Xavier/Glorat

Key Insights about Things not to do..

1.) Zero Initialization
    No Training
2.) Non-Zero Constant Initialization

3.) Random Initialization with Small Weights
4.) Random Initialization with Large Weights
    VGD milta hai 3rd & 4th scene mai
    So Humme weights chaye random but specific , with good range or variance.

TO Solve this Genius people lyk me have made some heuristics means jugaad,

Like

1.) Xavier or Glorat

In this → Uniform Verson & Normal Version

&

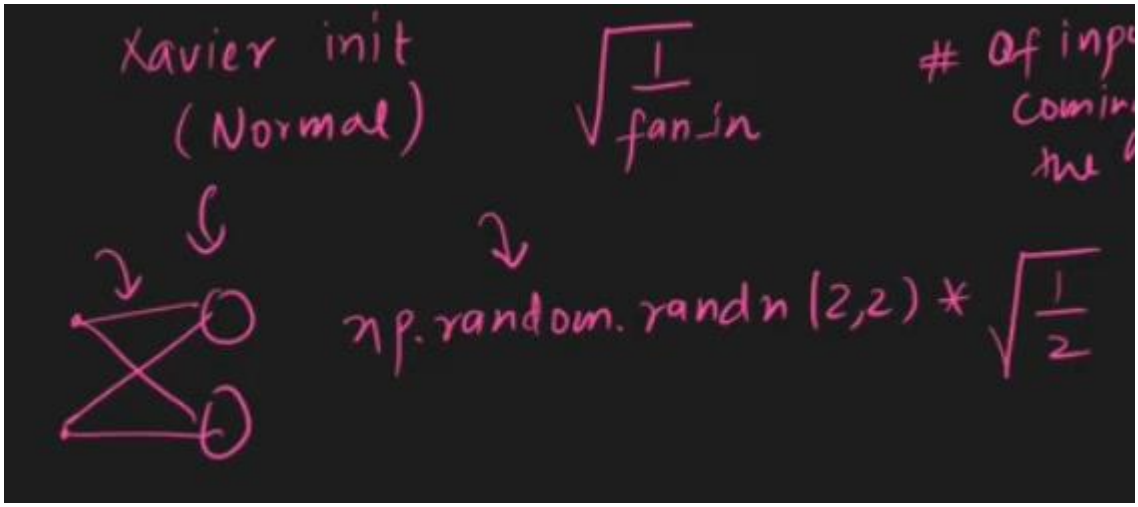
2.) He Initialization

Normal Version & Uniform Version.

If you are working on tanh then you must use Xavier Initialization

If you are working on ReLU then you must use He Initialization

This all has experimental proven.



Xavier/Glorat Initialization

Xavier Initialization, named after Xavier Glorot, is a method to set the initial weights of a neural network in a smart way. The goal is to prevent the issue of vanishing or exploding gradients during training.

In simpler terms:

1. **Problem it Solves:** During training, if the weights are too small, the signal diminishes as it goes through the network (vanishing gradients). If the weights are too large, the signal can grow uncontrollably (exploding gradients).
2. **Idea Behind It:** Xavier Initialization sets the initial weights in a way that maintains a balance, preventing the gradients from becoming too small or too large. This helps the network learn more effectively.
3. **How It's Done:** We calculate the bounds for initializing the weights using a formula that takes into account the number of input and output units in a layer. The weights are then randomly set within these bounds.

In summary, Xavier Initialization is a technique to give our neural network a good starting point for learning, making it more likely to train effectively without encountering problems related to gradient scaling.

OR

Xavier initialization, also known as Glorot initialization, is a method used to initialize the weights of a neural network in deep learning. It is named after Xavier Glorot, who proposed this technique to address the issue of vanishing or exploding gradients during training.

The idea behind Xavier initialization is to set the initial weights of the neural network in such a way that the variance remains approximately the same across different layers. This helps in preventing the gradients from becoming too small or too large during backpropagation, which can impede the learning process.

The Xavier initialization is typically applied to the weights of the network, and the initialization is done by drawing random values from a specific distribution. The formula for Xavier initialization is as follows:

$$W \sim \text{Uniform}(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}})$$

Here, W represents the weights of the neural network, n_{in} is the number of units in the layer's input, and n_{out} is the number of units in the layer's output. The weights are drawn from a uniform distribution with the specified bounds.

He Initialization (also known as Kaiming Initialization):

- It is specifically designed for networks using Rectified Linear Unit (ReLU) activation functions.
- The idea is similar to Xavier, but the scaling factor is adjusted for ReLU activation to better handle the characteristics of ReLU.

Why We Use initialization techniques what problems it solves ?

Initialization techniques in neural networks are used to address two primary problems during training: vanishing gradients and exploding gradients. These problems can hinder the convergence and stability of the learning process.

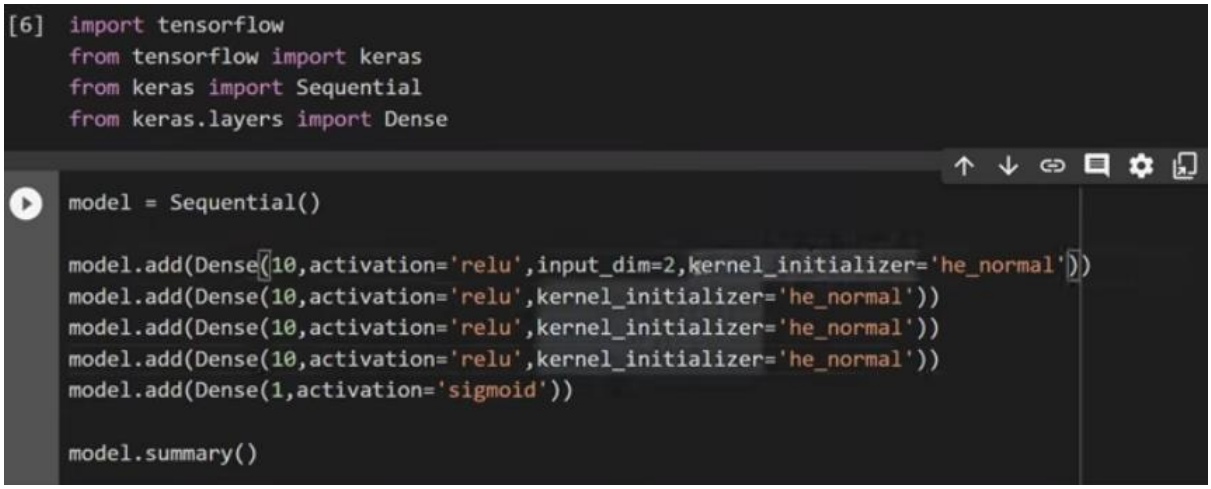
1. **Vanishing Gradients:**
    - In deep neural networks, during backpropagation, gradients are propagated backward through the network to update the weights. If the weights are initialized to very small values, the gradients can become extremely small as they propagate through the layers.
    - As a result, the weights may not be updated effectively, and the network may have difficulty learning from the training data. This is known as the vanishing gradient problem.
2. **Exploding Gradients:**
    - Conversely, if the weights are initialized to very large values, the gradients can explode during backpropagation. This means that the gradients become very large, causing unstable updates to the weights.
    - The exploding gradient problem can lead to oscillations in the training process, and it may prevent the network from converging to a good solution.

Initialization techniques aim to set the initial weights of the network in a way that mitigates these issues. By providing a suitable starting point for the optimization process, these techniques help ensure stable and efficient training. The specific challenges addressed by initialization techniques include:

- **Effective Learning Rates:** Proper initialization helps ensure that the gradients are neither too small (leading to slow learning) nor too large (leading to instability).
- **Balanced Signal Propagation:** Initializing weights appropriately helps maintain a balance in the signal as it passes through the network, preventing the signal from vanishing or exploding.
- **Improving Convergence:** A well-chosen initialization can speed up the convergence of the optimization algorithm, allowing the network to learn more efficiently.

Popular initialization methods, such as Xavier/Glorot initialization and He initialization, are designed to provide a good compromise between avoiding vanishing gradients and preventing exploding gradients. The choice of initialization method can significantly impact the training performance and the ability of the neural network to generalize well to unseen data.

Initialization in KERAS



```
[6] import tensorflow
    from tensorflow import keras
    from keras import Sequential
    from keras.layers import Dense

    model = Sequential()

    model.add(Dense(10,activation='relu',input_dim=2,kernel_initializer='he_normal'))
    model.add(Dense(10,activation='relu',kernel_initializer='he_normal'))
    model.add(Dense(10,activation='relu',kernel_initializer='he_normal'))
    model.add(Dense(10,activation='relu',kernel_initializer='he_normal'))
    model.add(Dense(1,activation='sigmoid'))

    model.summary()
```

He_normal , or he_uniform or Glorat_Normal or Glorat_Uniform

In Keras the Default Weight Initializer is Glorat_Uniform