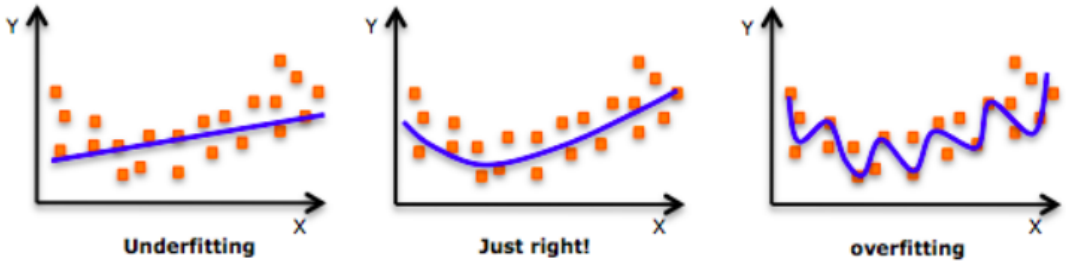
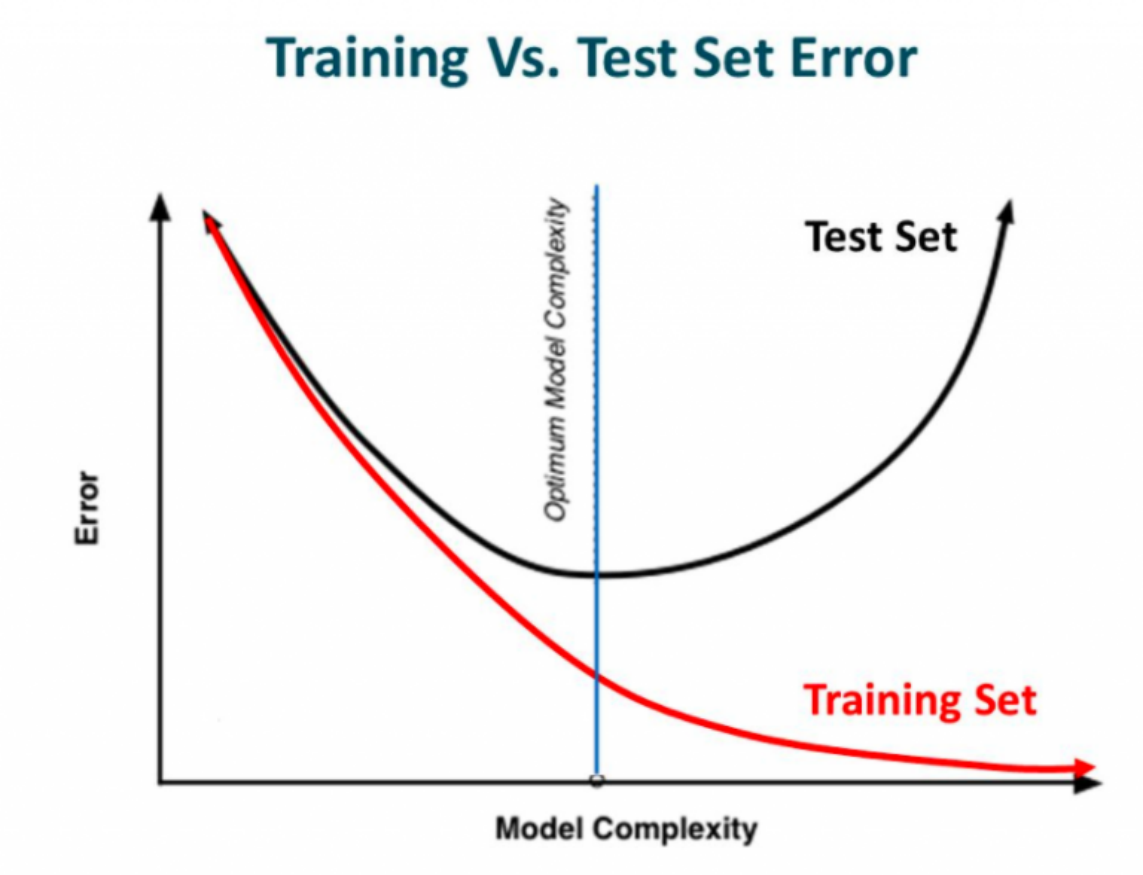
Regularization Techniques

- One of the most common problem data science professionals face is to avoid overfitting. Have you come across a situation where your model performed exceptionally well on train data, but was not able to predict test data.



- Have you seen this image before? As we move towards the right in this image, our model tries to learn too well the details and the noise from the training data, which ultimately results in poor performance on the unseen data.
- In other words, while going towards the right, the complexity of the model increases such that the training error reduces but the testing error doesn't. This is shown in the image below.



- If you've built a neural network before, you know how complex they are. This makes them more prone to overfitting. Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.

.

Different Regularization Techniques in Deep Learning

1. L1 & L2 regularization

- L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.

```
                                Cost function = Loss (say, binary cross entrop
    y) + Regularization term
```

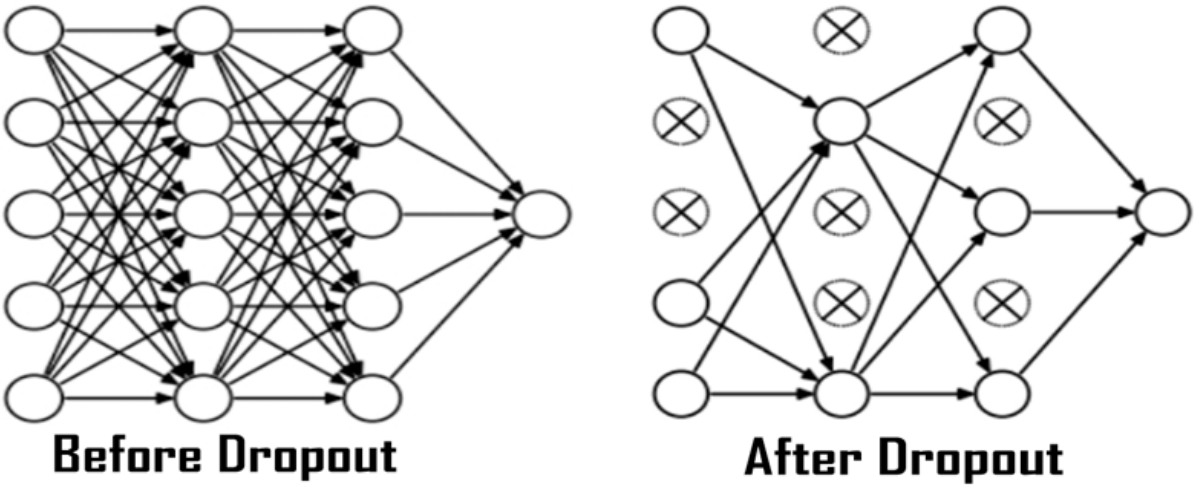
- Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

```
## Below is the sample code to apply L2 regularization to a Dense layer.

from keras import regularizers
model.add(Dense(64, input_dim=64,
                kernel_regularizer=regularizers.l2(0.01)


## Below is the sample code to apply L1 regularization to a Dense layer.

from keras import regularizers
model.add(Dense(64, input_dim=64,
                kernel_regularizer=regularizers.l1(0.01)
```

Note: Here the value 0.01 is the value of regularization parameter, i.e., lambda, which we need to optimize further. We can optimize it using the hyper parameter tuning method.

2. Dropout

- This is the one of the most interesting types of regularization techniques. It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.
- To understand dropout, let’s say our neural network structure is akin to the one shown below:



- So what does dropout do? At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below. So each iteration has a different set of nodes and this results in a different set of outputs. It can also be thought of as an ensemble technique in machine learning.

Point:-

- dropout is usually preferred when we have a large neural network structure in order to introduce more randomness.

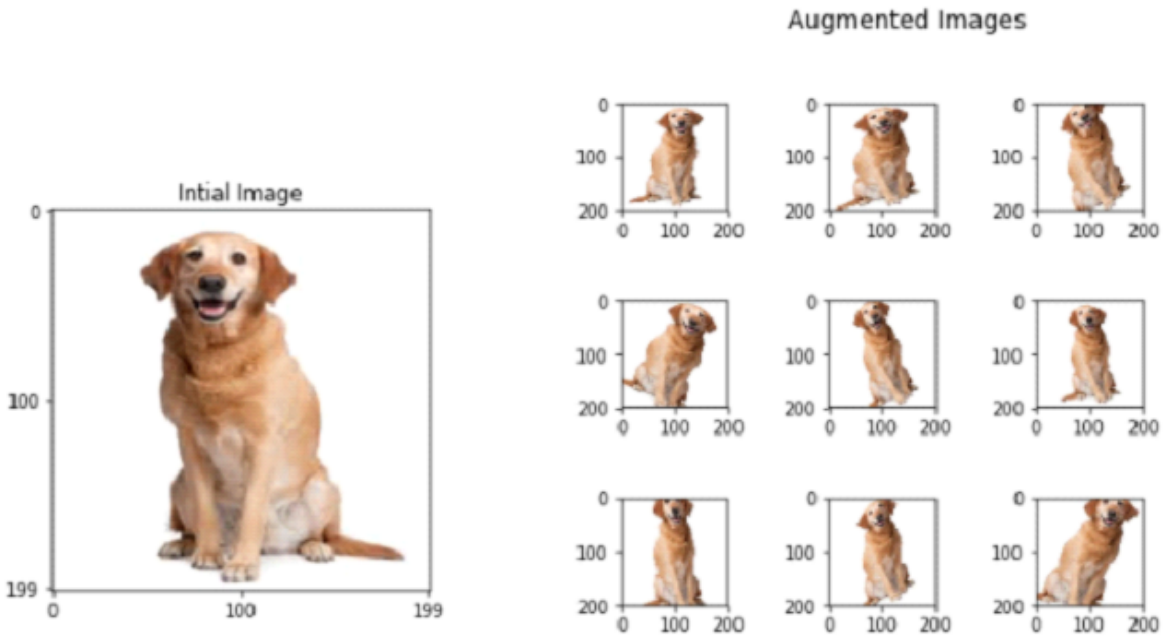
```
## In keras, we can implement dropout using the keras core layer. Below is the
python code for it:

from keras.layers.core import Dropout

model = Sequential([
 Dense(output_dim=hidden1_num_units, input_dim=input_num_units,
activation='relu'),
 Dropout(0.25),

Dense(output_dim=output_num_units, input_dim=hidden5_num_units,
activation='softmax'),
 ])
```

Note : As you can see, we have defined 0.25 as the probability of dropping. We can tune it further for better results using the Hyper parameter tuning method.

3. Data Augmentation (For Image Data)

- The simplest way to reduce overfitting is to increase the size of the training data. In machine learning, we were not able to increase the size of training data as the labeled data was too costly.
- But, now let's consider we are dealing with images. In this case, there are a few ways of increasing the size of the training data – rotating the image, flipping, scaling, shifting, etc. In the below image, some transformation has been done on the handwritten digits dataset.

- This technique is known as data augmentation. This usually provides a big leap in improving the accuracy of the model. It can be considered as a mandatory trick in order to improve our predictions.
- In keras, we can perform all of these transformations using ImageDataGenerator. It has a big list of arguments which you you can use to pre-process your training data.



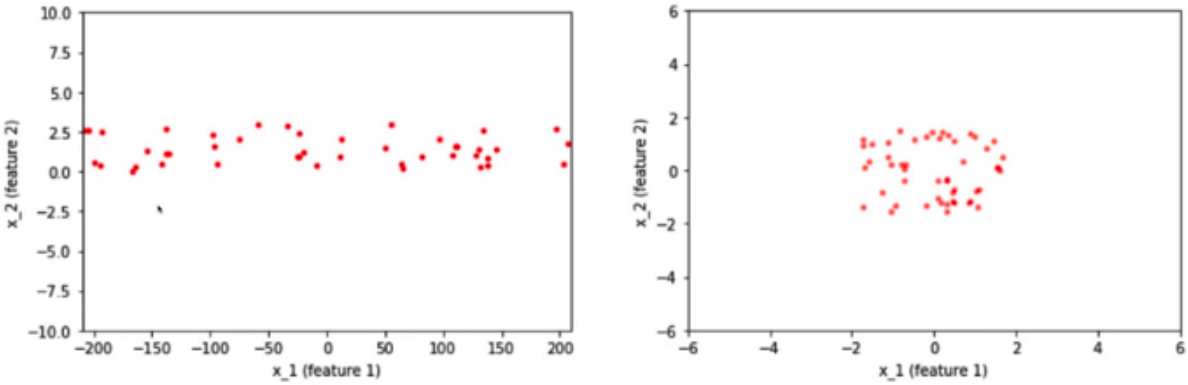

```
## Below is the sample code to implement it.
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(horizontal flip=True)
datagen.fit(train)
```

4. Batch Normalization

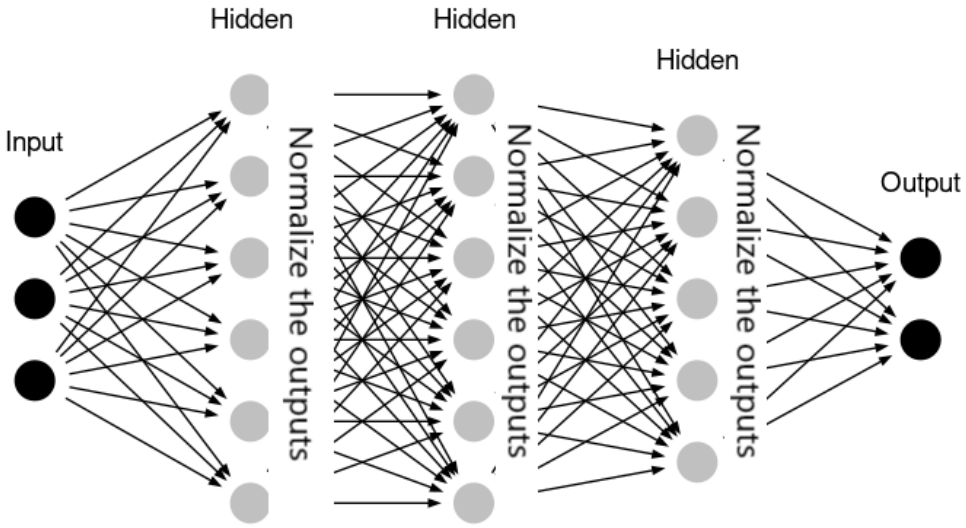
- Batch normalization is a technique for improving the speed, performance, and stability of artificial neural networks, also known as batch norm. The idea is to normalize the inputs of each layer in such a way that, they have a mean activation output zero and a unit standard deviation.
- The reason for the ‘batch’ in the term Batch Normalization is because neural networks are usually trained with a collated set of data at a time, this set or group of data is referred to as a batch. The operation within the BN technique occurs to an entire batch of input values as opposed to a single input value.

Why should we normalize the input?

- Let say we have 2D data, X1, and X2. X1 feature has a very wider spread between 200 to -200 whereas the X2 feature has a very narrow spread. The left graph shows the variance of the data which has different ranges. The right graph shows data lies between -2 to 2 and it’s normally distributed with 0 mean and unit variance.



- Essentially, scaling the inputs through normalization gives the error surface a more spherical shape, where it would otherwise be a very high curvature ellipse. Having an error surface with high curvature will mean that we take many steps that aren't necessarily in the optimal direction.
- When we scale the inputs, we reduce the curvature, which makes methods that ignore curvature like gradient descent work much better. When the error surface is circular or spherical, the gradient points right at the minimum.



Benefits of Batch Normalization

- Inclusion of Batch Normalization technique in deep neural networks improves training time
- BN enables the utilization of larger learning rates, this shortness the time of convergence when training neural networks
- Reduces the common problem of vanishing gradients
- Covariate shift within neural network is reduced

Point: In Batch normalization just as we standardize the inputs, the same way we standardize the activation at all the layers so that, at each layer we have 0 mean and unit standard deviation.

```
## In keras, we can implement BatchNormalization using the keras layer. Below
is the python code for it:
model = Sequential([
 Dense(output_dim=hidden1_num_units, input_dim=input_num_units,
activation='relu'),
 keras.layers.BatchNormalization(),

Dense(output_dim=output_num_units, input_dim=hidden5_num_units,
activation='softmax'),
 ])
```