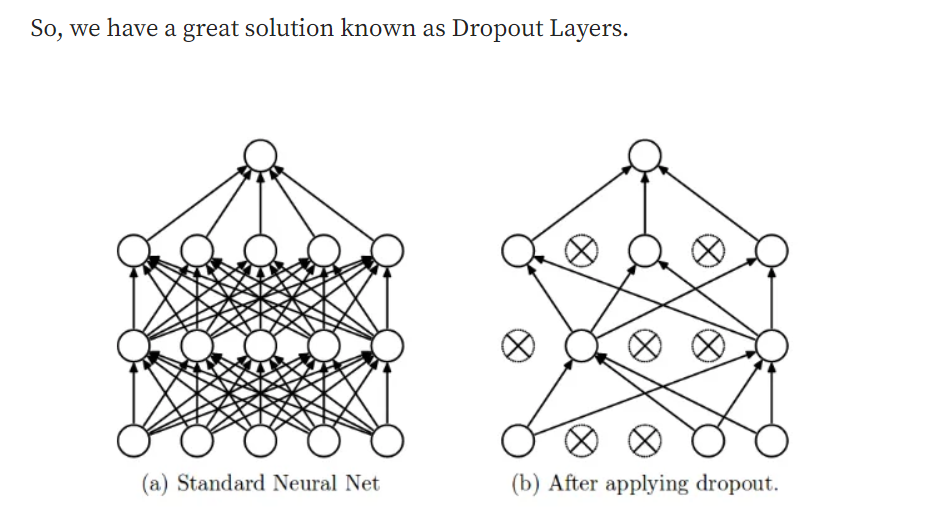
**Dropout in Neural Networks**

In machine learning, “dropout” refers to the practice of disregarding certain nodes in a layer at random during training. A dropout is a regularization approach that prevents overfitting by ensuring that no units are codependent with one another.

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 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 more sparse the model, where 0.5 is the most optimized keep probability, that states dropping 50% of the nodes.