INTRODUCTION

So before diving deep into its world, let's address the first question. What is the problem that we are trying to solve?

The deep neural networks have different architectures, sometimes shallow, sometimes very deep trying to generalise on the given dataset. But, in this pursuit of trying too hard to learn different features from the dataset, they sometimes learn the **statistical noise** in the dataset. This definitely improves the model performance on the training dataset but fails massively on new data points (test dataset). This is the problem of **overfitting.** To tackle this problem we have various regularisation techniques that penalise the weights of the network but this wasn't enough.

The best way to reduce overfitting or the best way to regularise a fixed-size model is to get the average predictions from all possible settings of the parameters and aggregate the final output. But, this becomes too computationally expensive and isn't feasible for a real-time inference/prediction.

The other way is inspired by the ensemble techniques (such as AdaBoost, XGBoost, and Random Forest) where we use multiple neural networks of different architectures. But this requires multiple models to be trained and stored, which over time becomes a huge challenge as the networks grow deeper.

So, we have a great solution known as Dropout Layers.

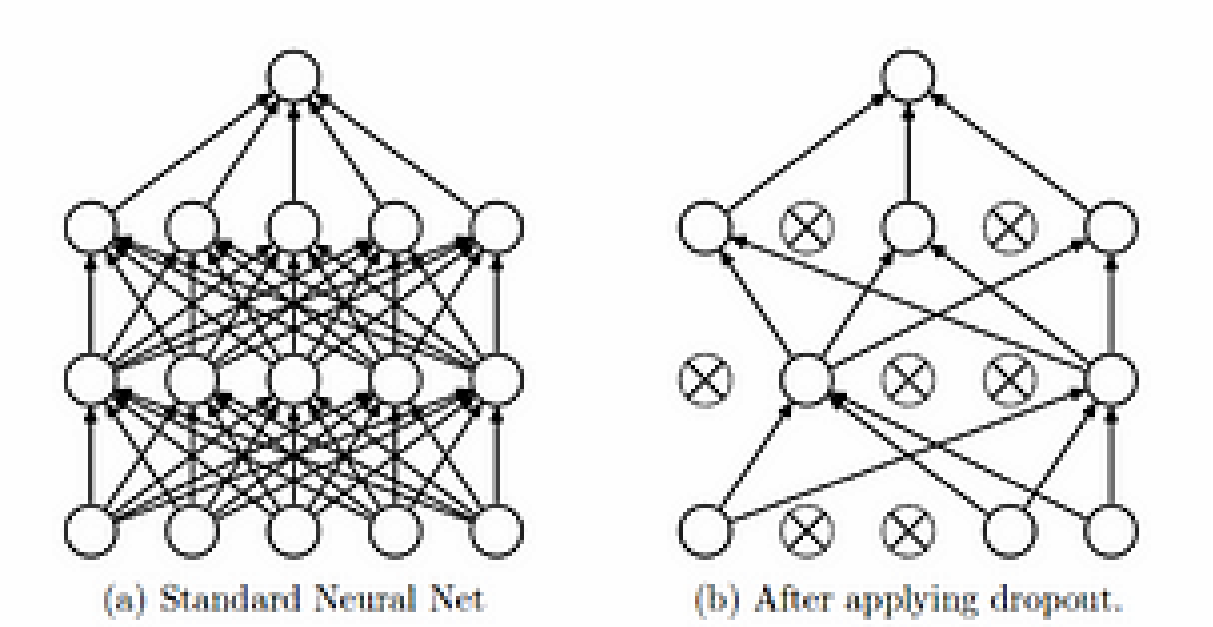

Figure 1: Dropout applied to a Standard Neural Network (Image by Nitish)

What is a Dropout?

> The term “dropout” refers to dropping out the nodes (input and hidden layer) in a neural network (as seen in Figure 1). All the forward and backwards connections with a dropped node are temporarily removed, thus creating a new network architecture out of the parent network. The nodes are dropped by a dropout probability of p.

Let’s try to understand with a given input x: {1, 2, 3, 4, 5} to the fully connected layer. We have a dropout layer with probability p = 0.2 (or keep probability = 0.8). During the forward propagation (training) from the input x, 20% of the nodes would be dropped, i.e. the x could become {1, 0, 3, 4, 5} or {1, 2, 0, 4, 5} and so on. Similarly, it applied to the hidden layers.

For instance, if the hidden layers have 1000 neurons (nodes) and a dropout is applied with drop probability = 0.5, then 500 neurons would be randomly dropped in every iteration (batch).

Generally, for the input layers, the keep probability, i.e. 1- drop probability, is closer to 1, 0.8 being the best as suggested by the authors. For the hidden layers, the greater the drop probability, the more sparse the model, where 0.5 is the most optimised keep probability, states dropping 50% of the nodes.

So how does dropout solve the problem of overfitting?

How does it solve the Overfitting problem?

In the overfitting problem, the model learns statistical noise. To be precise, the main motive of training is to decrease the loss function, given all the units (neurons). So in overfitting, a unit may change in a way that fixes up the mistakes of the other units. This leads to complex co-adaptations, which in turn leads to the overfitting problem because this complex co-adaptation fails to generalise on the unseen dataset.

Now, if we use dropout, it prevents these units to fix up the mistake of other units, thus preventing co-adaptation, as in every iteration the presence of a unit is highly unreliable. So by randomly dropping a few units (nodes), it forces the layers to take more or less responsibility for the input by taking a probabilistic approach.

This ensures that the model is getting generalised and hence reducing the overfitting problem.

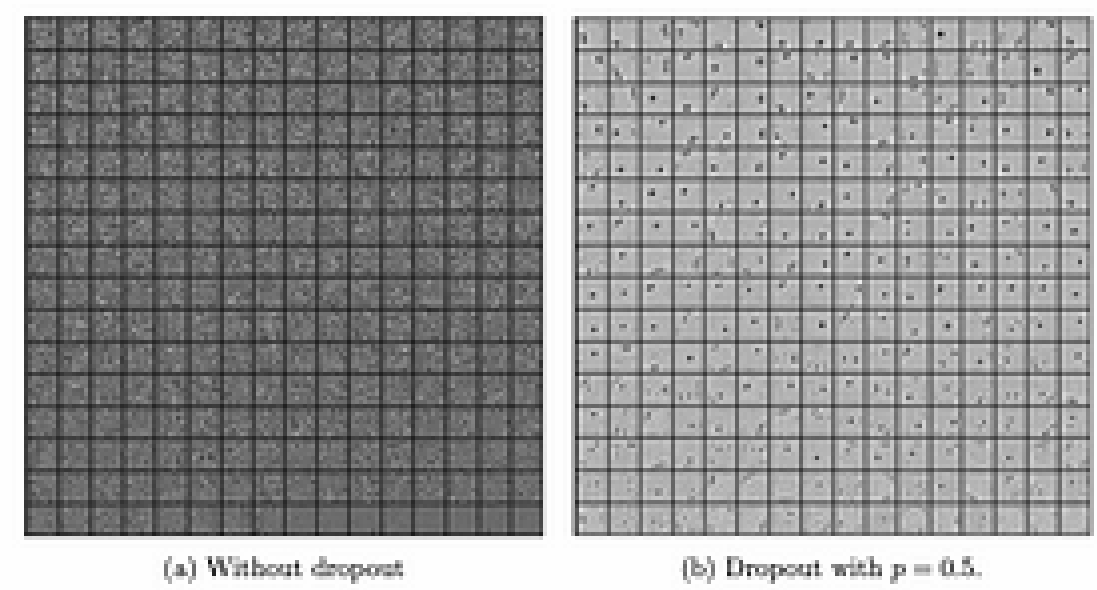

Figure 2: (a) Hidden layer features without dropout; (b) Hidden layer features with dropout (Image by Nitish)

From figure 2, we can easily make out that the hidden layer with dropout is learning more of the generalised features than the co-adaptations in the layer without dropout. It is quite apparent, that dropout breaks such inter-unit relations and focuses more on generalisation.

Dropout Implementation

Enough of the talking! Let's head to the mathematical explanation of the dropout.

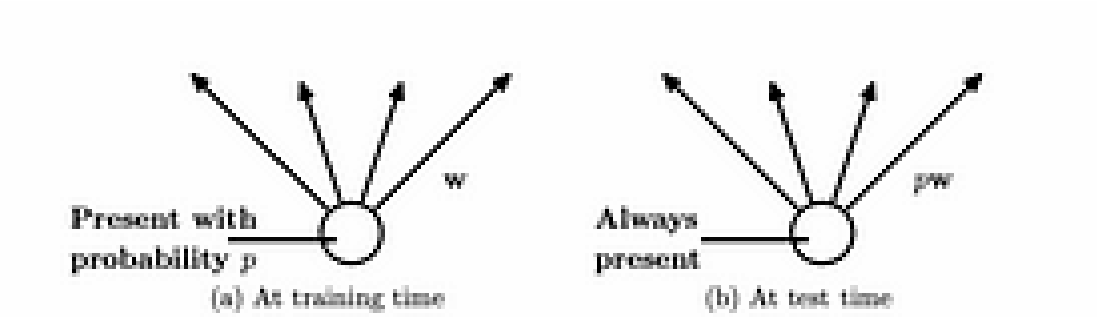

Figure 3: (a) A unit (neuron) during training is present with a probability p and is connected to the next layer with weights '**w**' ; (b) A unit during inference/prediction is always present and is connected to the next layer with weights, '**pw**' (Image by Nitish)

In the original implementation of the dropout layer, during training, a unit (node/neuron) in a layer is selected with a keep probability (1-drop probability). This creates a thinner architecture in the given training batch, and every time this architecture is different.

In the standard neural network, during the forward propagation we have the following equations:

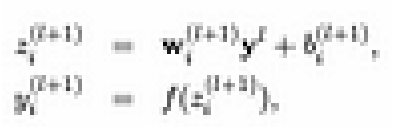

Figure 4: Forward propagation of a standard neural network (Image by Nitish)

where:
z: denote the vector of output from layer (l + 1) before activation
y: denote the vector of outputs from layer l
w: weight of the layer l
b: bias of the layer l

Further, with the activation function, z is transformed into the output for layer (l+1).

Now, if we have a dropout, the forward propagation equations change in the following way:

Figure 5: Forward propagation of a layer with dropout (Image by Nitish)

So before we calculate **z,** the input to the layer is sampled and multiplied element-wise with the independent Bernoulli variables. **r** denotes the Bernoulli random variables each of which has a probability p of being 1. Basically, **r** acts as a mask to the input variable, which ensures only a few units are kept according to the keep probability of a dropout. This ensures that we have thinned outputs "**y(bar)"**, which is given as an input to the layer during feed-forward propagation.

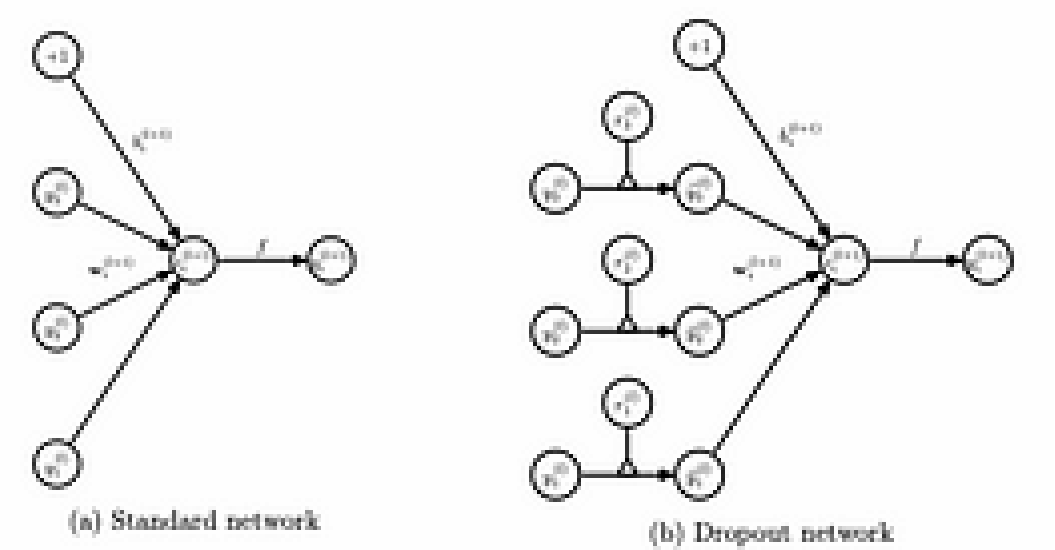

Figure 6: Comparison of the dropout network with the standard network for a given layer during forward propagation (Image by Nitish)

Dropout during Inference

Now, we know the dropout works mathematically but what happens during the inference/prediction? Do we use the network with dropout or do we remove the dropout during inference?

This is one of the most important concepts of dropout which very few data scientists are aware of.

> According to the original implementation (Figure 3b) during the inference, we do not use a dropout layer. This means that all the units are considered during the prediction step. But, because of taking all the units/neurons from a layer, the final weights will be larger than expected and to deal with this problem, weights are first scaled by the chosen dropout rate. With this, the network would be able to make accurate predictions.

To be more precise, if a unit is retained with probability p during training, the outgoing weights of that unit are multiplied by p during the prediction stage.

How Dropout was conceived

According to Geoffrey Hinton, one of the authors of “Dropout: A Simple Way to Prevent Neural Networks from Overfitting” there were a set of events that inspired the fundamental dropout.

1. The analogy with Google Brain is that it should be big because it learns a large ensemble of models. In neural networks, it is not a very efficient use of hardware since the same features would need to be invented separately by different models. This is when the idea of using the same subset of neurons was discovered.
2. Bank Teller: In those days, the tellers keep changing regularly and it must be because it would require cooperation between the employees to successfully defraud the bank. This implanted the idea of randomly selecting different neurons such that with every iteration there is a different set of neurons used. This would ensure that neurons are unable to learn the co-adaptations and prevent overfitting, similar to preventing the conspiracies in the bank.
3. Sexual Reproduction: It involves talking half of the genes of one parent and half of the other, adding a very small amount of random mutation, to produce an offspring. This creates a mixed ability of the genes and makes them more robust. This can be linked to a dropout which is used to break co-adaptations (adds randomness just like a gene mutation).

Isn't the entire journey fascinating?

The main motive to introduce the idea of how the dropout was conceived is to motivate the readers and explore the world around them and relate it to the working principles of several other neural networks. It would definitely give birth to many such innovations.