leaves the weights of the initial or lower layers nearly unchanged
- As a result, the gradient descent never converges to the optimum. This is known as the ***vanishing gradients*** problem.

- In some cases, the gradients keep on getting larger and larger as the backpropagation algorithm progresses.
- This, in turn, causes very large weight updates and causes the gradient descent to diverge. This is known as the ***exploding gradients*** problem.

How to know if our model is suffering from the Exploding/Vanishing gradient problem?

Exploding	Vanishing
There is an exponential growth in the model parameters.	The parameters of the higher layers change significantly whereas the parameters of lower layers would not change much (or not at all).
The model weights may become NaN during training.	The model weights may become 0 during training.
The model experiences avalanche learning.	The model learns very slowly and perhaps the training stagnates at a very early stage just after a few iterations

Solutions

- **Proper Weight Initialization**
- **Using Non-saturating Activation Functions**
- **Batch Normalization**
- **Gradient Clipping**
  - mitigate the exploding gradients problem is to clip the gradients during backpropagation so that they never exceed some threshold.
  - This is called Gradient Clipping.