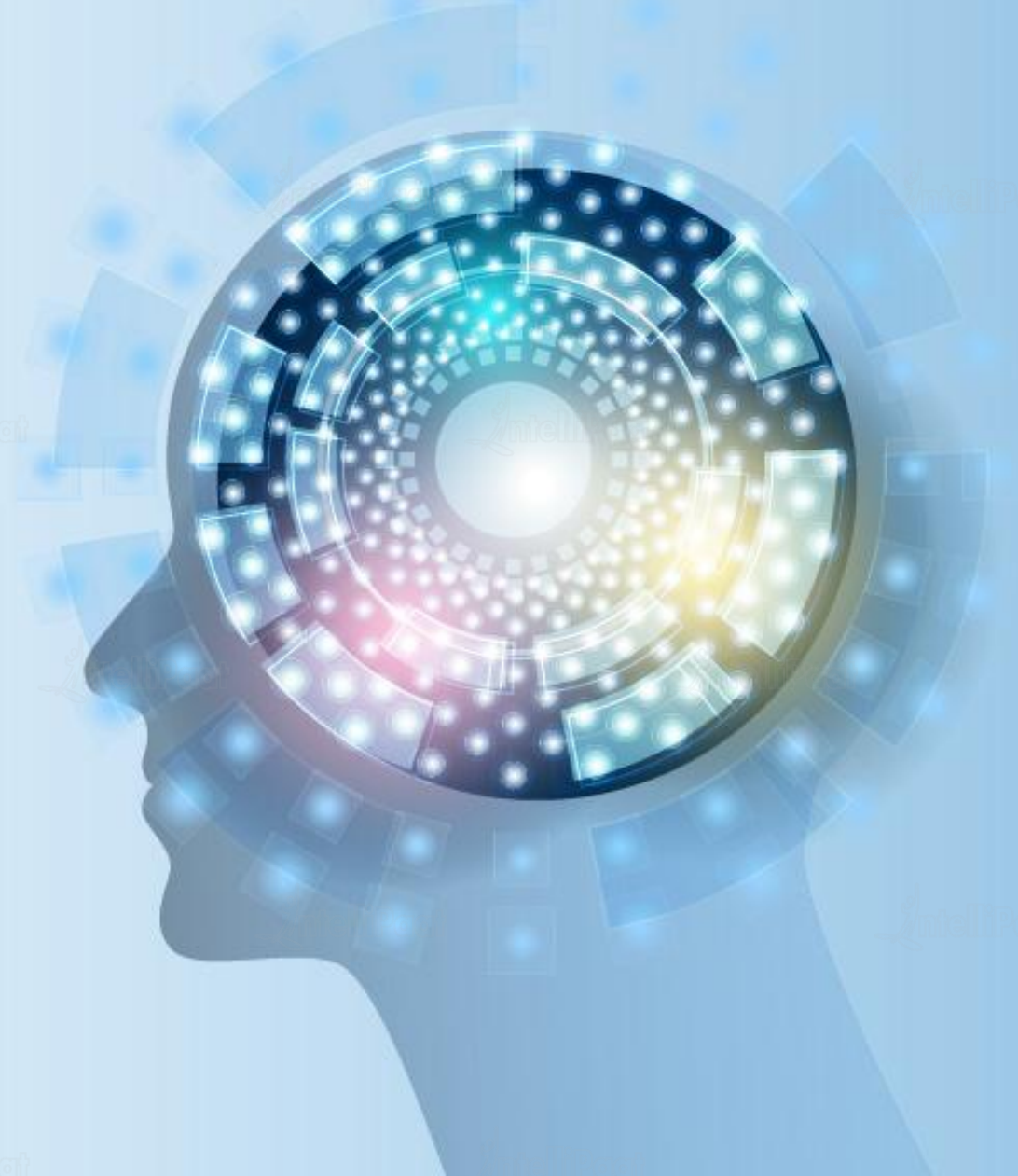




# Artificial Intelligence

Model Building Using Keras



# Agenda

**01**

Why Keras?

**02**

What Is Keras?

**03**

Models in Keras

**04**

Layers in Keras

**05**

Regularization Techniques

**06**

Batch Normalization

**07**

Keras Workflow

**08**

Use Case 1: Sequential Model

**09**

Use Case 2: Functional Model

There are countless Deep Learning frameworks available today. Why do we prefer Keras the most?



# Why Keras?

1

Keras prioritizes  
developer experience



# Why Keras?

2

Keras is broadly adopted in the industry and among the research community

# Why Keras?

3

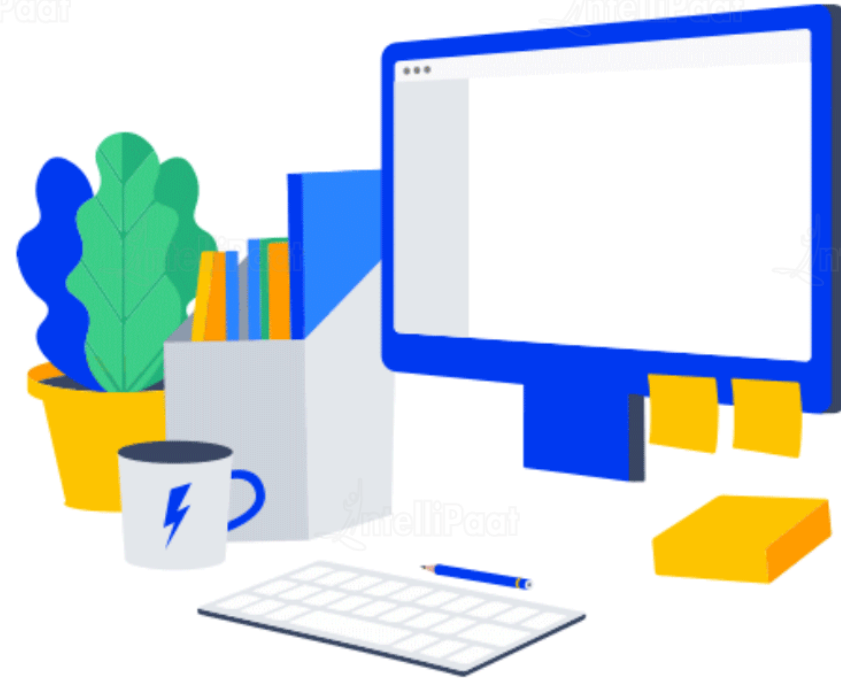
Keras makes it easy to  
turn models into  
products



# Why Keras?

# 4

Keras supports multiple backend engines and does not lock you into one ecosystem



# Why Keras?

5

Keras has strong multi-GPU support and distributed training support





# Why Keras?

6

Keras development is backed by key companies in the Deep Learning ecosystem

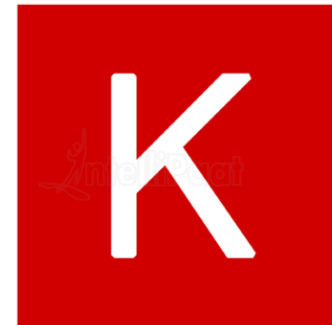
The Google logo, consisting of the word "Google" in its multi-colored, sans-serif typeface.The Amazon Web Services logo, featuring a cluster of orange cubes to the left of the text "amazon web services™" in a black, sans-serif font.The Microsoft logo, showing the four-colored square icon followed by the word "Microsoft" in a black, sans-serif font.The NVIDIA logo, with the word "NVIDIA" in a bold, black, sans-serif font, followed by a stylized, colorful swoosh graphic.

# What Is Keras?




Keras is a high-level neural networks API. It is written in Python and can run on top of Theano, TensorFlow, or CNTK. It is designed to be modular, fast, and easy to use

- It was developed by François Chollet, a Google engineer
- It was developed with the concept of:  
*‘Being able to go from idea to result with the least possible delay is key to doing good research’*
- Keras doesn't handle low-level computation. Instead, it relies on a specialized, well optimized tensor manipulation library to do so, serving as the "backend engine" of Keras
- So, Keras is the high-level API wrapper for the low-level API



# Keras

A cartoon illustration of a man with brown hair, a beard, and glasses, wearing a blue button-down shirt and tan pants. He is standing with his arms crossed, looking upwards and to the right. A thought bubble is connected to his head by three small circles.

Let us understand the different  
types of models in Keras

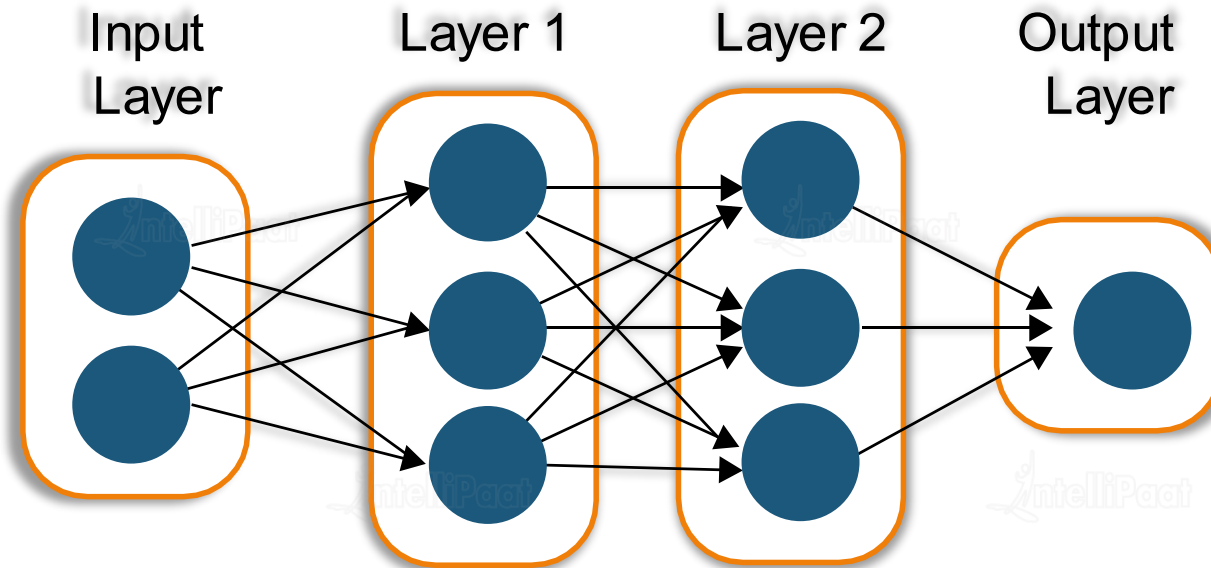
# Composing Models in Keras

You can create two types of models available in Keras, i.e., the Sequential model and the Functional model



# Sequential Models

- The Sequential model is a linear stack of layers
- You can create a Sequential model by passing a list of layer instances to the constructor
- Stacking *convolutional layers one above the other* can be an example of a sequential model



# Sequential Models



```
from keras.models import Sequential
from keras.layers import Dense, Activation
model = Sequential([
    Dense(32, input_shape=(784,)),
    Activation('relu'),
    Dense(10),
    Activation('softmax'),
])
```

*You can also simply add  
layers via the .add()  
method:*

```
model = Sequential()
model.add(Dense(32, input_dim=784))
model.add(Activation('relu'))
```

# Functional Models



The Keras functional API is used for defining complex models, such as multi-output models, directed acyclic graphs, or models with shared layers

Models are defined by creating instances of layers and connecting them directly to each other in pairs, then specifying the layers to act as the input and output to the model

Three unique aspects of the Keras Functional API are as follows:

- *Defining the Input*
- *Connecting Layers*
- *Creating the Model*

# Defining the Input



- Unlike in the Sequential model, you must create and define a standalone Input Layer that specifies the shape of the input data
- The Input Layer takes a shape argument which is a tuple that indicates the dimensionality of the input data
- In the case of one-dimensional input data, such as for a multilayer perceptron, the shape must explicitly leave room for the shape of the mini-batch size used when splitting the data while training the network
- Therefore, the shape tuple is always defined with a hanging last dimension when the input is one-dimensional

```
from keras.layers import Input  
visible = Input(shape=(2,))
```



# Connecting Layers



- Layers in the model are connected pairwise
- This is achieved by specifying where the input comes from while defining each new layer
- A bracket notation is used to specify the layer from which the input is received to the current layer, after the layer is created
- Example: We can create the Input Layer as above, and then create a Hidden Layer as a Dense Layer that receives input only from the Input Layer

```
from keras.layers import Input
from keras.layers import Dense
visible = Input(shape=(2,))
hidden = Dense(2)(visible)
```

# Creating the Model



- After creating all of your model layers and connecting them together, you must define the model

- Keras provides a model class that you can use to create a model from your created layers. It requires you to only specify the

Input and Output Layers

- Example:

```
from keras.models import Model
from keras.layers import Input
from keras.layers import Dense
visible = Input(shape=(2,))
hidden = Dense(2)(visible)
model = Model(inputs=visible, outputs=hidden)
```

# Predefined Neural Network Layers



- Keras has a number of predefined layers, such as:

1. Core Layers

2. Convolutional Layers

3. Pooling Layers

4. Locally-connected Layers

5. Recurrent Layers

6. Normalization Layers

7. Noise Layers

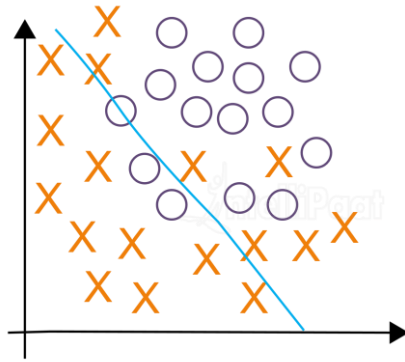
8. Embedding Layers

9. Merge Layers

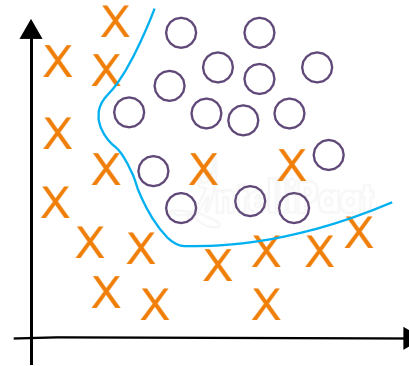
10. Advanced Activation Layers

# Performing Regularization Using Keras

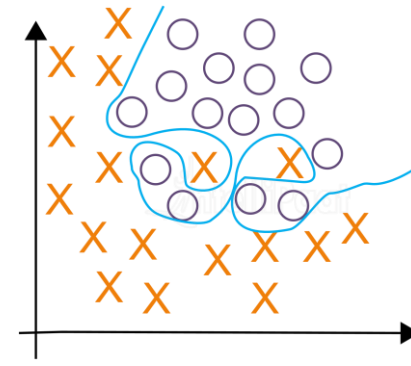
Take a look at this graph:



Underfitting



Appropriate fitting

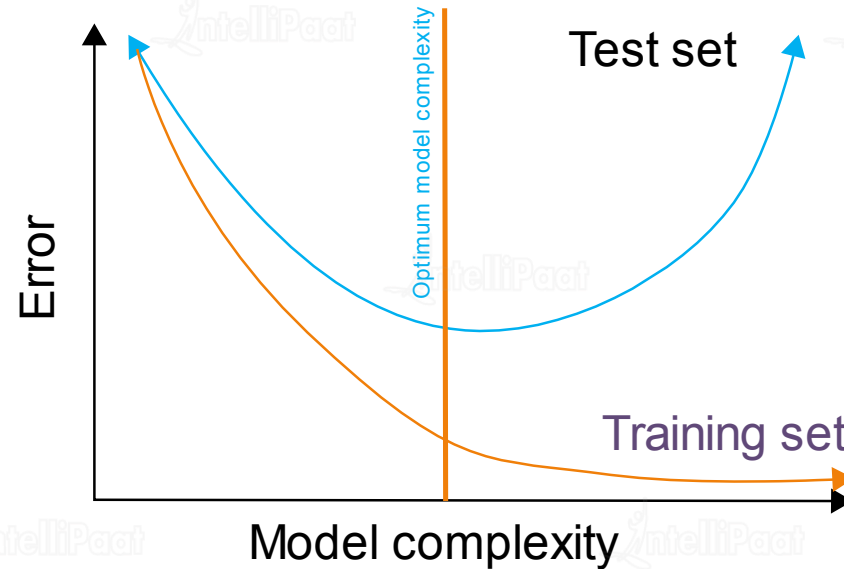


Overfitting

- As we move toward the right in this graph, our model tries to learn too well the details and the noise from the training data, which results in poor performance on the unseen data
- In other words, while going toward 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 graph on the next slide

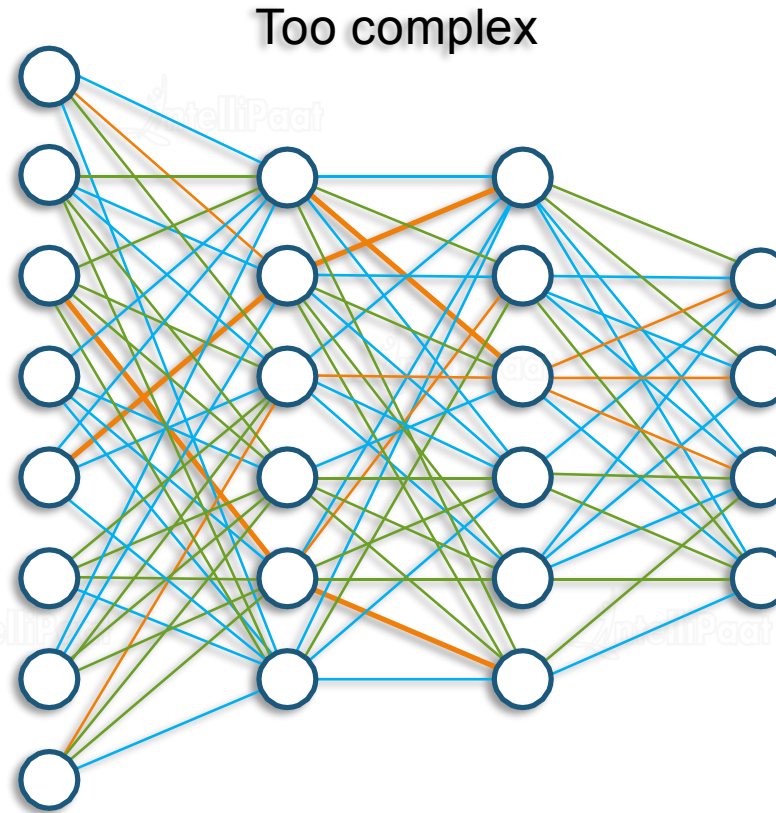
# Performing Regularization Using Keras

## Training Vs. Test Set Error



- Have you come across a situation where your model performed exceptionally well on train data but was not able to predict test data?
- Or, were you ever on the top of a competition in public leaderboard only to fall hundreds of places in the final ranking?
- Do you know how complex neural networks are and how it makes them prone to overfitting? This is one of the most common problems Data Science professionals face these days

# Performing Regularization Using Keras



*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*

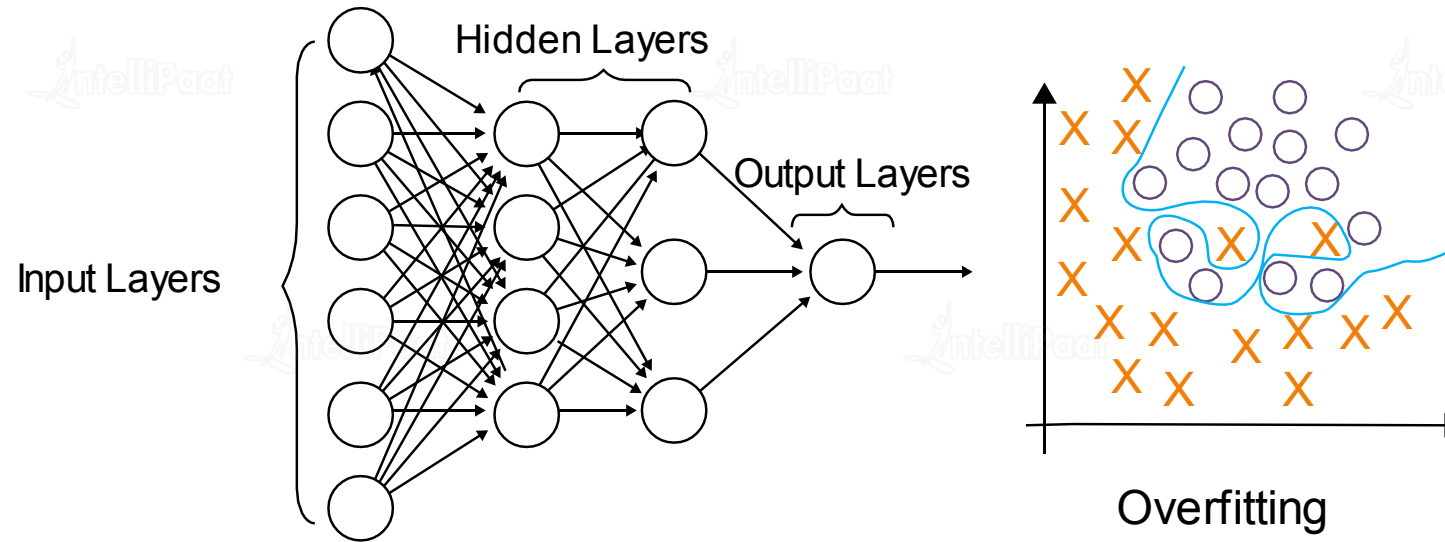
# Performing Regularization Using Keras



How does Regularization help reduce overfitting?

# Performing Regularization Using Keras

How does Regularization help reduce overfitting?

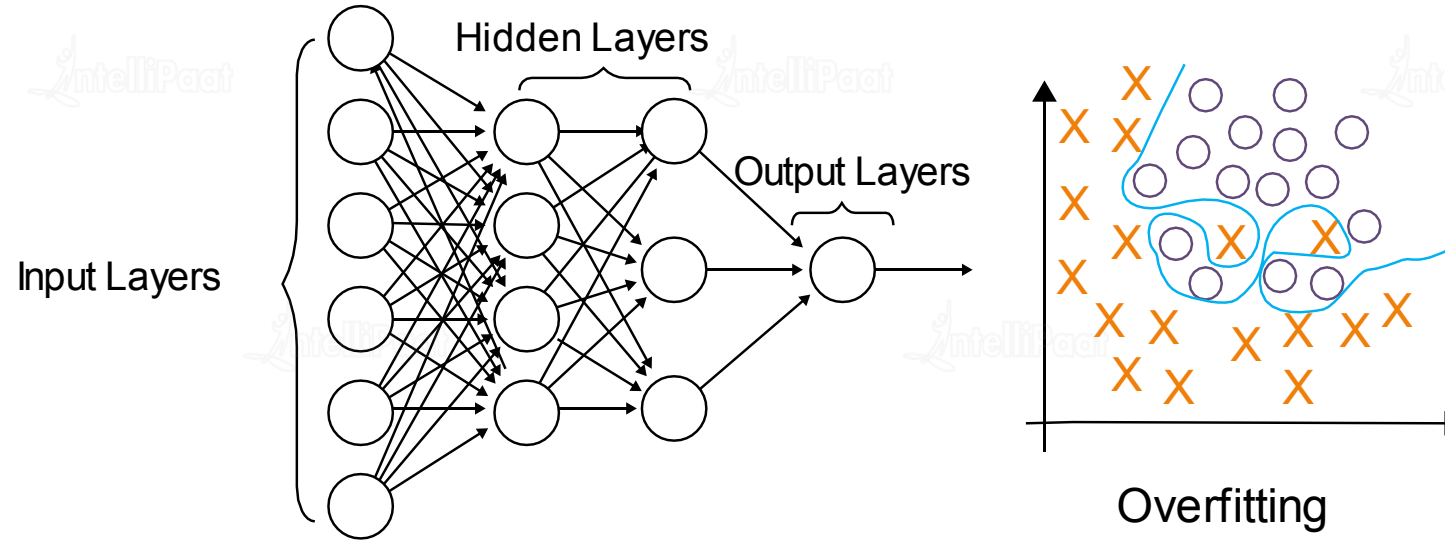


Let's consider a neural network which is overfitting on the training data as shown in the above image



# Performing Regularization Using Keras

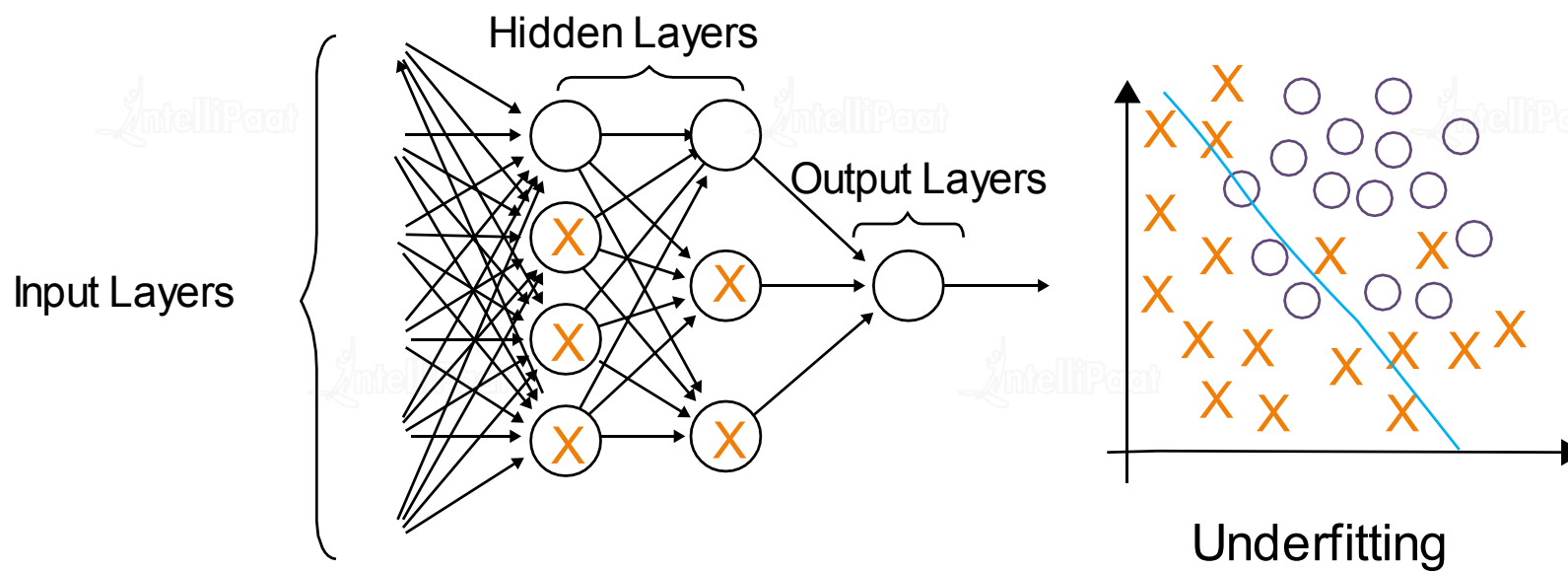
How does Regularization help reduce overfitting?



*Regularization penalizes the weight matrices of the nodes*

# Performing Regularization Using Keras

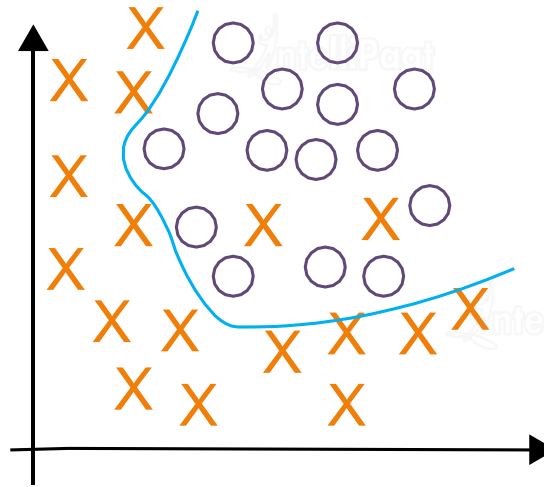
Assume that our regularization coefficient is so high that some of the weight matrices are nearly equal to zero



This will result in a much simpler linear network and slight underfitting of the training data

# Performing Regularization Using Keras

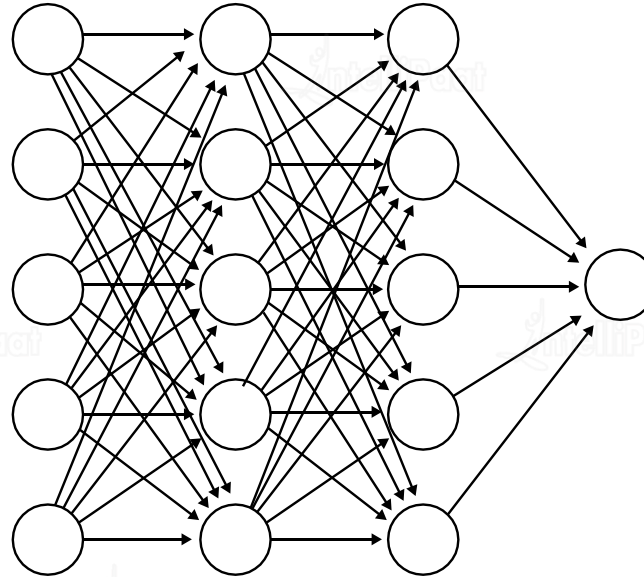
- Such a large value of the regularization coefficient is not that useful
- We need to optimize the value of the regularization coefficient in order to obtain a well-fitted model as shown in the image below



Appropriate fitting

# Performing Regularization Using Keras

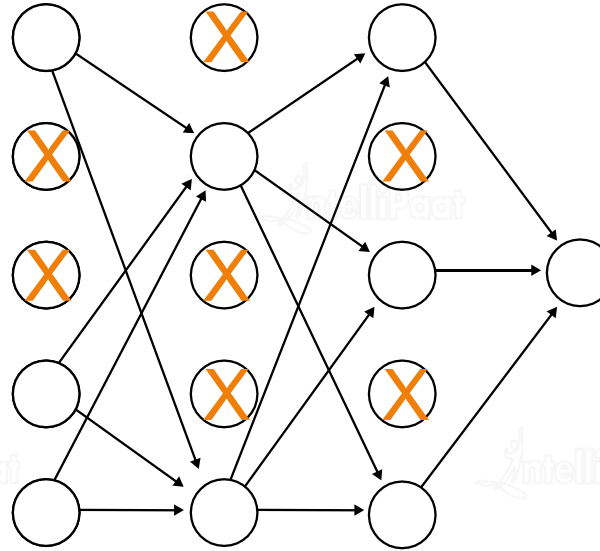
- **Dropout** produces very good results and is consequently the most frequently used Regularization technique in the field of Deep Learning
- Let's say our neural network structure is akin to the one shown below:



*So, what does dropout do?*

# Dropout

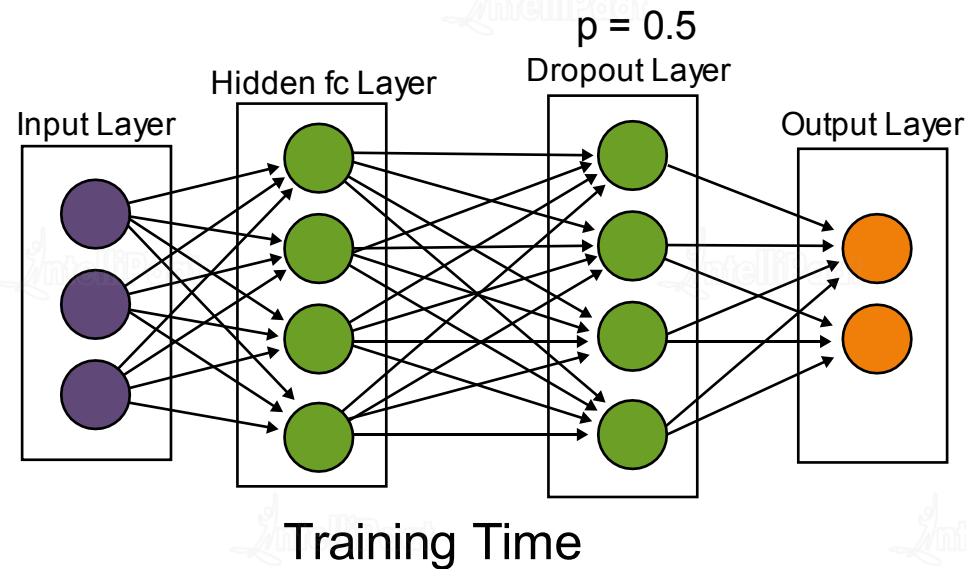
- At every iteration, it randomly selects some nodes and removes them, along with all of their incoming and outgoing connections as shown below:



- 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

# Dropout

- Ensemble models usually perform better than a single model as they capture more randomness. Similarly, dropout also performs better than a normal neural network model
- The probability of choosing how many nodes should be dropped out is the hyperparameter of the dropout function. As seen in the image below, dropout can be applied to both the Hidden Layers as well as the Input Layers



- Due to these reasons, *dropout is usually preferred when we have a large neural network structure in order to introduce more randomness*

# Dropout

*In Keras, we can implement dropout using the Keras core layer*

```
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'),
])
```

# Data Augmentation



- The simplest way to reduce overfitting is to increase the size of the training data
- Let's consider, we are dealing with images
- 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 to improve our predictions



# Data Augmentation



- In Keras, we can perform all these transformations using *ImageDataGenerator*
- It has a big list of arguments which you can use to pre-process your training data
- Example:

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(horizontal_flip=True)
datagen.fit(train)
```

# Batch Normalization



- When the data is fed through a deep neural network and weights and parameters adjust those values, sometimes making the data too big or too small, it becomes a problem. By normalizing the data in each mini-batch, this problem is largely avoided
- *Batch Normalization* normalizes each batch by both mean and variance reference
- It is just another layer, you can use to create your desired network architecture
- It is generally used *between the linear and the non-linear layers* in your network, because it normalizes the input to your activation function so that you're centered in the linear section of the activation function (such as Sigmoid)

# Batch Normalization



- When the data is fed through a deep neural network and weights and parameters adjust those values, sometimes making the data too big or too small, it becomes a problem. By normalizing the data in each mini-batch, this problem is largely avoided
- Batch Normalization normalizes each batch by both mean and variance reference
- It is just another layer, you can use to create your desired network architecture
- It is generally used *between the linear and the non-linear layers* in your network, because it normalizes the input to your activation function so that you're centered in the linear section of the activation function (such as Sigmoid)

*A normal **Dense** fully connected layer looks like this:*

```
model.add(layers.Dense(64, activation='relu'))
```

# Batch Normalization




- When the data is fed through a deep neural network and weights and parameters adjust those values, sometimes making the data too big or too small, it becomes a problem. By normalizing the data in each mini-batch, this problem is largely avoided
- Batch Normalization normalizes each batch by both mean and variance reference
- It is just another layer, you can use to create your desired network architecture
- It is generally used *between the linear and the non-linear layers* in your network, because it normalizes the input to your activation function so that you're centered in the linear section of the activation function (such as Sigmoid)

*A normal **Dense** fully connected layer looks like this:*

```
model.add(layers.Dense(64, activation='relu'))
```

*To make it Batch normalization enabled, we have to tell the Dense Layer not to use bias, since it is not needed, and thus it can save some calculation. Also, put the Activation Layer after the BatchNormalization() layer*

```
model.add(layers.Dense(64, use_bias=False))  
model.add(layers.BatchNormalization())  
model.add(Activation("relu"))
```

A cartoon illustration of a man with brown hair, a beard, and glasses, wearing a blue button-down shirt and tan pants. He is standing with his arms crossed, looking thoughtful. A thought bubble originates from his head.

Let us see how to build models in  
Keras!

# Building Models in Keras

*Let us see the 4-step workflow in developing neural networks with Keras*

**Define the Training Data**




```
In [1]: # Define the training data
import numpy as np

X_train = np.random.random((5000, 32))
y_train = np.random.random((5000, 5))
```

# Building Models in Keras

Let us see the 4-step workflow in developing neural networks with Keras

Define a Neural Network  
Model



```
In [6]: # Define the neural network model
from keras import models
from keras import layers

INPUT_DIM = X_train.shape[1]

model = models.Sequential()
model.add(layers.Dense(16, activation='relu', input_dim=INPUT_DIM))
model.add(layers.Dense(5, activation='softmax'))
```

# Building Models in Keras

*Let us see the 4-step workflow in developing neural networks with Keras*

**Configure the Learning  
Process**



```
In [7]: # Configure the Learning process
        from keras import optimizers
        from keras import metrics

        model.compile(optimizer='adam',
                      loss='categorical_crossentropy',
                      metrics=['accuracy'])
```



# Building Models in Keras

*Let us see the 4-step workflow in developing neural networks with Keras*

**Train the Model**

A large, solid black arrow pointing downwards, indicating the flow from the training step to the code implementation.

```
In [8]: # Train the model
        model.fit(X_train, y_train,
                  batch_size=128,
                  epochs=10)
```

# Building a Simple Sequential Model

# Use Case 1



Here, we will try to build a ***Sequential Network of Dense Layers***, and the dataset used is MNIST. MNIST is a classic dataset of handwritten images, released in 1999, and has served as the basis for benchmarking classification algorithms

# Building a Keras Model Using Functional Model API

## Use Case 2



Here, we will be using Keras to create a simple neural network to predict, as accurately as we can, digits from handwritten images. In particular, we will be calling the **Functional Model API of Keras** and creating a 4-layered and 5-layered neural network. Also, we will be experimenting with various optimizers: the plain Vanilla Stochastic Gradient Descent optimizer and the Adam's optimizer. We will also introduce dropout, a form of regularization technique, in our neural networks to prevent overfitting

# Quiz

# Quiz 1

Keras uses TensorFlow in the backend?

A

True

B

False

## Answer 1

Keras uses TensorFlow in the backend?

A

True

B

False



## Quiz 2

Models in Keras are of type..

**A** Sequential and Functional API

**B** Linear and Functional API

**C** Sequential and Batch

**D** None of these

## Answer 2

Models in Keras are of type..

**A** Sequential and Functional API

**B** Linear and Functional API

**C** Sequential and Batch

**D** None of these

## Quiz 3

Dropout is the other name for Data Augmentation?

A

Yes

B

No

## Answer 3

Dropout is the other name for Data Augmentation?

A

Yes

B

No

*Thank you!*



**India: +91-7847955955**

**US: 1-800-216-8930 (TOLL FREE)**



**[sales@intellipaate.com](mailto:sales@intellipaate.com)**



**24/7 Chat with Our Course Advisor**