greatlearning
Power Ahead

Deep Learning

SUNIL KUMAR VUPPALA

SUNIL.VUPPALA@GMAIL.COM

WWW.LINKEDIN.COM/IN/SUNILVUPPALA/

Agenda

Session-1:

- DL - What, Where, Why, How?
- Why deep learning now?
- Applications of DL
- Machine learning vs Deep learning
- Fundamentals of Artificial neural network
- Tensorflow playground

Session-2:

- Feed forward networks
- Various layers in DL
- Activation Functions
- Hyper parameters in DL
- Building DL models using Keras+Tensorflow
- Convolutional Neural Network

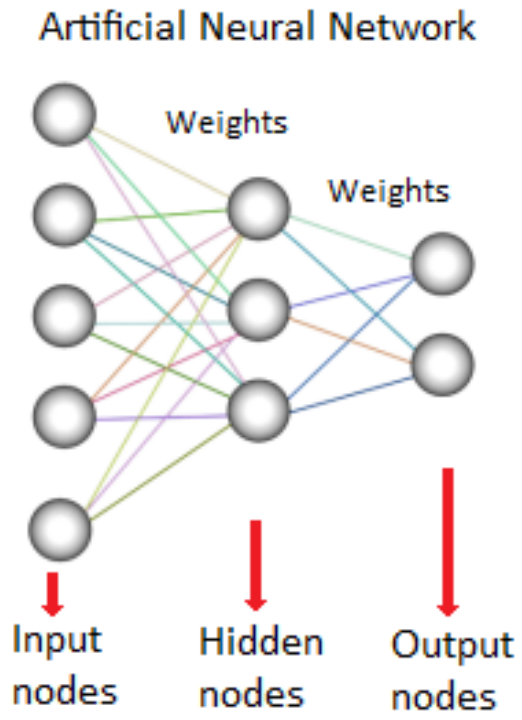
What is Deep learning (DL)?

Deep learning (DL) is a class of machine learning (ML) algorithms that*:

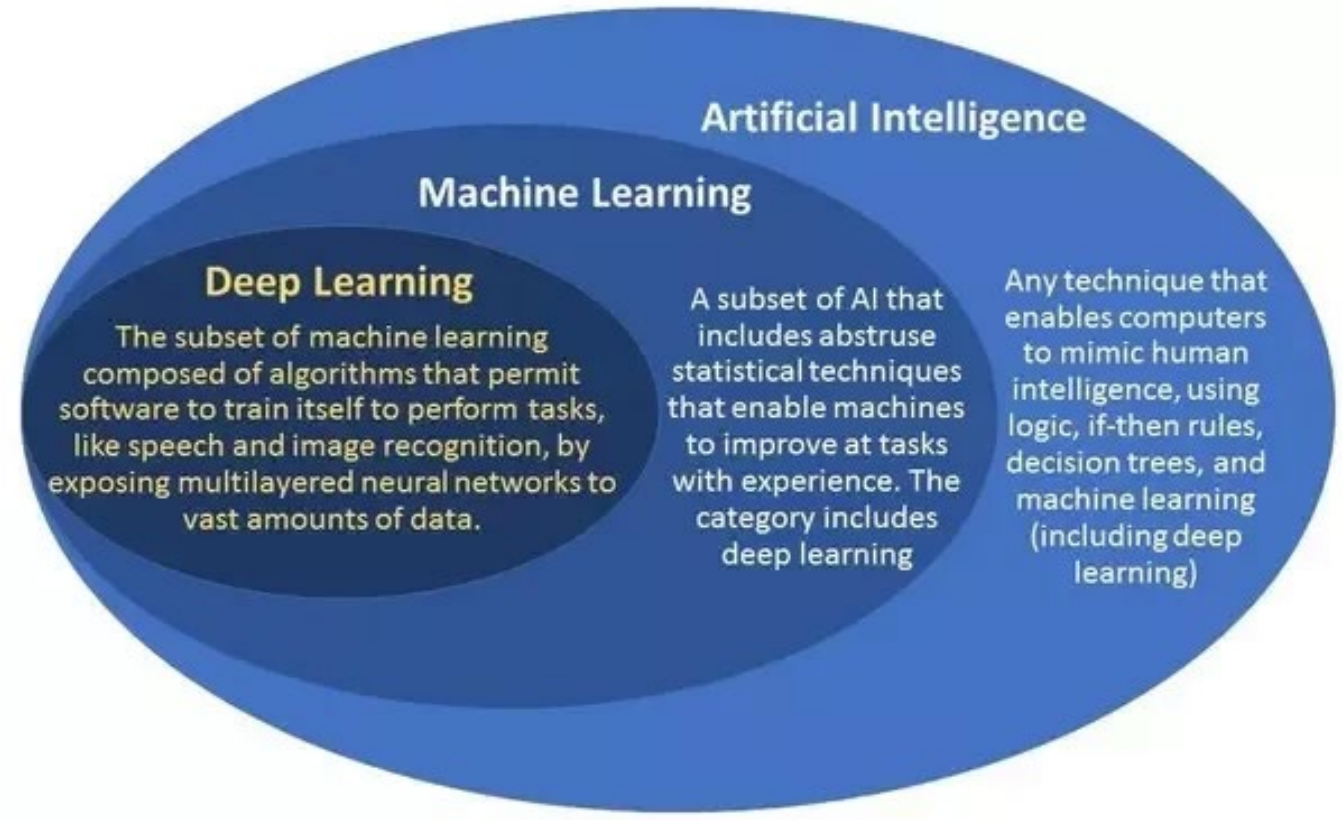
- use a cascade of many layers of **nonlinear processing** units for **feature extraction** and **transformation**
- Each **successive layer** uses the output from the previous layer as input
- learn multiple levels of **representations** that correspond to **different levels of abstraction**

DL is inspired by the structure and function of the **brain** called **artificial neural networks**.

*wikipedia



Where it fits in?



Where to use DL?

- Finance domain (**Categorical and Numerical** data) :
  - Identify the fraud detection in credit card transactions
- Healthcare domain (**Image** data):
  - Lung cancer classification of images
- Social media(**Image** data):
  - Face recognition and tag the people
- Across the domains (**Text**):
  - Identify the potential cases of automation from historical ticket data using
  - Build a chat bot

Few more applications of DL are: Personalized recommendations, Prediction, Anomaly detection, Drug discovery, Autonomous cars, Video analytics etc…

- ***But is it NEW concept?***

Brief history

1958 Perceptron

1974 Backpropagation

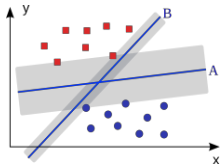
Convolution Neural Networks for Handwritten Recognition
1998

Google Brain Project on 16k Cores
2012

awkward silence (AI Winter)

1969
Perceptron criticized

1995
SVM reigns



2006
Restricted Boltzmann Machine

2012
AlexNet wins ImageNet

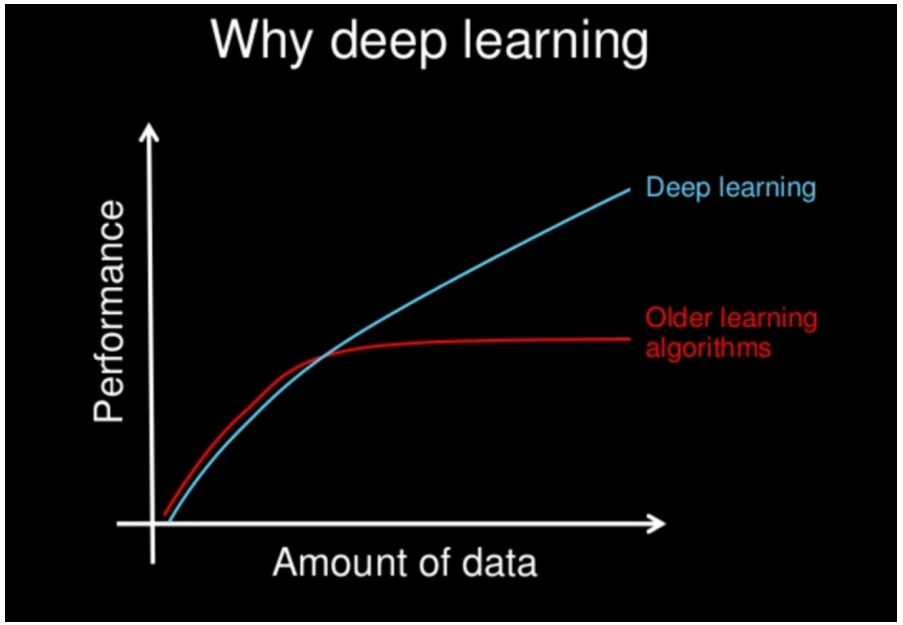
IMAGENET

Why second wave?

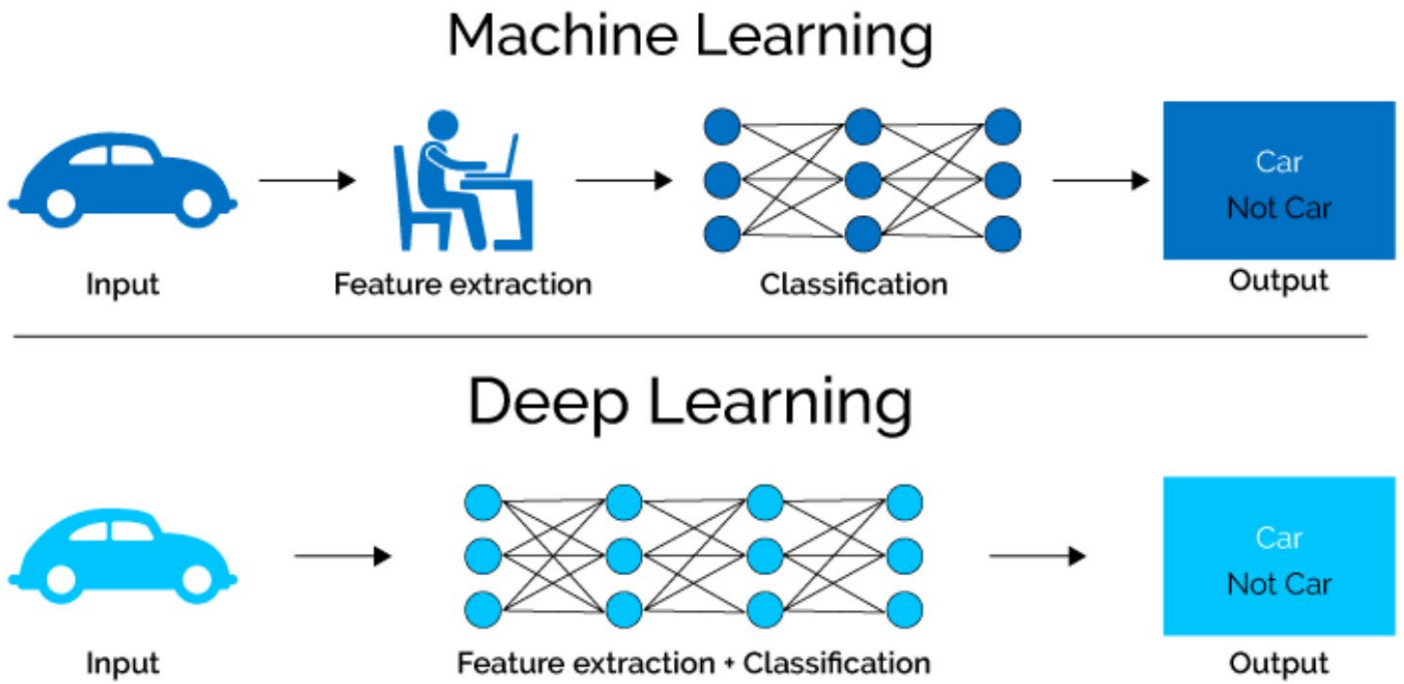
- More data from systems and sensors (IoT)
- More compute power : GPUs, multi-core CPUs

Important property:

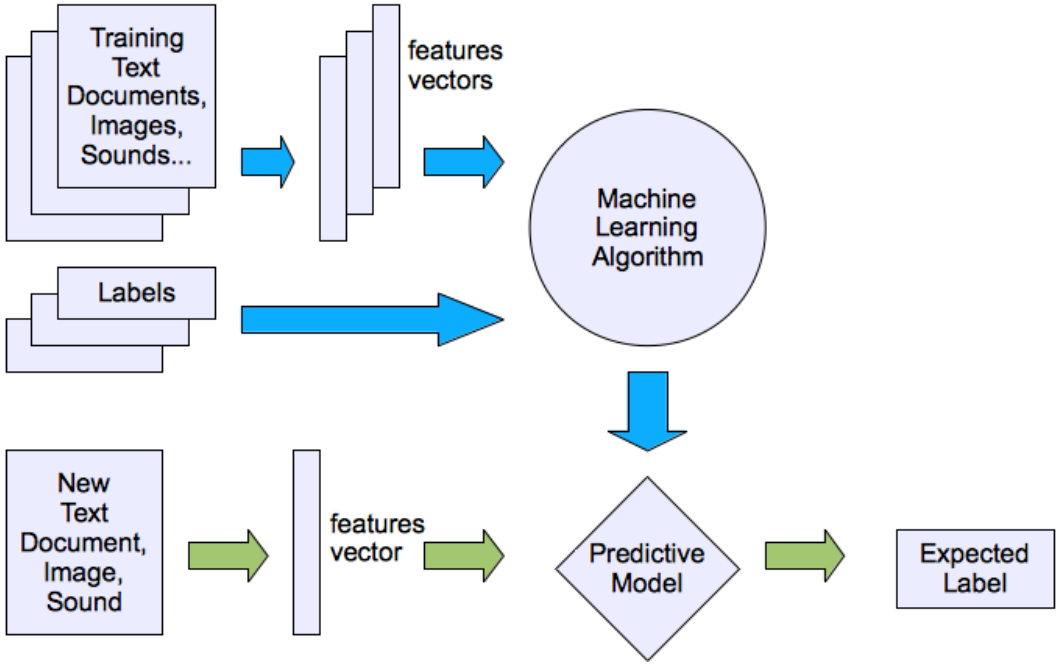
Results get better with **more data + bigger models** + more computation

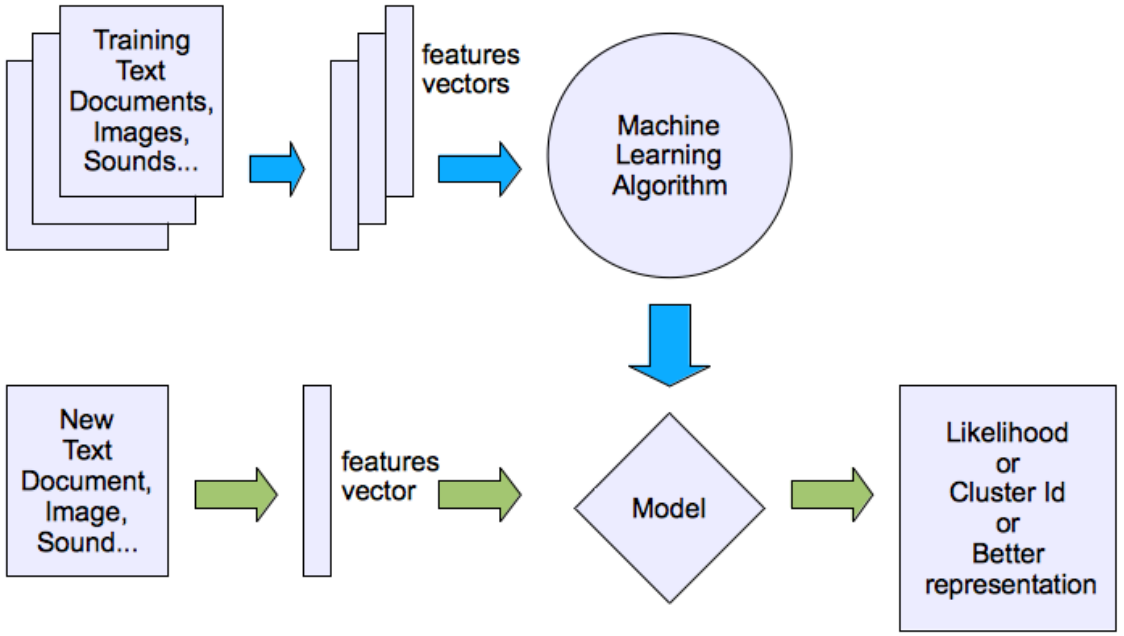


Machine Learning vs Deep Learning

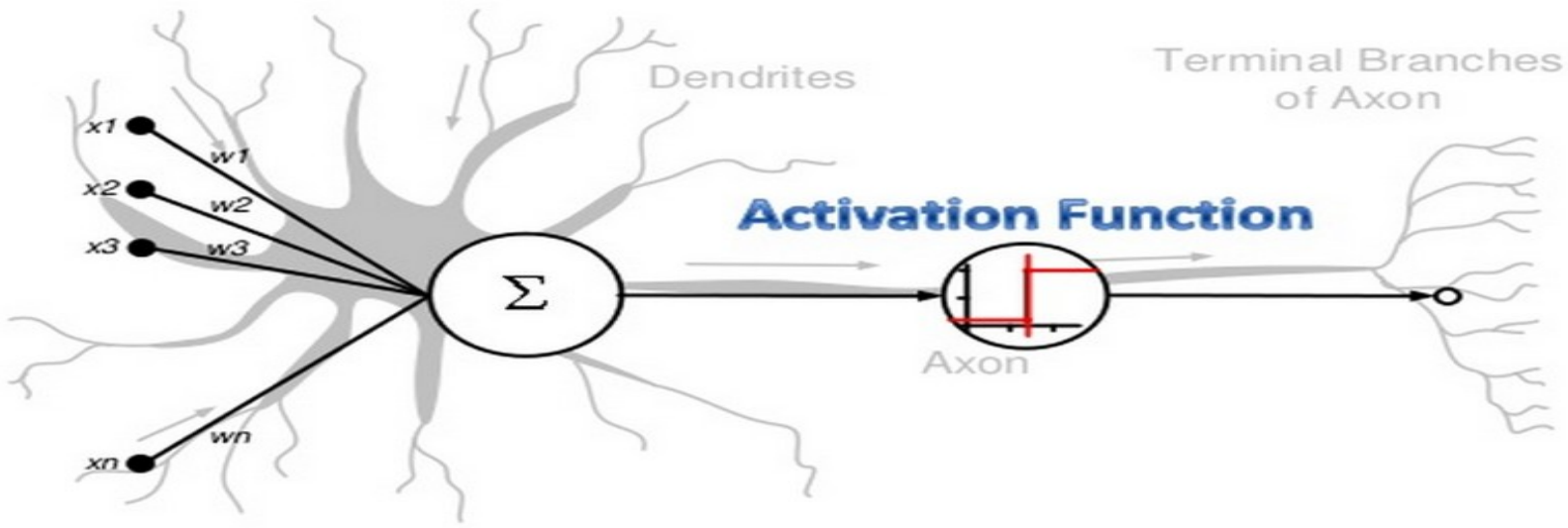


Supervised & unsupervised learning (Recap)





Artificial neural network



Slide credit : Andrew L. Nelson

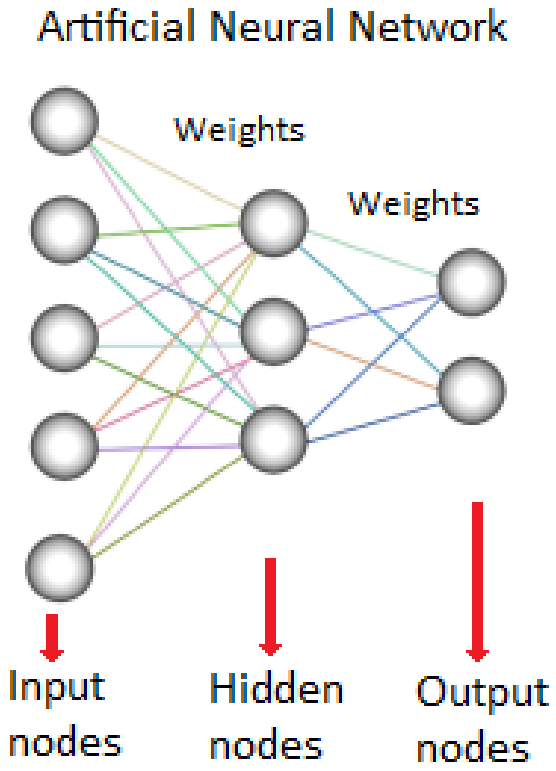
Tensorflow platground demo

A neural net model is composed a set of **Layers**.

Run multiple examples in the increasing order of complexity

- Linear
- Comlicated circle
- Spiral
- Shallow learning
- Deep learning

There are many types of layers available and each layer has many parameters. Thus we can have infinitely many different network architectures.



greatlearning
Power Ahead

Deep Learning Fundamentals

Artificial neural network

Artificial **neurons** are elementary units in an artificial neural network.

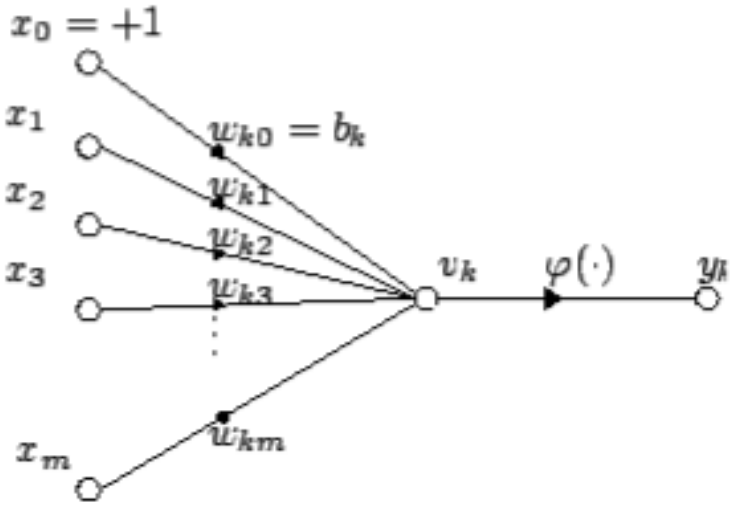
The artificial neuron receives one or more **inputs** (representing ***dendrites***) and sums them to produce an **output** (or activation) (representing a neuron's ***axon***).

Usually the sums of each node are weighted, and the sum is passed through a non-linear function known as an **activation function**.

The output of the *k*th neuron is:

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

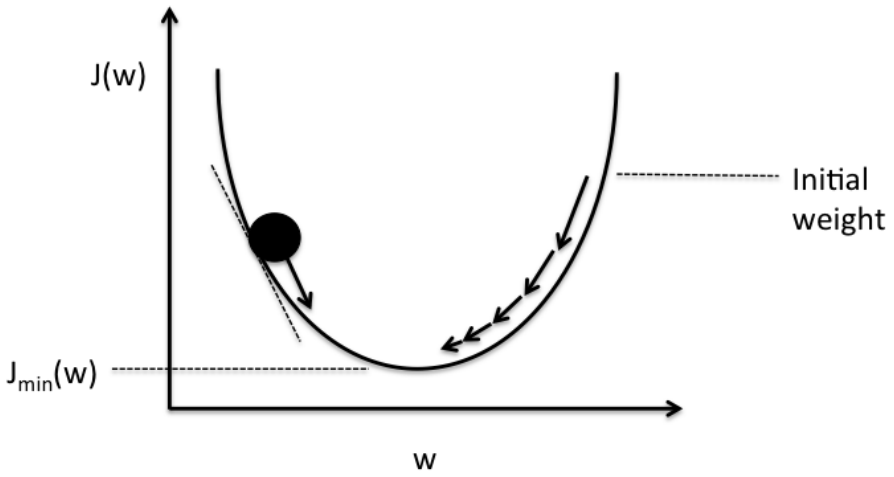
Where φ (phi) is the transfer function.



Back propagation

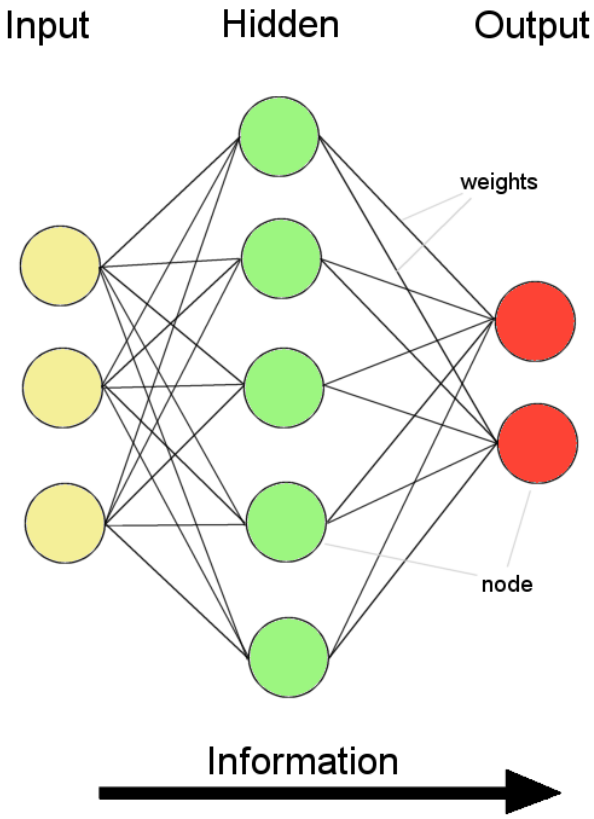
With the concept of **gradient descent**

- Forward propagation: Activation [input * weight matrix]
- To get optimal value of each weight:
  - Direction is opp. to the gradient
  - Find weight with minimum error
  - Derivative (slope of a tangent line - rate of change of a function)
  - Partial derivative (wrt one of the variables)
  - Chain rule (derivatives of composite functions)
  - calculate the error wrt each weight
- **New weight = old weight - Derivative Rate * learning rate**



Schematic of gradient descent.

Feed forward nets



Information flow is unidirectional

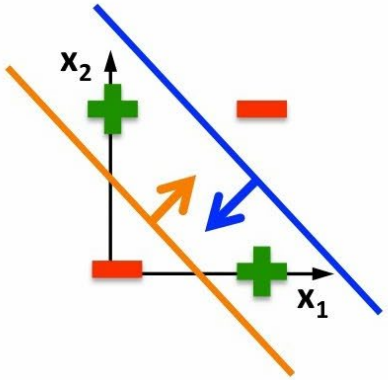
- Data is presented to *Input layer*
- Passed on to *Hidden Layer*
- Passed on to *Output layer*

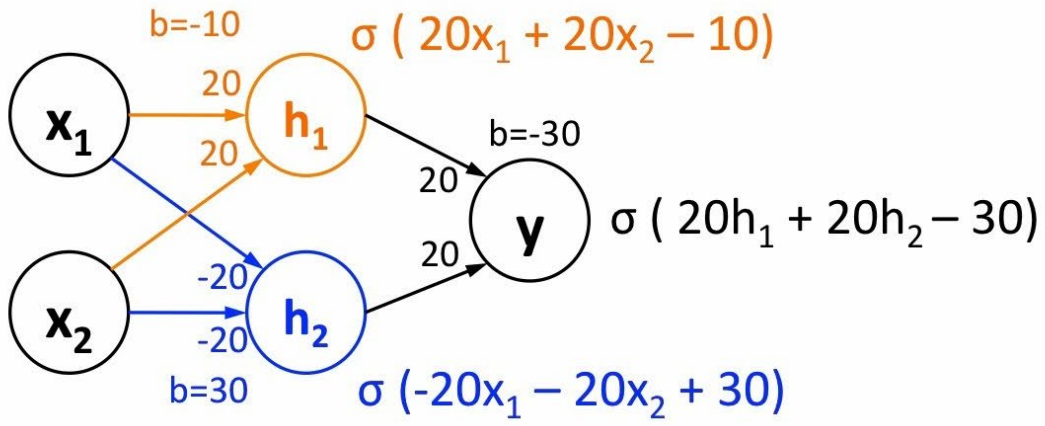
- Information is distributed
- Information processing is parallel
- Backpropagation

- Requires training set (input / output pairs)
- Starts with small random weights
- Error is used to adjust weights (supervised learning)

Solving XOR with a neural net

Linear classifiers
cannot solve this





σ(20*0 + 20*0 – 10) ≈ 0
σ(20*1 + 20*1 – 10) ≈ 1
σ(20*0 + 20*1 – 10) ≈ 1
σ(20*1 + 20*0 – 10) ≈ 1

σ (-20*0 – 20*0 + 30) ≈ 1
σ (-20*1 – 20*1 + 30) ≈ 0
σ (-20*0 – 20*1 + 30) ≈ 1
σ (-20*1 – 20*0 + 30) ≈ 1

σ (20*0 + 20*1 – 30) ≈ 0
σ (20*1 + 20*0 – 30) ≈ 0
σ (20*1 + 20*1 – 30) ≈ 1
σ (20*1 + 20*1 – 30) ≈ 1

Basic set of Layers

- Dense Layer
- Dropout Layer
- Convolution1D
- Convolution2D
- MaxPooling1D
- LSTM

Dense and Dropout layers

Dense Layer:

It creates a regular fully connected Neural net layer

Dense (***output_dim*** , ***activation='linear'***)

- **output_dim**: (integer > 0) Specifies the size of the Layer (Number of Neurons)
- **activation**: name of activation function

Dropout Layer:

Dropout*: A Simple Way to Prevent Neural Networks from Over- fitting*

Dropout (p)

Applies Dropout to the input. Dropout consists in randomly setting a fraction p of the input units to 0 at each update during the training phase, which helps prevent over-fitting.

Convolution1D/2D

Convolution operator for **filtering neighborhoods of one-dimensional inputs**.

1D convolution layer (e.g. temporal convolution)

2D convolution layer (e.g. spatial convolution over images)

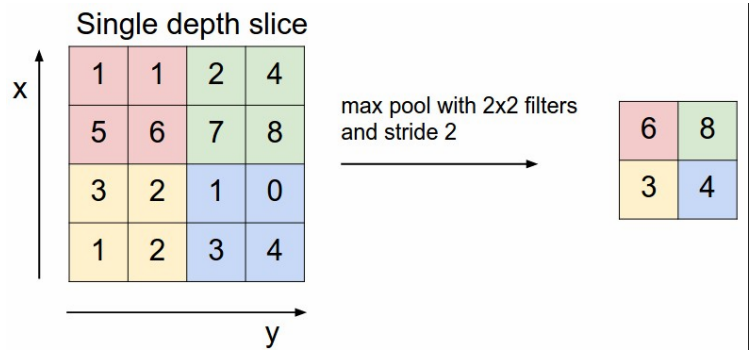
Convolution1D (nb_filter, filter_length, activation='linear', border_mode='valid', subsample_length=1)

- **nb_filter**: Number of convolution kernels to use (dimensionality of the output).
- **filter_length**: The extension (spatial or temporal) of each filter.
- **activation**: name of activation function to use
- **border_mode**: 'valid' or 'same'.
- **subsample_length**: factor by which to subsample output.

MaxPooling1D

Max pooling operation for temporal data.

The max-pooling layer would **reduce the input Matrix into a down sampled size with max value for each block**. Please refer to the image shown below for an example.



MaxPooling1D (***pool_length***=2, stride=None, border_mode='valid')

pool_length: size of the region to which max pooling is applied

stride: integer, or None. factor by which to downscale. 2 will halve the input. If None, it will default to pool_length.

border_mode: 'valid' or 'same'

Activation Functions

In Neural Networks, the activation function of a node defines the **output of that node given an input or set of inputs.**

A standard computer chip circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on input.

This is similar to the behavior of the **linear perceptron** in neural networks.

It is the nonlinear activation function that allows such networks to compute nontrivial problems using only a small number of nodes.

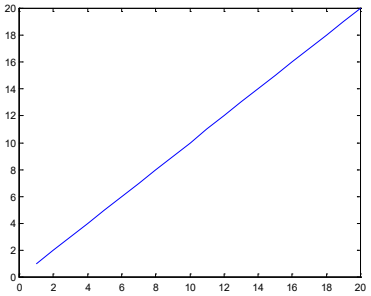
ReLu: max(0,x)

Properties of activation function

- **Nonlinear**:
  - When the activation function is non-linear, then a two-layer neural network can be proven to be a universal function approximator.
- **Continuously differentiable:**
  - This property is necessary for enabling gradient-based optimization methods.
- **Range**:
  - When the range of the activation function is finite, gradient-based training methods tend to be more stable.
  - Smaller learning rates are typically necessary.
- **Monotonic**:
  - When the activation function is monotonic, the error surface associated with a single-layer model is guaranteed to be convex.
- **Smooth**
  - Functions with a Monotonic derivative have been shown to generalize better in some cases.

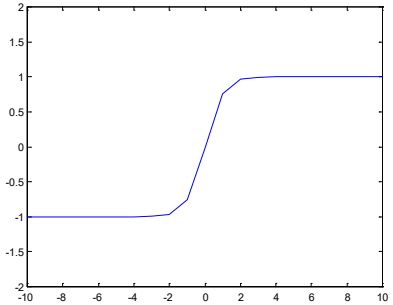
  Approximates identity near the origin:
  - The neural network will learn efficiently when its weights are initialized with small random values.
  - When the activation function does not approximate identity near the origin, special care must be used when initializing the weights.
- https://en.wikipedia.org/wiki/Activation_function

Activation functions



Linear

$$y = x$$

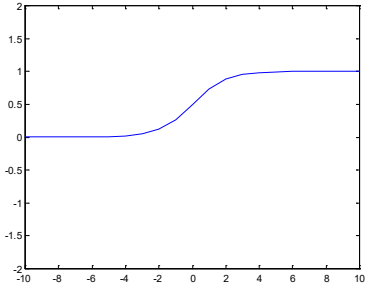


Hyperbolic tangent

$$y = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$

Logistic

$$y = \frac{1}{1 + \exp(-x)}$$



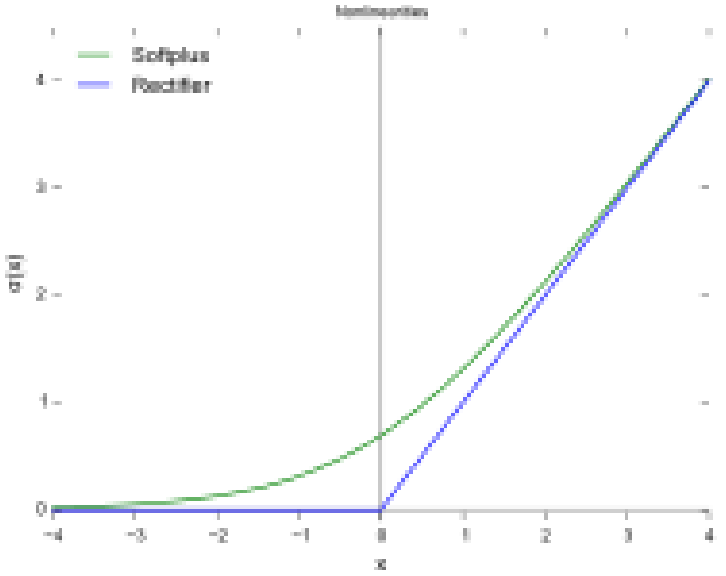
Rectifier / ramp function

f(x) = max(0,x)

x is the input to a neuron.

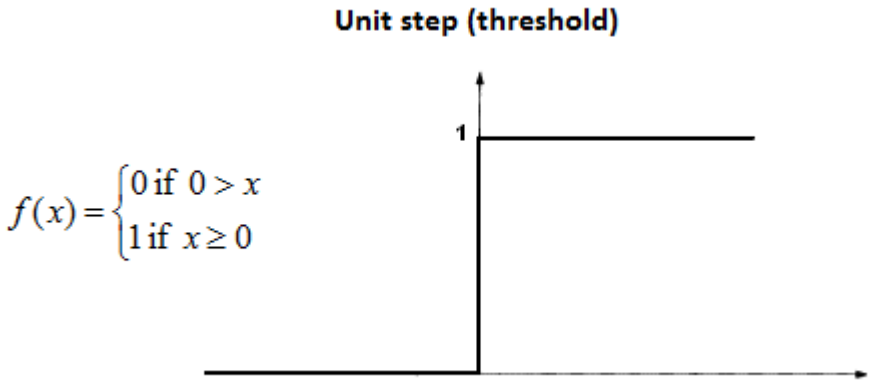
smooth approximation to the rectifier is softplus

$f(x) = \ln(1+e^x)$

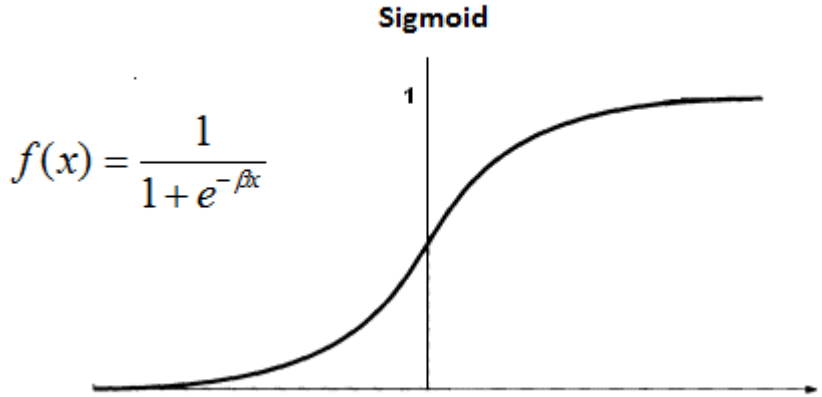


Activation functions

Unit step (threshold):
The transfer function translates the input signals to output signals. Four types of transfer functions are commonly used, Unit step (threshold), sigmoid, piecewise linear, and Gaussian.



Sigmoid:
The sigmoid function consists of 2 functions, logistic and tangential. The values of logistic function range from 0 and 1 and -1 to +1 for tangential function.



Deep Learning Algorithms

MLP – Multi Layer perceptron

- A multilayer perceptron (**MLP**) is a feed forward artificial **neural network** model that maps sets of input data onto a set of appropriate outputs.
- An **MLP** consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one.
- Had multiple hidden layers with logistic regression classifier transformation

Parameters to vary for tuning

- Number of layers
- Number of neurons in each layer
- Activation function in each layer
- Number of epochs
- Error/loss functions
- Iteration (equivalent to when a weight update is done)
- Learning rate (α)
  - Size of the step in the direction of the negative gradient
- Batch size
- Momentum parameter (weightage given to earlier steps taken in the process of gradient descent)
- Kernels
- Number of features
- Number of filters for images
- Filter sizes for images
- Gradient descent methods

Recap of evaluation measures

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Precision: When it predicts yes, how often is it correct?
TP/predicted yes = 100/110 = 0.91

"Sensitivity" or "Recall": When it's actually yes, how often does it predict yes?
TP/actual yes = 100/105 = 0.95

Accuracy: Overall, how often is the classifier correct?
(TP+TN)/total = (100+50)/165 = 0.91

Misclassification Rate: Overall, how often is it wrong?
(FP+FN)/total = (10+5)/165 = 0.09
equivalent to 1 minus Accuracy also known as "**Error Rate**"

False Positive Rate: When it's actually no, how often does it predict yes?
FP/actual no = 10/60 = 0.17

Specificity: When it's actually no, how often does it predict no?
TN/actual no = 50/60 = 0.83
equivalent to 1 minus False Positive Rate

Keras and Demos

Why we need Keras?

- **Keras**: Deep Learning library for Theano and TensorFlow
- An API spec for building DL models across many platforms

Guiding **principles**: modularity, minimalism, extensibility, and Python-nativeness

Simple

Keras' community is growing, while Theano's is declining

Less flexible

Less projects available online than caffe

Other alternate frameworks:

- Caffe
- Tensorflow
- Torch/PyTorch

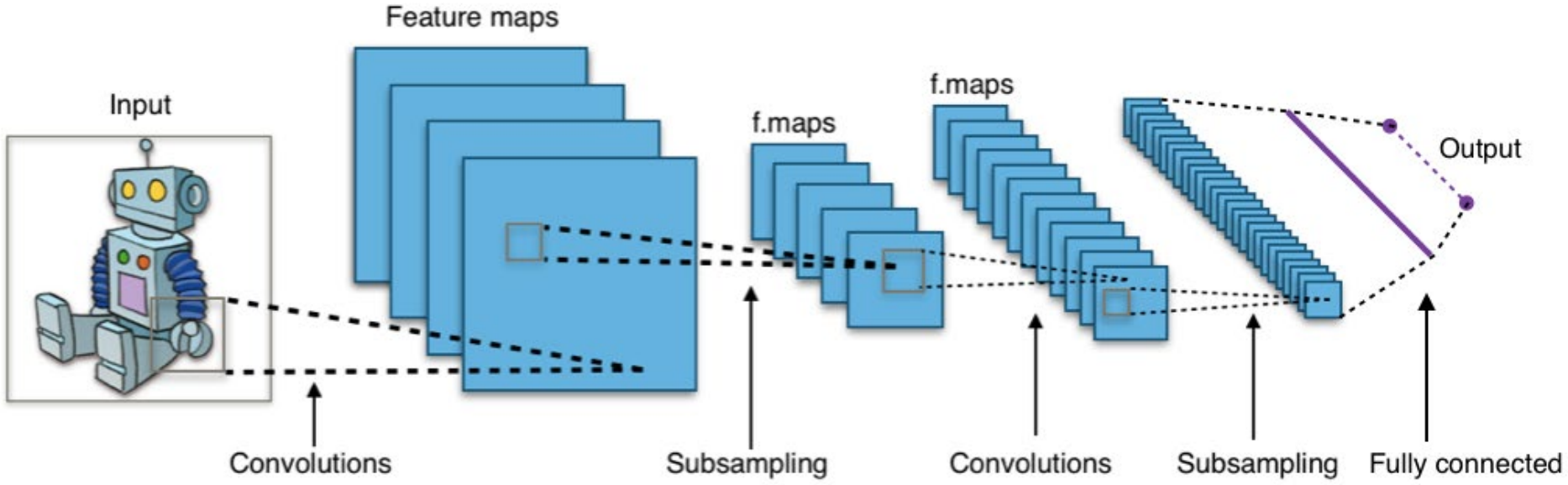
Keras+Tensorflow Demos

Convolutional Neural Networks

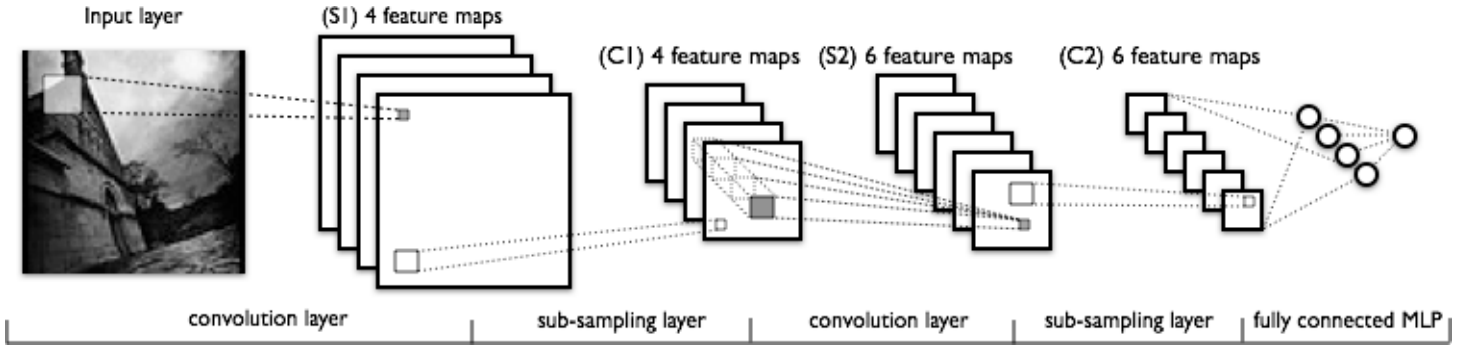
Convolutional Neural Network

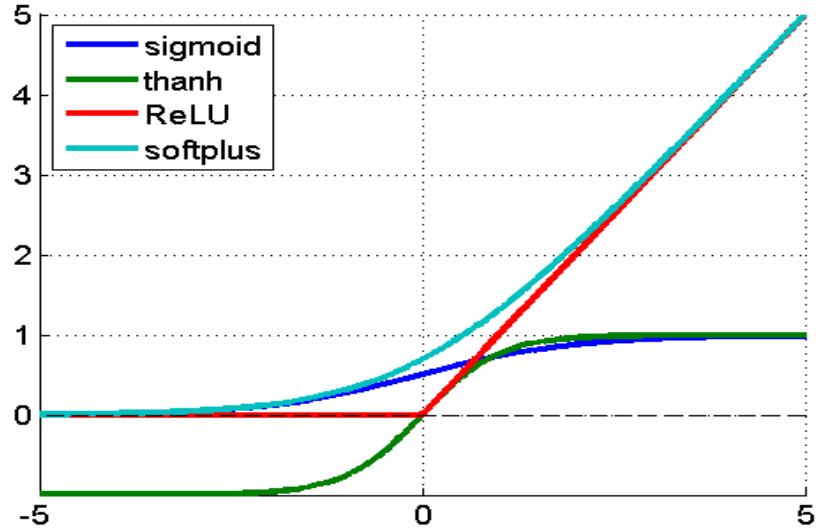
CNN - Convolutional Neural Network

- Feed-forward artificial neural network
- Convolutional networks were inspired by biological processes



Convolution operation





- Non-linearity is needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function

- Most deep networks use ReLU - max(0,x), since it trains much faster, is more expressive than logistic function and prevents the gradient vanishing problem.

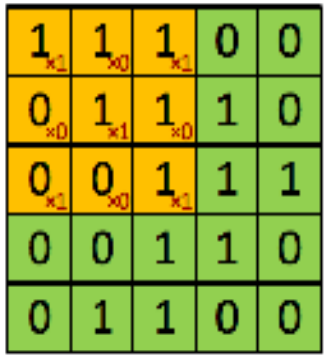
Convolution operation

Raw Image Pixel

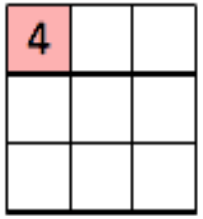
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter or Kernel or Feature detector

1	0	1
0	1	0
1	0	1



Image



Convolved Feature

Convolved Feature or Activation Map or the Feature Map.

Convolutional Neural Network

An image input constitutes a 3-dimensional structure called the *Input Volume* (255x255x3).

CNN's use **filters as kernels** where the parameters or weights have to be learnt.A filter is a matrix of lower size than the input to it.

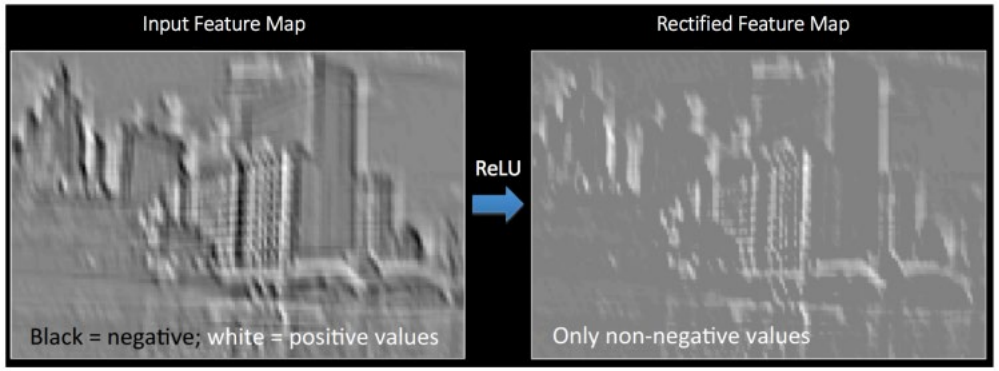
The inputs are convolved with the filters and passed through the activation function.

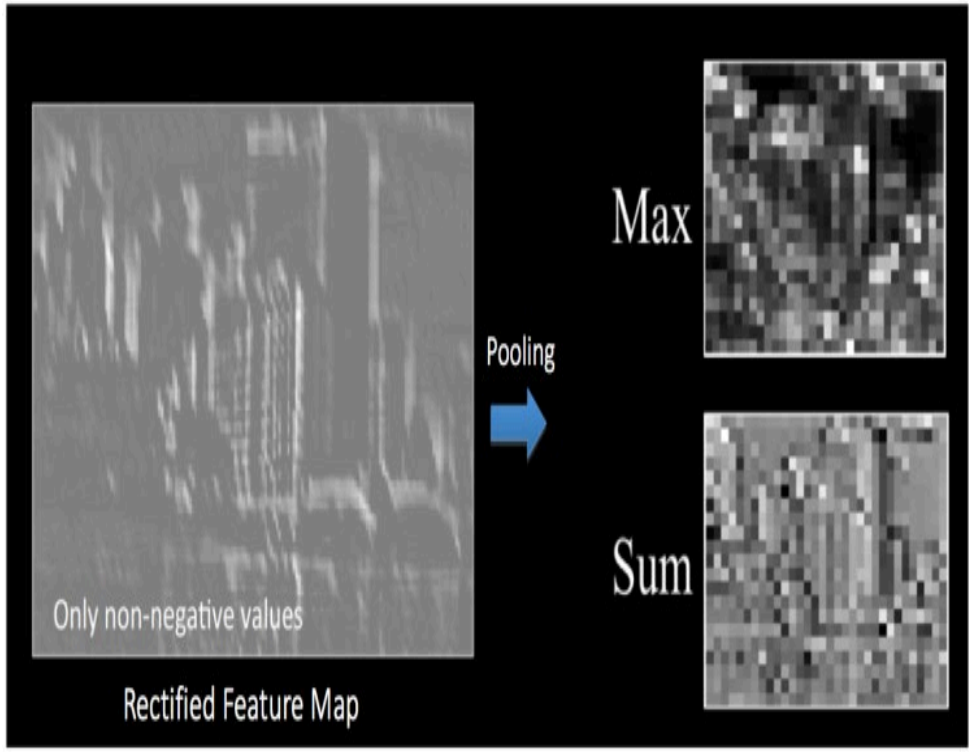
The weights of the kernels are randomly initialized and are modified during training based on error-minimization using **backpropagation**.

The real values of the **kernel matrix** change with each learning iteration over the training set, indicating that the network is learning to identify which regions are of significance for extracting features from the data.

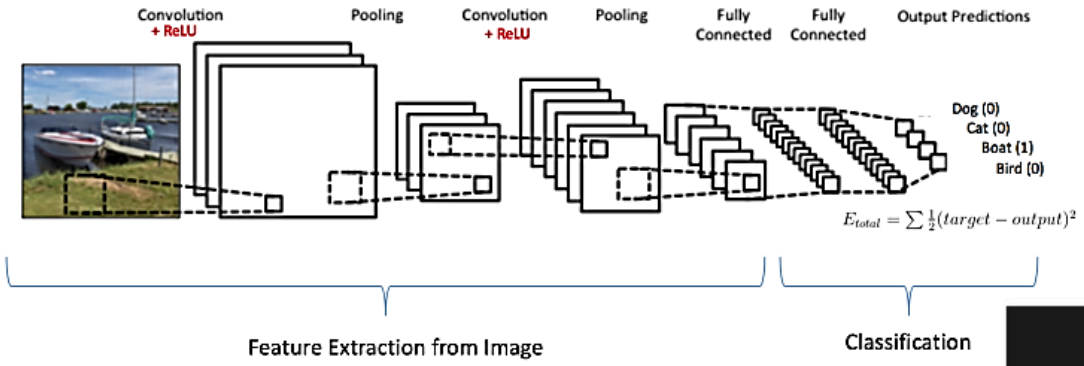
Stride: The shift of filter after each convolution.It can be increased from 1 to a larger value to decrease overfitting.

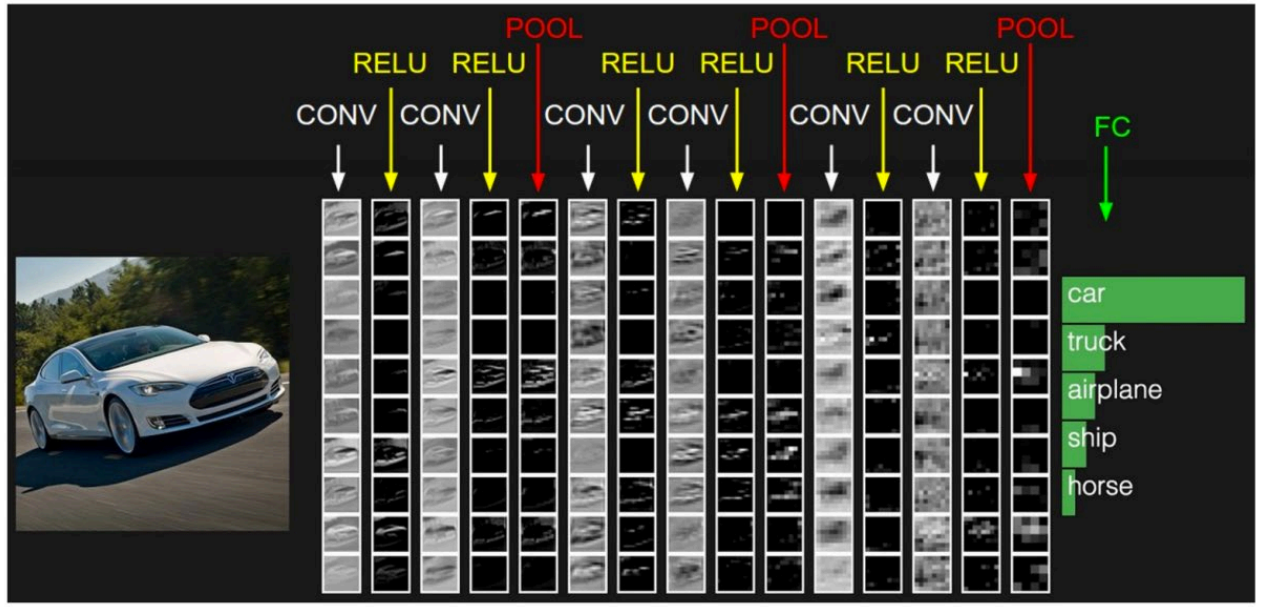
ReLu and Max pooling





CNN in summary





greatlearning
Power Ahead

RNN and LSTM

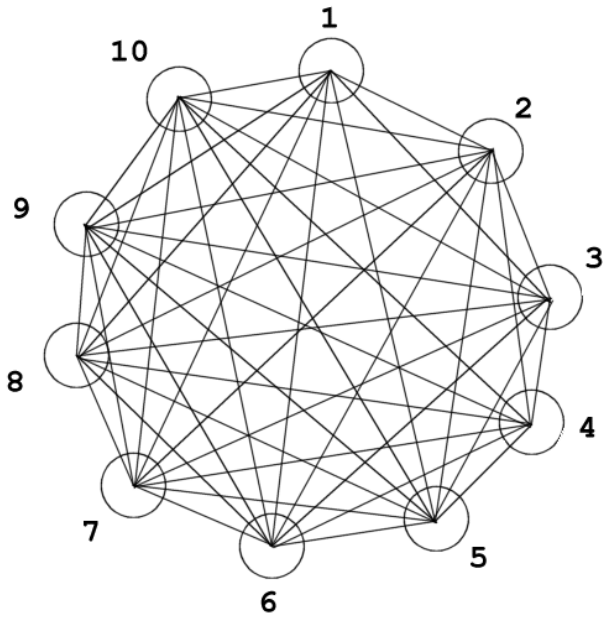
Recurrent Networks

Feed forward networks:

- Information only flows one way
- One input pattern produces one output
- No sense of time (or memory of previous state)

Recurrency

- Nodes connect back to other nodes or themselves
- Information flow is multidirectional
- Sense of time and memory of previous state(s)



Possible applications of RNN's are in domains where data is **seq**

For example:

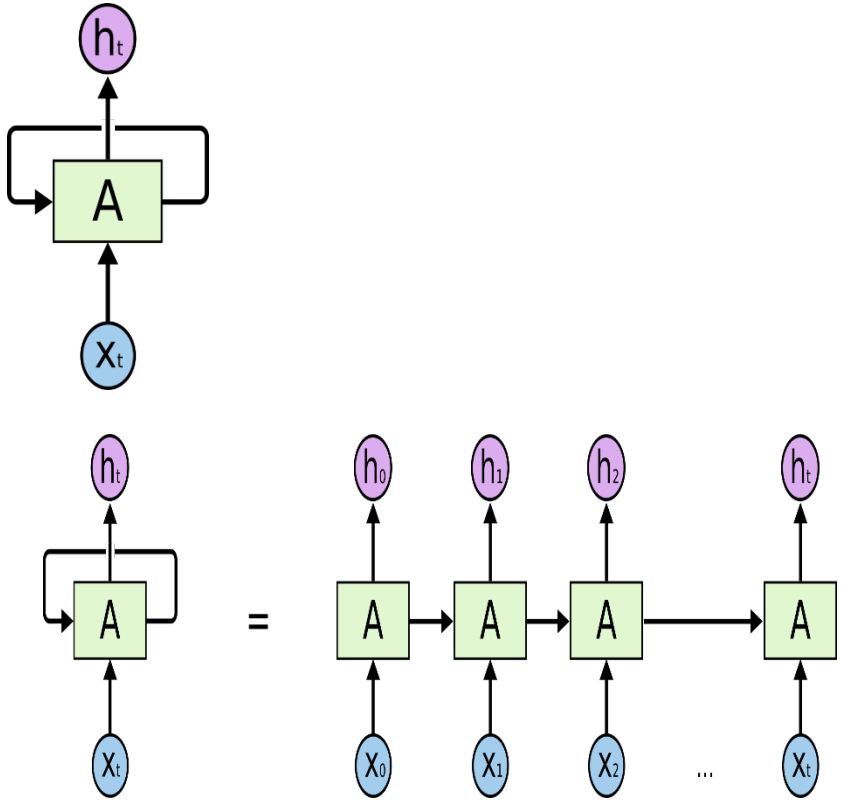
Speech and Text (NLP)

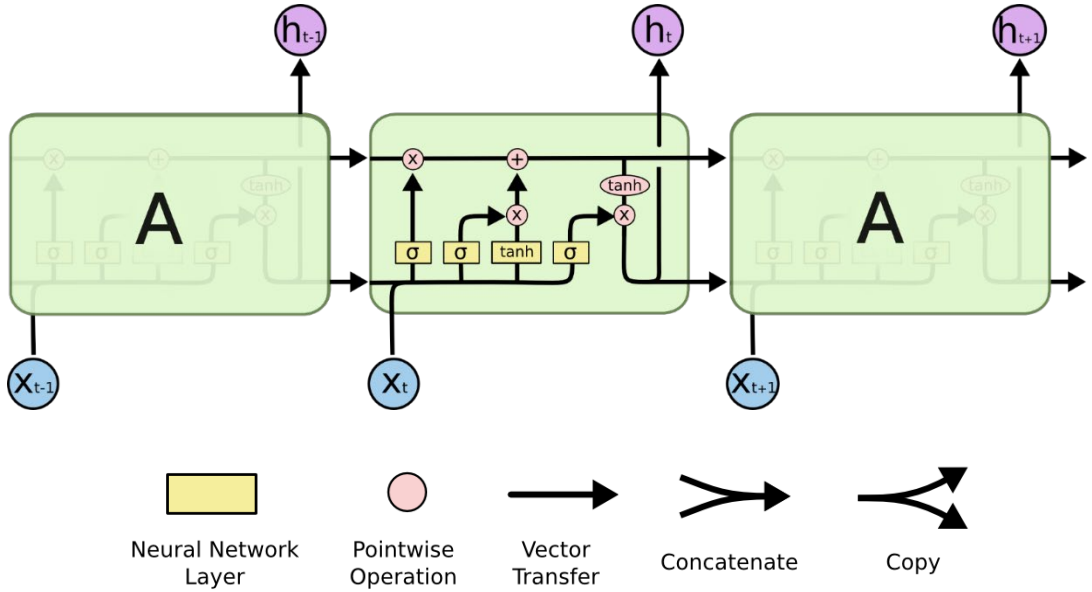
Music

Protein and DNA sequences

Time series from trade data

RNN and LSTM





LSTM (Long Short Term Memory)

It create a layer of Long-Short Term Memory units .

LSTM (**output_dim** , **activation**='tanh', **inner_activation**='hard_sigmoid')

output_dim: dimension of the internal projections and the final output.

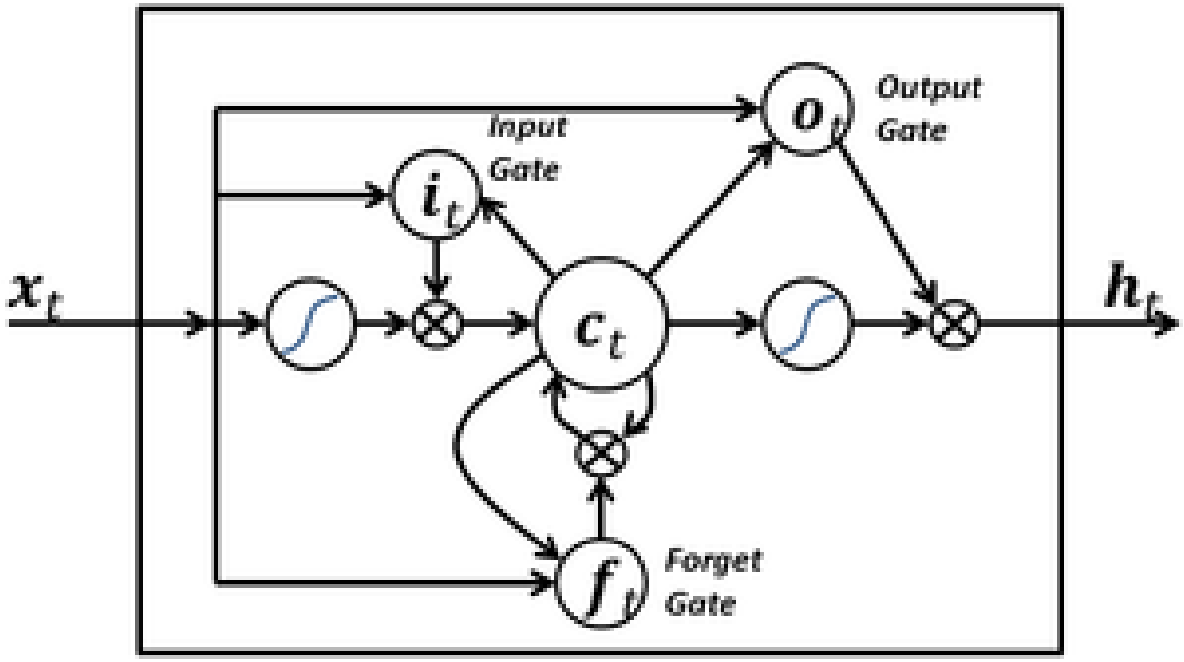
activation: name of activation function to use

Inner_activation: name of activation function to use for inner cells

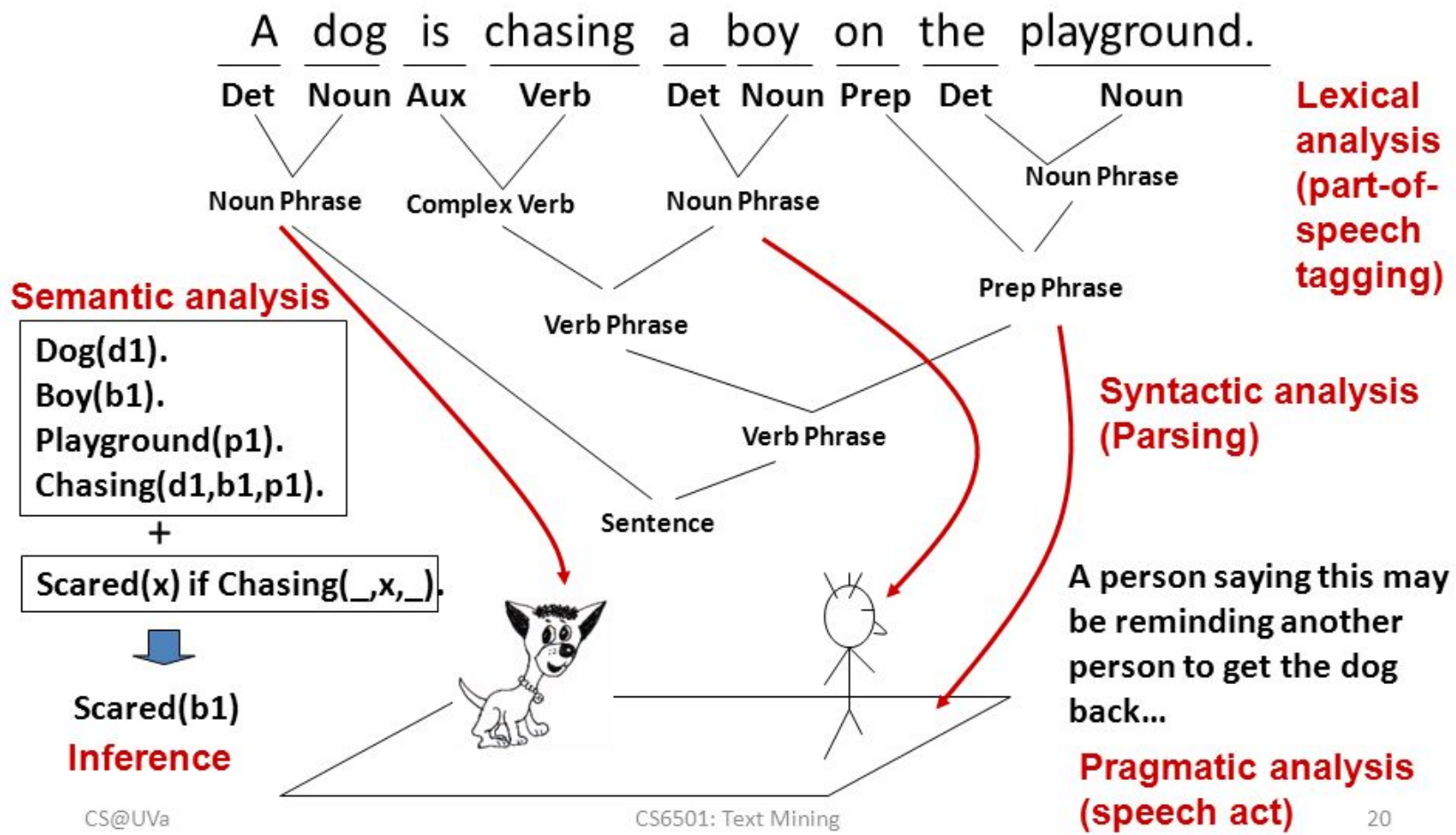
Long short-term memory

LSTM - Long short-term memory

- Recurrent neural network (RNN)
  - Take input not just the current input example they see, but also what they perceived one step back in time. **Feedback loop**, ingesting their own outputs moment after moment as input
- an LSTM network is well-suited to learn from experience to classify, process and predict time series
- LSTM blocks contain three or four "gates" that they use **to control the flow** of information into or out of their memory.



Text analysis

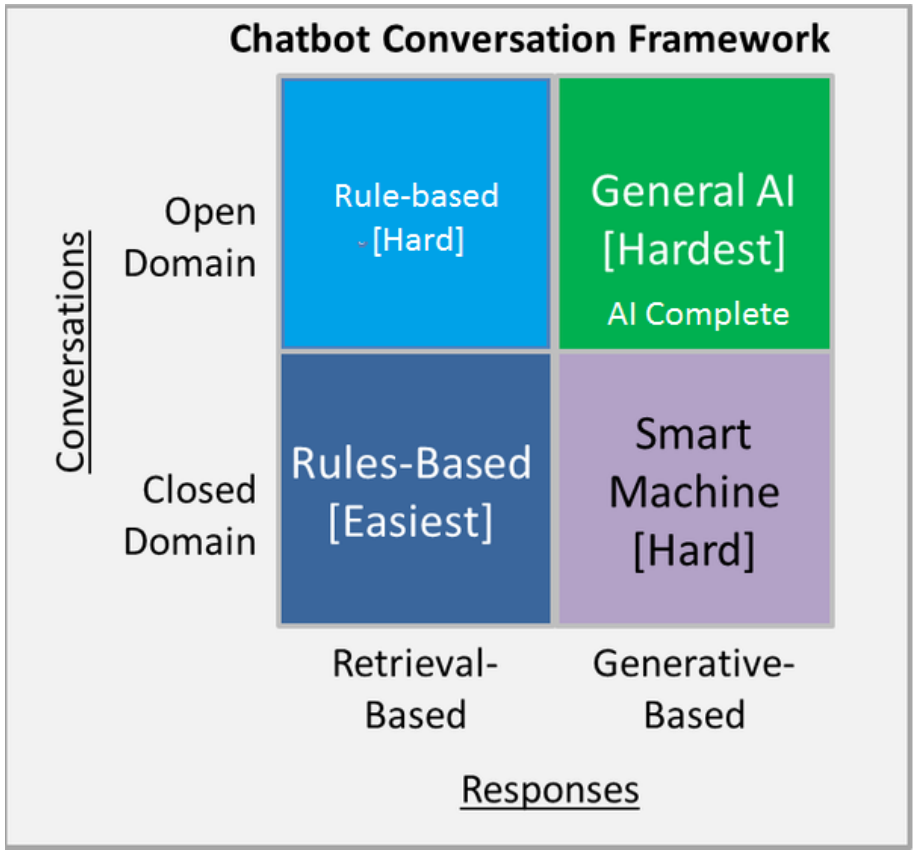


http://slideplayer.com/slide/9885340/

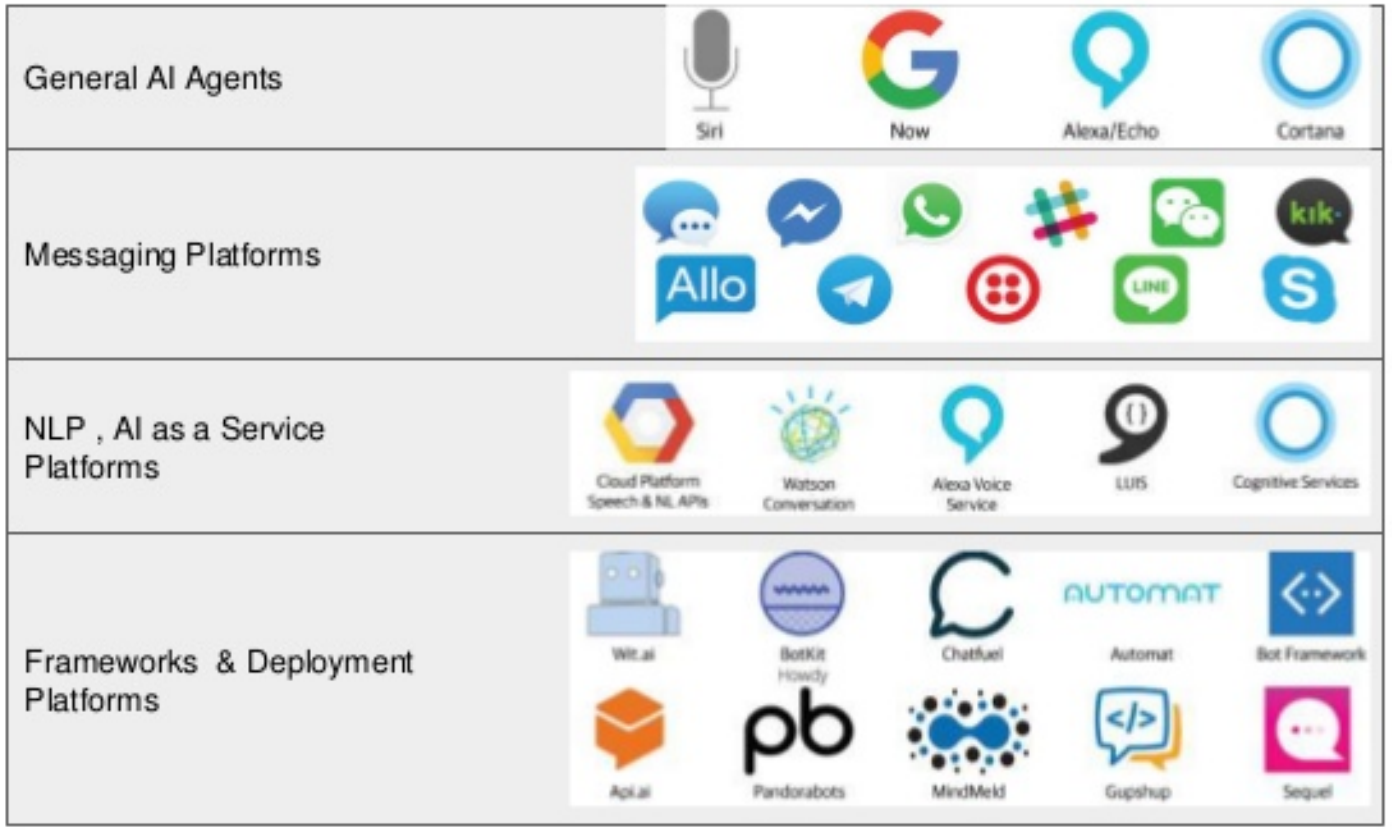
Type of chatbots

Usecases:

- Pizza Hut to help you order a pizza
- Uber to book a taxi
- CNN to keep you up-to-date with news content



Conversational interfaces

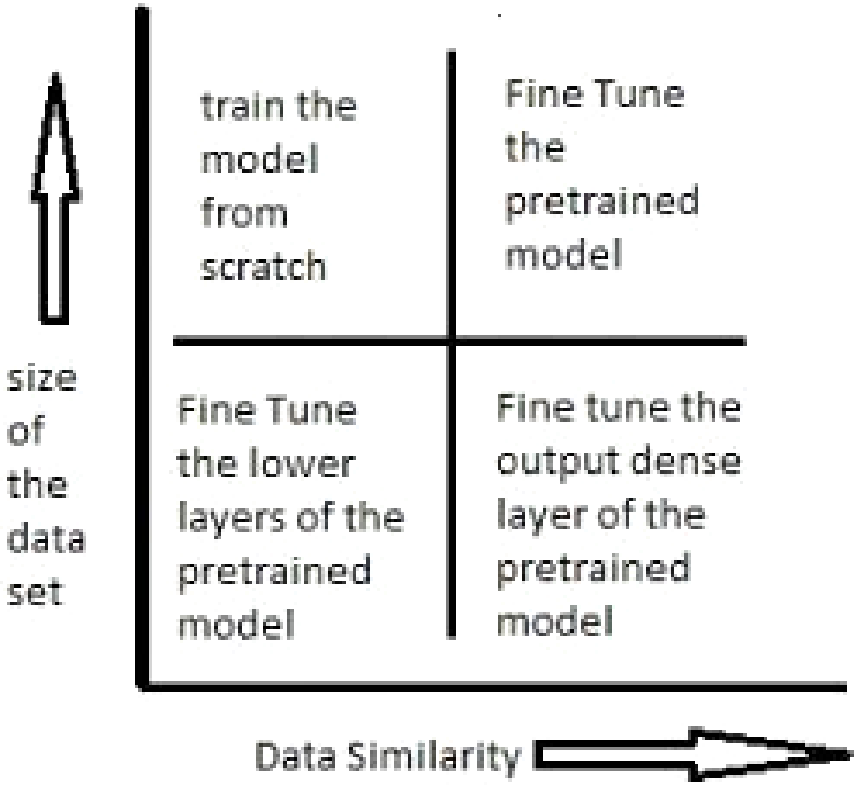


Source: https://www.oreilly.com/ideas/infographic-the-bot-platform-ecosystem

Pretrained models

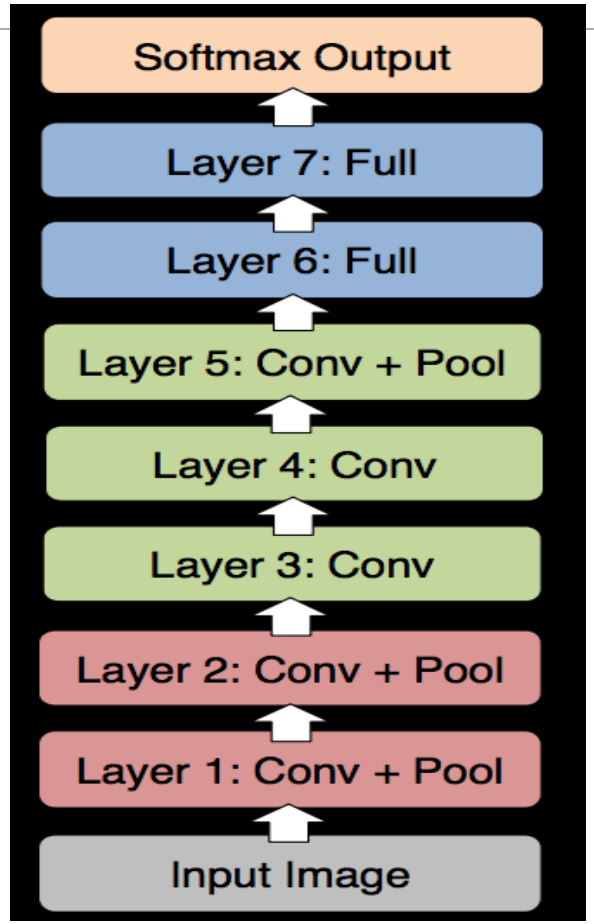
Convnets and pretrained models

- LeNet (1990s)
- AlexNet (2012)
- ZF Net (2013)
- GoogLeNet (2014)
- VGGNet (2014)
- ResNets (2015)
- DenseNet (August 2016)



Architecture of Alex Krizhevsky et al.

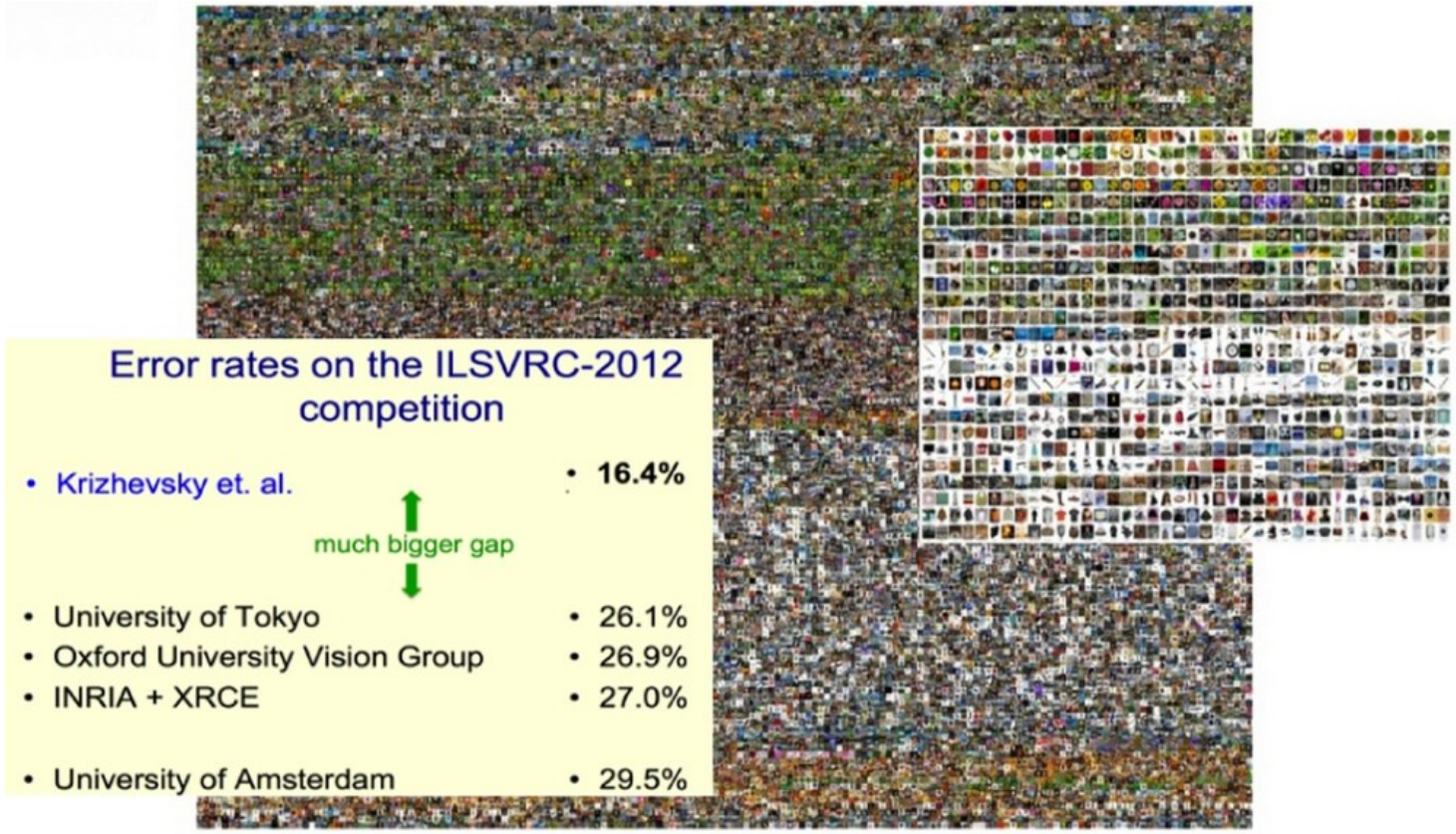
- 8 layers total
- Trained on Imagenet Dataset (1000 categories, 1.2M training images, 150k test images)
- 16.4% top-5 error
  - Winner of the ILSVRC- 2012 challenge.



Impact on Computer Vision

ImageNet Challenge2012

1.2M images with 1000 object categories



Practical tips

DL Project ideas

Image
- Captioning
- extract embedded text
- emoji - extract sentiment

Text
- Sarcasm
- Chatbots - specific topic
- Sentiment analysis
- Local languages

Speech
- Alexa / Home APIs
- Local languages

Numerical / Categorical
- Volume, result prediction
- Time series forecasting - weather / server
- Satillite data analysis - ISRO
- Govt data analysis
  - http://data.gov.in

DL project ideas contd

Video
- analytics - speed control
- search
- annotated data

Multimodal
- Chatbots
  - Get information from multiple sources
  - Generative

Robot
- Path planning / recommendations
  - reinforcement learning

Recommender systems
- Specific product / item category

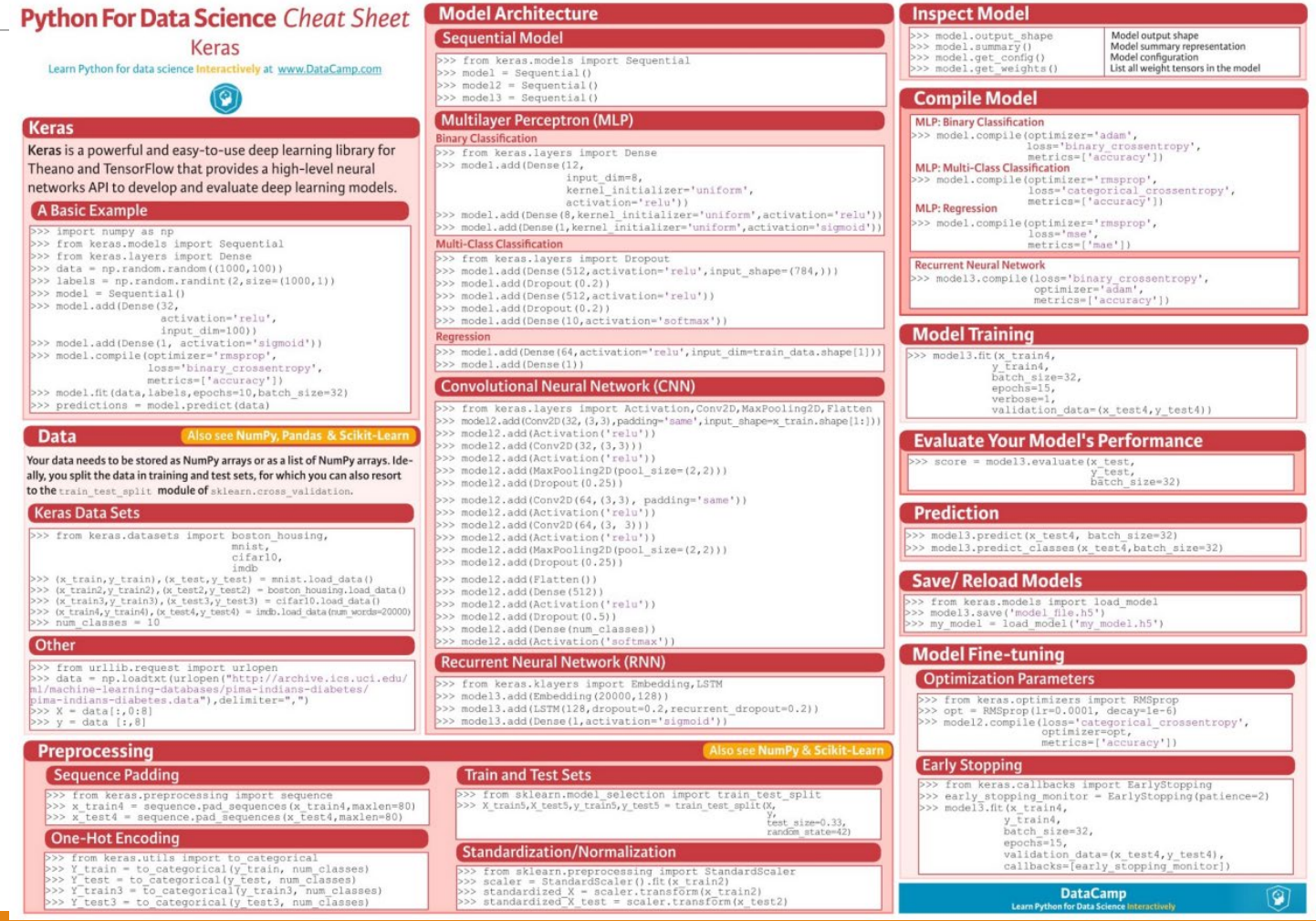
Practical challenges and tips while using ML / DL

- Data availability
- Rules based + ML

- Application revisited: Gien ticket data of client, identify the potential candidates for automation
  - Used ML for clustering the ticket data with preprocessing
  - Tried ML algorithms including ensemble to reach
  - Extended DL algorithms but the improvement is not more than 5%

- Discussion

Cheat sheets



Python For Data Science *Cheat Sheet*

Keras

Learn Python for data science Interactively at www.DataCamp.com

Keras

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> import numpy as np
>>> from keras.models import Sequential
>>> from keras.layers import Dense
>>> data = np.random.random((1000,100))
>>> labels = np.random.randint(2,size=(1000,1))
>>> model = Sequential()
>>> model.add(Dense(32,
                    activation='relu',
                    input_dim=100))
>>> model.add(Dense(1, activation='sigmoid'))
>>> model.compile(optimizer='rmsprop',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
>>> model.fit(data,labels,epochs=10,batch_size=32)
>>> predictions = model.predict(data)
```

Data

Also see NumPy, Pandas & Scikit-Learn

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, you split the data in training and test sets, for which you can also resort to the `train_test_split` module of `sklearn.cross_validation`.

Keras Data Sets

```
>>> from keras.datasets import boston_housing,
                               mnist,
                               cifar10,
                               imdb
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston_housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4),(x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
```

Other

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/
ml/machine-learning-databases/pima-indians-diabetes/
pima-indians-diabetes.data"),delimiter=",")
>>> X = data[:,0:8]
>>> y = data [:,8]
```

Model Architecture

Sequential Model

```
>>> from keras.models import Sequential
>>> model = Sequential()
>>> model2 = Sequential()
>>> model3 = Sequential()
```

Multilayer Perceptron (MLP)

Binary Classification

```
>>> from keras.layers import Dense
>>> model.add(Dense(12,
                    input_dim=8,
                    kernel_initializer='uniform',
                    activation='relu'))
>>> model.add(Dense(8,kernel_initializer='uniform',activation='relu'))
>>> model.add(Dense(1,kernel_initializer='uniform',activation='sigmoid'))
```

Multi-Class Classification

```
>>> from keras.layers import Dropout
>>> model.add(Dense(512,activation='relu',input_shape=(784,)))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(512,activation='relu'))
>>> model.add(Dropout(0.2))
>>> model.add(Dense(10,activation='softmax'))
```

Regression

```
>>> model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))
>>> model.add(Dense(1))
```

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation,Conv2D,MaxPooling2D,Flatten
>>> model2.add(Conv2D(32,(3,3),padding='same',input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32,(3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Conv2D(64,(3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64,(3, 3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

Recurrent Neural Network (RNN)

```
>>> from keras.klayers import Embedding,LSTM
>>> model3.add(Embedding(20000,128))
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
```

Inspect Model

```
>>> model.output_shape          Model output shape
>>> model.summary()             Model summary representation
>>> model.get_config()          Model configuration
>>> model.get_weights()         List all weight tensors in the model
```

Compile Model

MLP: Binary Classification

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

MLP: Multi-Class Classification

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

MLP: Regression

```
>>> model.compile(optimizer='rmsprop',
                  loss='mse',
                  metrics=['mae'])
```

Recurrent Neural Network

```
>>> model3.compile(loss='binary_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
```

Model Training

```
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               verbose=1,
               validation_data=(x_test4,y_test4))
```

Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test,
                            y_test,
                            batch_size=32)
```

Prediction

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

Save/ Reload Models

```
>>> from keras.models import load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

Model Fine-tuning

Optimization Parameters

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
                   optimizer=opt,
                   metrics=['accuracy'])
```

Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
               y_train4,
               batch_size=32,
               epochs=15,
               validation_data=(x_test4,y_test4),
               callbacks=[early_stopping_monitor])
```

Preprocessing

Also see NumPy & Scikit-Learn

Sequence Padding

```
>>> from keras.preprocessing import sequence
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
```

One-Hot Encoding

```
>>> from keras.utils import to_categorical
>>> Y_train = to_categorical(y_train, num_classes)
>>> Y_test = to_categorical(y_test, num_classes)
>>> Y_train3 = to_categorical(y_train3, num_classes)
>>> Y_test3 = to_categorical(y_test3, num_classes)
```

Train and Test Sets

```
>>> from sklearn.model_selection import train_test_split
>>> X_train5,X_test5,y_train5,y_test5 = train_test_split(X,
                                                         y,
                                                         test_size=0.33,
                                                         random_state=42)
```

Standardization/Normalization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(x_train2)
>>> standardized_X = scaler.transform(x_train2)
>>> standardized_X_test = scaler.transform(x_test2)
```

DataCamp
Learn Python for Data Science Interactively

Learning path

Building knowledge :

K1. Refresh your fundamentals on statistics, probability and linear algebra

K2. Do Course era deep learning

K3. Refer specific concepts in deeplearningbook.org by Goodfellow

K4. Refer advanced topics of deep learning based on the need - Generative adversial networks, Auto encoders, deep reinforcement learning, visualization techniques

K5. Attend webinars and AV meets or conferences to network and see latest trends

Practise with assignments and projects:

P1. Kaggle challenges with available data

Cuisine prediction, lung cancer,

Choose text or image or categorical/nurmerical probems

P2. Participate in hackathons and assesments

P3. Github profile and upload your codes

P4. Define your problem with your domain experience and follow steps of data science project execution with github repositories

Learning path references

K1. Relearning

https://in.udacity.com/course/intro-to-descriptive-statistics--ud827

https://www.khanacademy.org/math/linear-algebra

K2. Online courses:

https://www.coursera.org/specializations/deep-learning

https://in.udacity.com/course/deep-learning--ud730

K3: Books:

http://www.deeplearningbook.org/

http://neuralnetworksanddeeplearning.com/

http://deeplearning.stanford.edu/tutorial/

K4. Advanced topics

https://www.youtube.com/watch?v=RtxI449ZjSc&feature=relmfu

P1. Competitions

https://www.kaggle.com/competitions

http://www.image-net.org/challenges/LSVRC/

P2. Assessement of your skills

https://datahack.analyticsvidhya.com/contest/all/

P3. Github profile

References

- http://machinelearningmastery.com/4-steps-to-get-started-in-machine-learning/
- https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html
- https://archive.ics.uci.edu/ml/datasets/Adult
- http://www.datasciencecentral.com/profiles/blogs/the-data-science-project-lifecycle
- https://project.inria.fr/deeplearning/files/2016/05/DLFrameworks.pdf
- http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/
- http://cs231n.stanford.edu/
- http://www.iro.umontreal.ca/~lisa/pointeurs/theano_scipy2010.pdf
- http://www.kdnuggets.com/software/text.html
- http://neuralnetworksanddeeplearning.com/
- deepmind.com/blog
- http://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/
- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxhcmlqaXRsYWhhfGd4OjdhNzhiOTdlZjM4NTRiOWY
- Machine learning, Deep learning courses in CourseEra by Andrew NG
- https://becominghuman.ai/cheat-sheets-for-ai-neural-networks-machine-learning-deep-learning-big-data-678c51b4b463

greatlearning
Power Ahead

Thank you