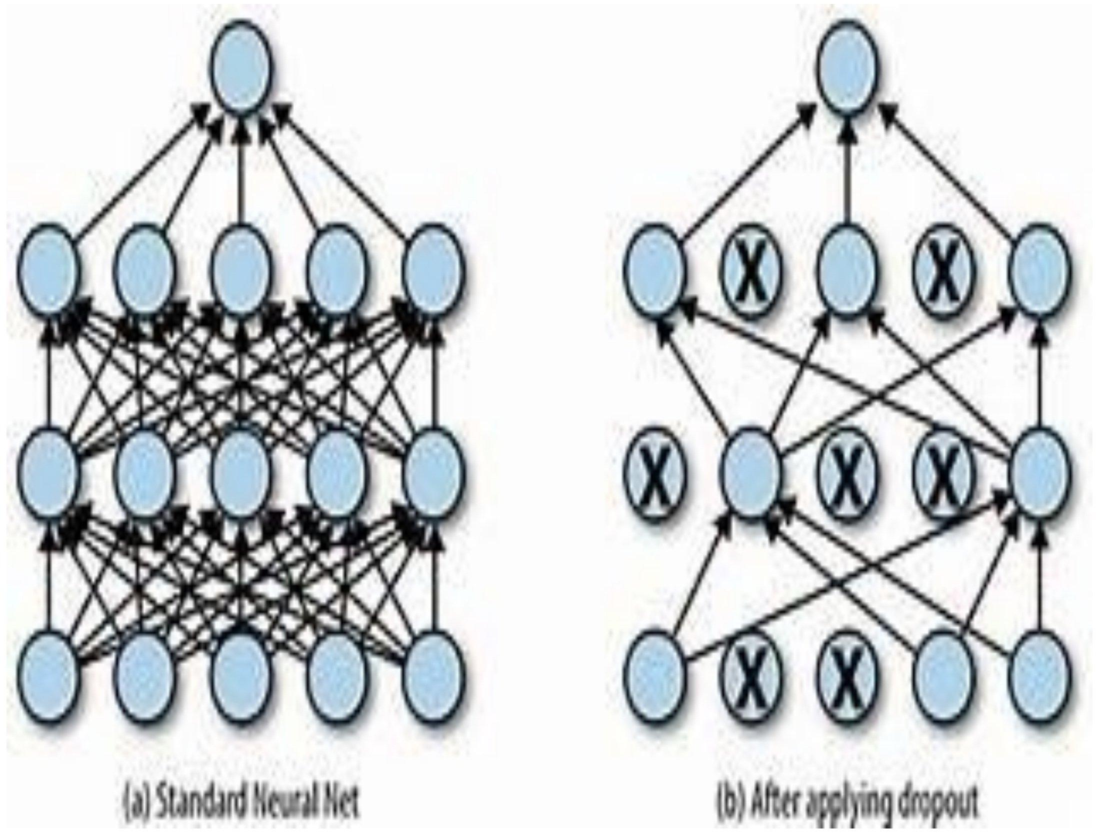
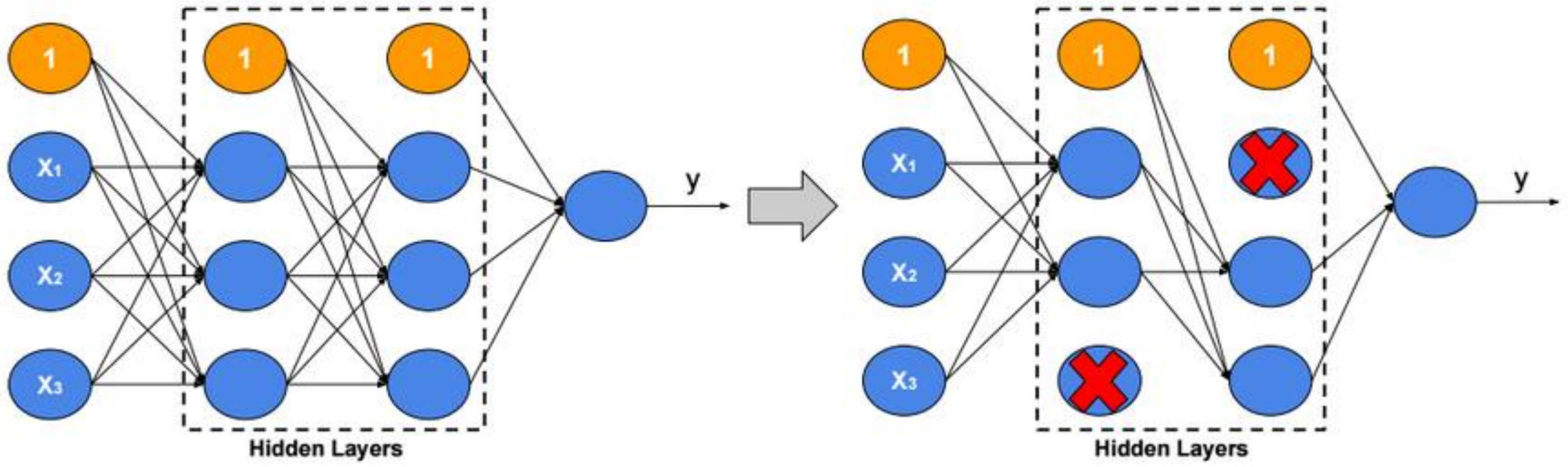
DROPOUT LAYER

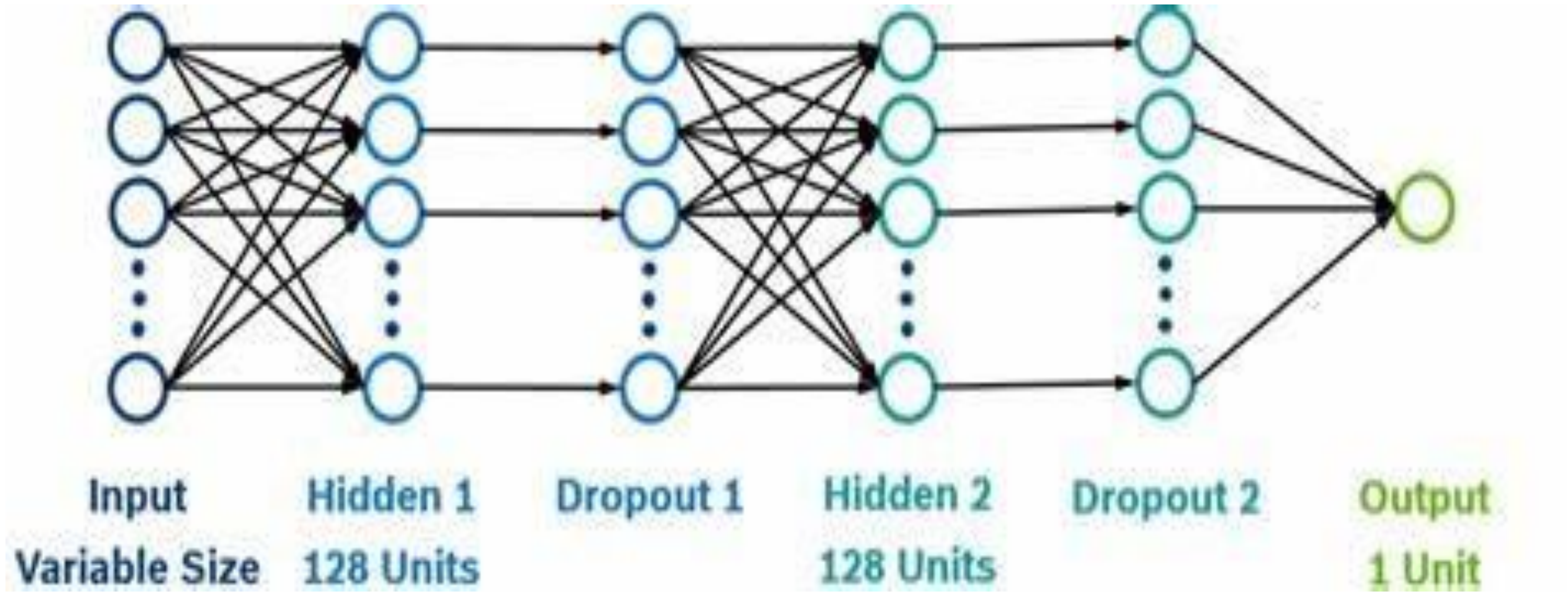
- Dropout Layer is one of the most popular regularization techniques to reduce overfitting in the deep learning models.
- Overfitting in the model occurs when it shows more accuracy on the training data but less accuracy on the test data or unseen data.
- A first “deep” regularization technique is dropout .It consists of removing units at random during the forward pass on each sample and putting them all back during test.
- A key idea in deep learning is to engineer architectures to make them easier to train. So far, we saw that we can choose the architecture (number of layers, units, filters, filter sizes, etc.), the activation function(s), and the parameter initialization.
- We can go one step further by adding mechanisms specifically designed to facilitate the training, such as “dropout”.

- As pictured on the right it removes at random some of the units during the forward pass. The units to remove are selected at random independently for every sample. The backward pass is done consistently, i.e. through the kept units alone.



- There are many ways to regularize a machine learning algorithm, but for deep learning, one of the most common is by using dropout layers. This idea was introduced by Geoffrey Hinton in 2012.
- Dropout layers are very simple. During training, each dropout layer chooses a random set of units from the preceding layer and sets their output to zero.

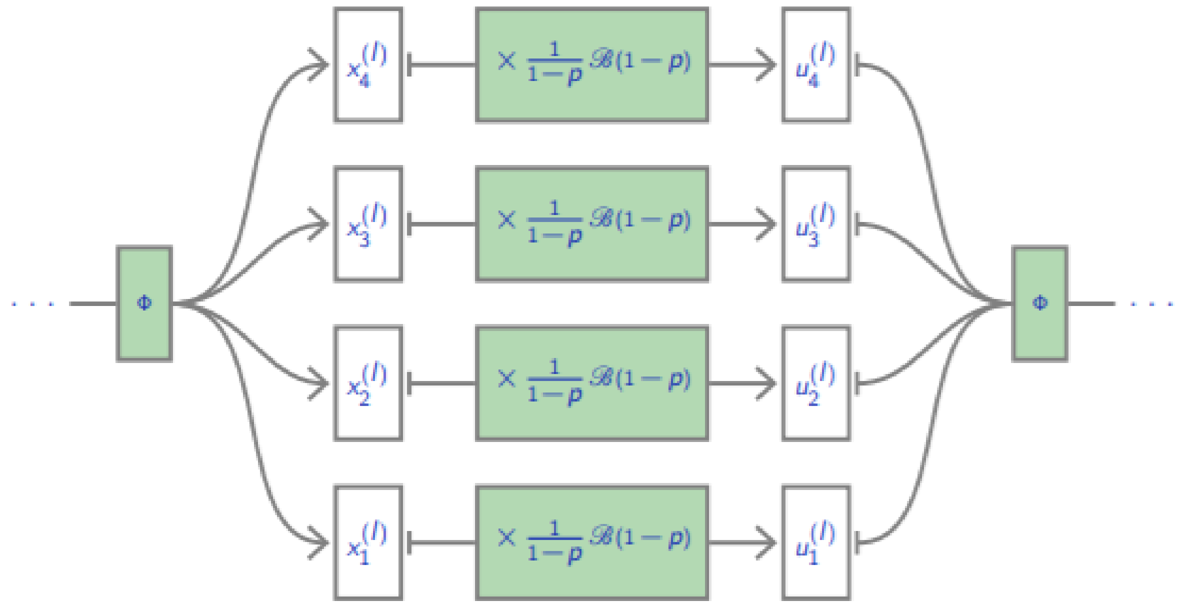



Input
Variable Size
Hidden 1
128 Units
Dropout 1
Hidden 2
128 Units
Dropout 2
Output
1 Unit

- This simple addition drastically reduces overfitting, by ensuring that the network doesn’t become overdependent on certain units or groups of units
- Just remember observations from the training set.
- If we use dropout layers, the network cannot rely too much on any one unit and therefore knowledge is more evenly spread across the whole network.
- This makes the model much better at generalizing to unseen data, because the network has been trained to produce accurate predictions even under unfamiliar conditions, such as those caused by dropping random units.
- There are no weights to learn within a dropout layer, as the units to drop are decided stochastically.
- At test time, the dropout layer doesn’t drop any units, so that the full network is used to make predictions.

- The Dropout layer in Keras implements this functionality, with the rate parameter specifying the proportion of units to drop from the preceding layer: Dropout(rate = 0.25)
- Dropout layers are used most after Dense layers since these are most prone to overfitting due to the higher number of weights, though you can also use them after convolutional layers.
- Batch normalization also has been shown to reduce overfitting, and therefore many modern deep learning architectures don’t use dropout at all and rely solely on batch normalization for regularization.
- Dropout is not implemented by actually switching off units, but equivalently as a module that drops activations at random on each sample.

Dropout is not implemented by actually switching off units, but equivalently as a module that drops activations at random on each sample.



Φ
$x_4^{(l)}$
$x_3^{(l)}$
$x_2^{(l)}$
$x_1^{(l)}$
$\times \frac{1}{1-p} \mathscr{B}(1-p)$
$\times \frac{1}{1-p} \mathscr{B}(1-p)$
$\times \frac{1}{1-p} \mathscr{B}(1-p)$
$\times \frac{1}{1-p} \mathscr{B}(1-p)$
$u_4^{(l)}$
$u_3^{(l)}$
$u_2^{(l)}$
$u_1^{(l)}$
Φ

- One has to decide on which units/layers to use dropout, and with what probability p units are dropped.
- During training, for each sample, as many Bernoulli variables as units are sampled independently to select units to remove.
- Let X be a unit activation, and D be an independent Boolean random variable of probability $1 - p$.
- We have $E(D\ X) = E(D)\ E(X) = (1 - p)E(X)$ To keep the means of the inputs to layers unchanged, the initial version of dropout was multiplying activations by $1 - p$ during test. The standard variant in use is the “inverted dropout”. It multiplies activations by 1 $1-p$ during train and keeps the network untouched during test.
- To keep the means of the inputs to layers unchanged, the initial version of dropout was multiplying activations by $1 - p$ during test.
- The standard variant in use is the “inverted dropout”. It multiplies activations by $1 / 1-p$ during train and keeps the network untouched during test.
- Dropout is implemented in PyTorch as nn.DropOut, which is a torch.Module.
- Default probability to drop is $p = 0.5$, but other values can be specified.