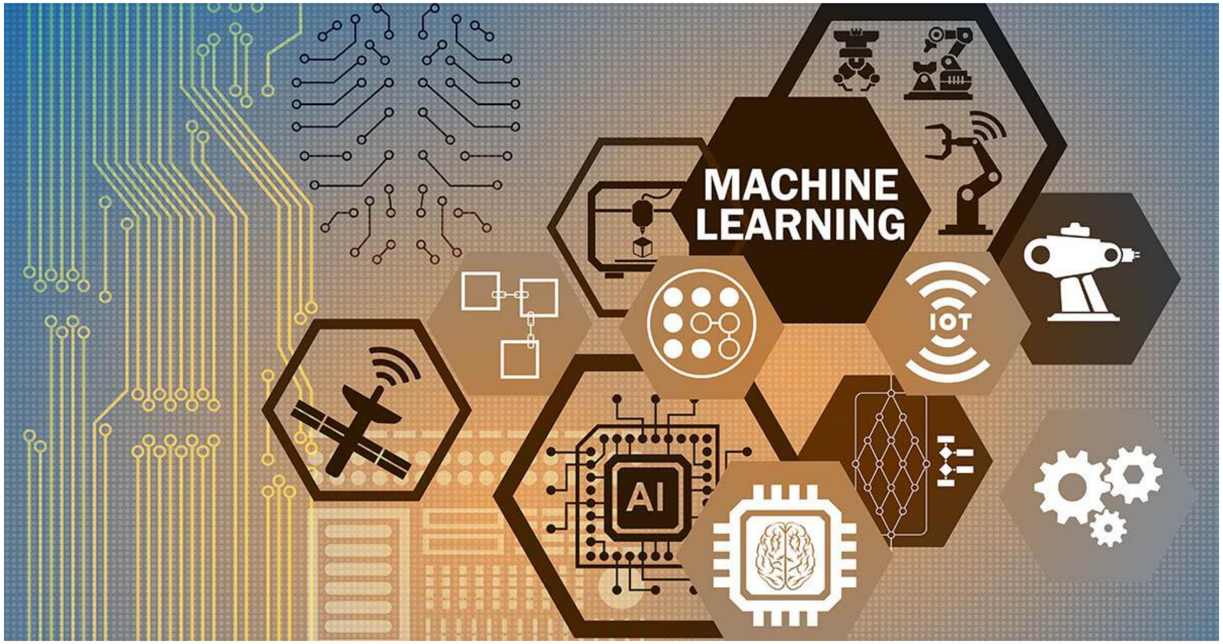


Applied Machine learning at Scale

By : Mehdi Zadeh

Nov 2021

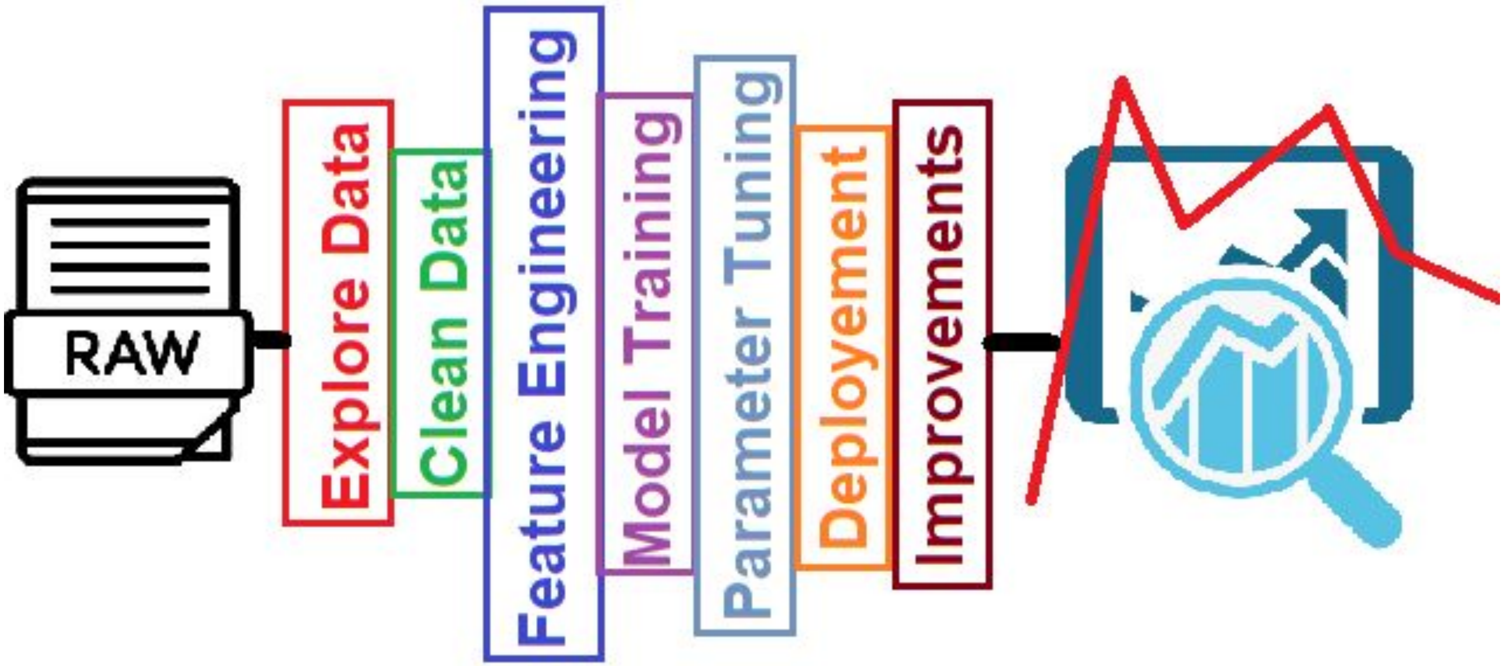
Outlines :

- **Applied Machine Learning**
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
  - Feature Extraction
  - Feature Selection
  - Conventional Machine Learning
  - Deep Learning Solution
  - Software and Hardware settings
  - Pre Processing Steps (IMPORTANT)
  - Improve Learning Performance (IMPORTANT)
  - Deploy Models; Production & @Scale (IMPORTANT)
  - Performance Measurement (IMPORTANT)

Data Science Skills

- Applied Math; Statistics and Probability principles (Minimum & In Practice : Undergrad courses)
- English Proficiency (Reading, Listening Skill(s) in Particular)
- Python Programming Language
- Applied Machine Learning concept (Conventional and Deep Learning)
- Python in Applied Machine Learning
- Data Analysis Frameworks (TensorFlow, Pytorch, Keras, MxNet ,...)
- Programming Optimization (GPU Programming, Speed, Serving, Server and Client, Multiple Concurrent Requests, package install from source, CPU execution of multiple instructions, ...)
- Containerization and Micro-Services Solutions (Process of scaling, managing, updating and removing containers; Docker and Kubernetes)
- EntryPoint Interface (API Gateway, Web Server Gateway Interface; FastAPI, Flask, Django, Unicorn, ...)
- Big Data Frameworks (Spark and Hadoop, in Particular)
- Databases (SQL , NoSQL)
- Cloud Computing (AWS, Google Cloud, Azure , ...)
- Version Control (Git, GitHub, Bitbucket, CodeCommit, ...)

Data Science Skills



Machine Learning Intro

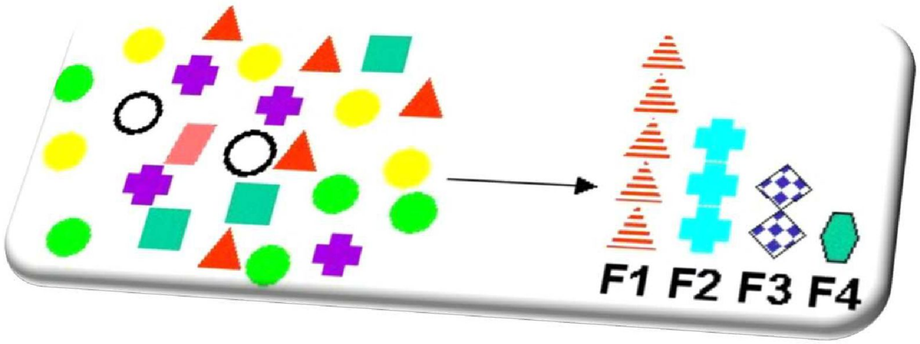
- **Machine learning (For dummies)**
  - Ability to learn without being explicitly programmed

  - **Supervised Learning**
    - Consist of a target / outcome variable
  - **Unsupervised Learning :**
    - No target or outcome variable to predict
  - **Reinforcement Learning:**
    - Trained to make specific decisions (trial and error).

Machine Learning Intro (Cont.)

- **Feature Extraction**
  - Intended to be informative and non-redundant.
  - Facilitating the subsequent learning, generalization steps.

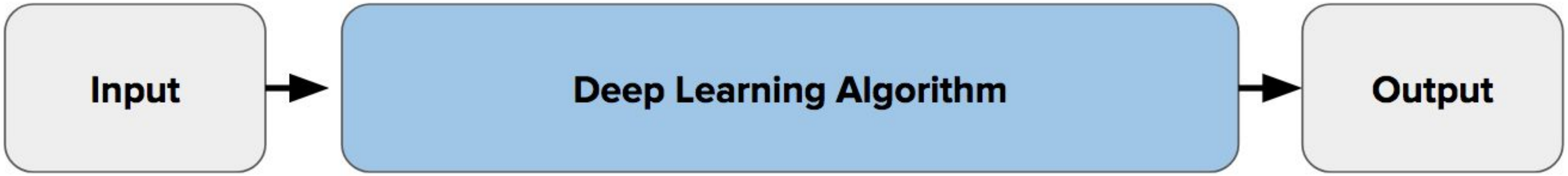
- **Feature Selection**
  - Discover correlation between features
  - Dominant features



Machine Learning Intro (Cont.)

- Conventional Machine Learning
- Deep Learning

Traditional Machine Learning Flow



Deep Learning Flow

Machine Learning Intro (Cont.)

- **Conventional Machine Learning**
- **Deep Learning**

statistical learning where each instance in a dataset is described by a set of features or attributes. Feature extraction and feature selection should be done by human intervention.

In contrast, the term “**Deep Learning**” is a method of **statistical learning** that extracts features or attributes from raw data.

Machine Learning Intro (Cont.)

- **Conventional Machine learning (Dummies):**
  - ❑ Linear Regression
  - ❑ Logistic Regression
  - ❑ Decision Tree
  - ❑ SVM
  - ❑ Naive Bayes
  - ❑ KNN
  - ❑ K-Means
  - ❑ Random Forest
  - ❑ Neural Network ------> Deep Learning
- **References :**
  - https://machinelearningmastery.com/
  - https://towardsdatascience.com/
  - https://medium.com/
  - https://www.edureka.co/blog/interview-questions/data-science-interview-questions/#basic

Machine Learning Intro (Cont.)

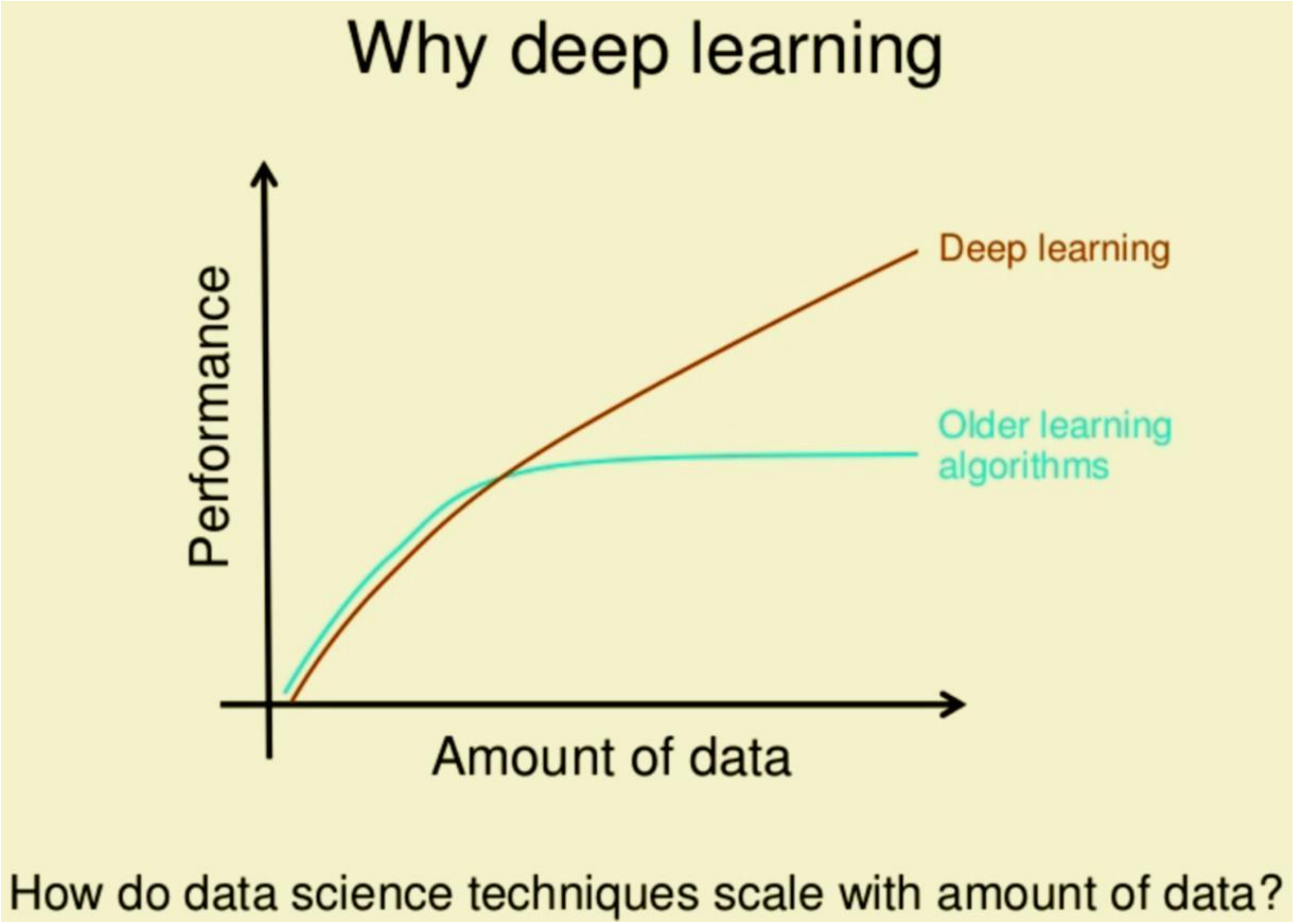
- For Example: **Naïve Bayes (Dummies):**
  - Bayes' theorem with an assumption of independence between predictors
  - Presence of a particular feature in a class is unrelated to the presence of any other feature

An apple if it is red, round, and about 3 inches in diameter.

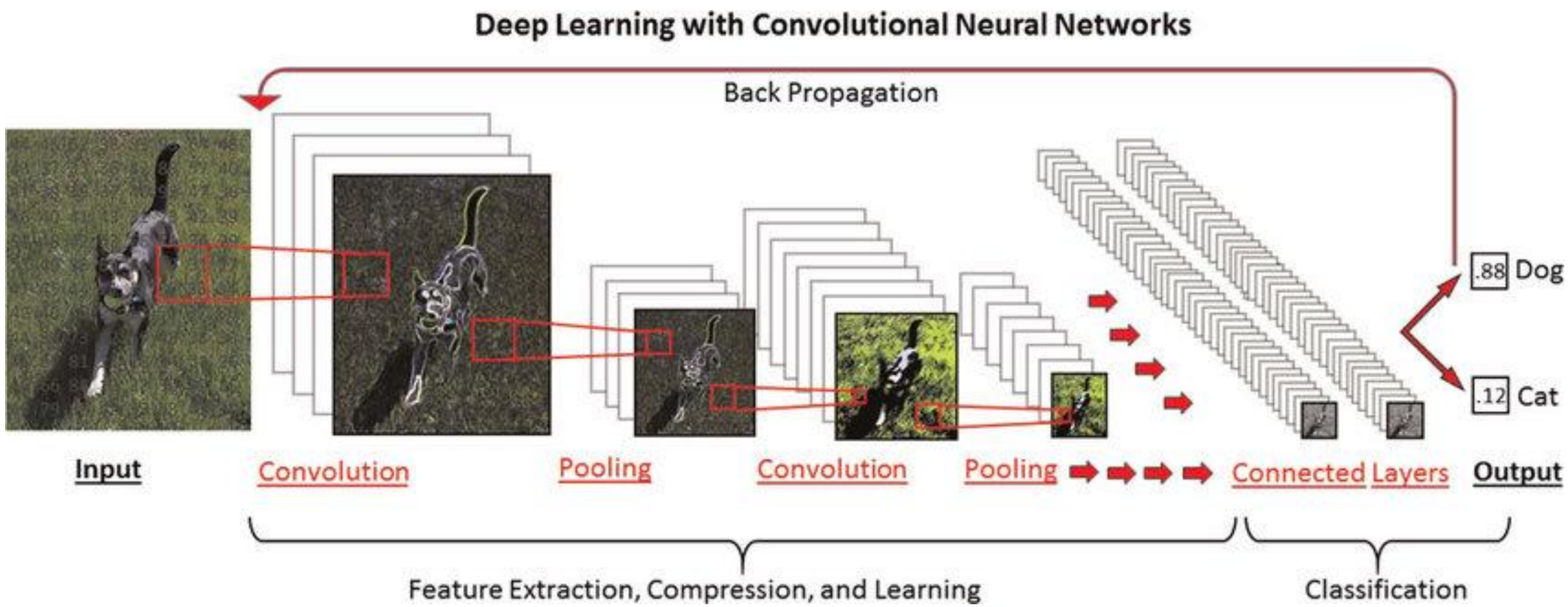
Naive Bayes :

All of these properties to independently contribute to the probability

Machine Learning Intro (Cont.)



Machine Learning Intro (Cont.)



Convolutional Neural Networks

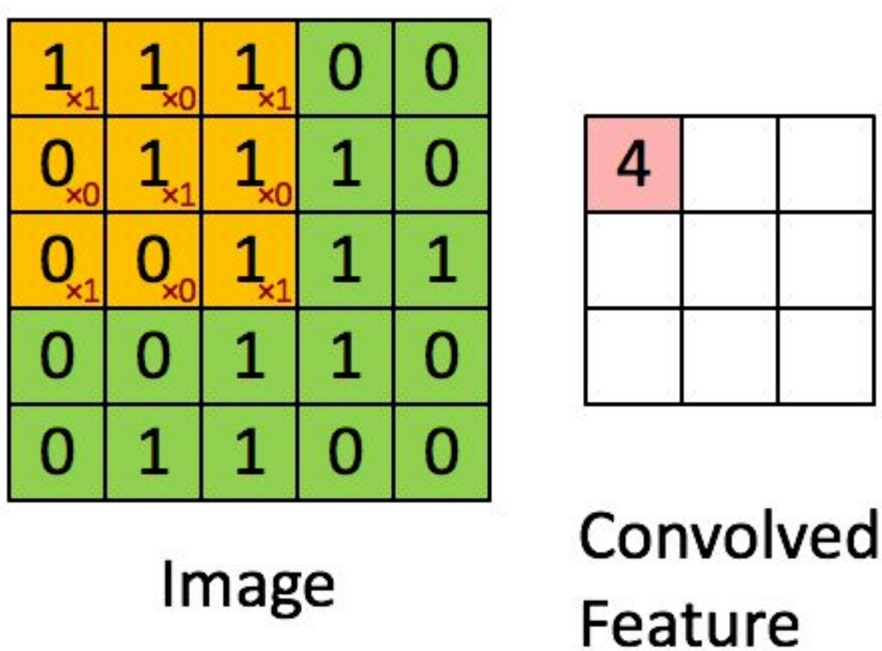
Guide to Convolutional Convolutional Neural Networks

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

Convolutional Neural Networks

Guide to Convolutional Convolutional Neural Networks



Machine Learning Intro (Cont.)

Hyperparameter tuning in Deep Learning :

- Number of Layers
- Number of Neurons
- Types of Layers
- Activation Functions
- Batch Sizes
- Initialization
- Batch Normalization and Layer Normalization
- Optimizers
- Learning Rate and Scheduling
- Regularization
- Dropout

Machine Learning Intro (Cont.)

Improve Deep Learning Algorithm Performance

- ❑ Get Data (Deep learning get better)
- ❑ Generate Data. (Data Augmentation / adding noise, call adding jitter)
- ❑ Rescale Data. (Rescale data to the bounds of activation functions.)
- ❑ Transform Your Data. (Guesstimate the univariate distribution)
- ❑ With Algorithms (Steals from literature)
- ❑ With Algorithm Tuning (tuning hyperparameters)
- ❑ Improve Performance With Ensembles.

Ref : https://machinelearningmastery.com/improve-deep-learning-performance/

Machine Learning Intro (Cont.)

Generate Data. (Data Augmentation / adding noise, call adding jitter)

Image :

Perspective Skewing

Elastic Distortions

Rotating

Shearing

Cropping

Mirroring

Colour distortion

Ex: https://augmentor.readthedocs.io/en/master/

Text:

Generate conditional synthetic text samples

Ex: https://github.com/openai/gpt-2

Machine Learning Intro (Cont.)

Rescale Data

- Normalized to 0 to 1.
- Rescaled to -1 to 1.

Transform Your Data (Numerical data):

- Like a skewed Gaussian, adjusting the skew with a Box-Cox transform.
- Like an exponential distribution, adjusting with a log transform.

Machine Learning Intro (Cont.)

With Algorithm Tuning (tuning hyperparameters):

- Diagnostics.
- Weight Initialization.
- Learning Rate.
- Activation Functions.
- Batches and Epochs.
- Regularization.
- Optimization and Loss.
- Early Stopping.

Machine Learning Intro (Cont.)

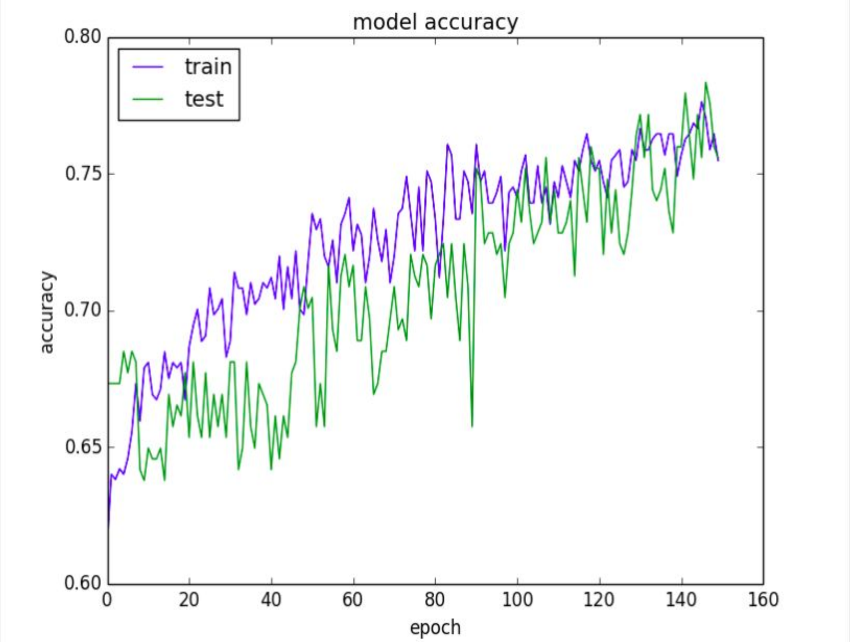
Diagnostics:

Get insight into the learning behavior of your model
Plotting training vs validation over epoches

Overfitting:
If training is much better than the validation; use regularization technique.

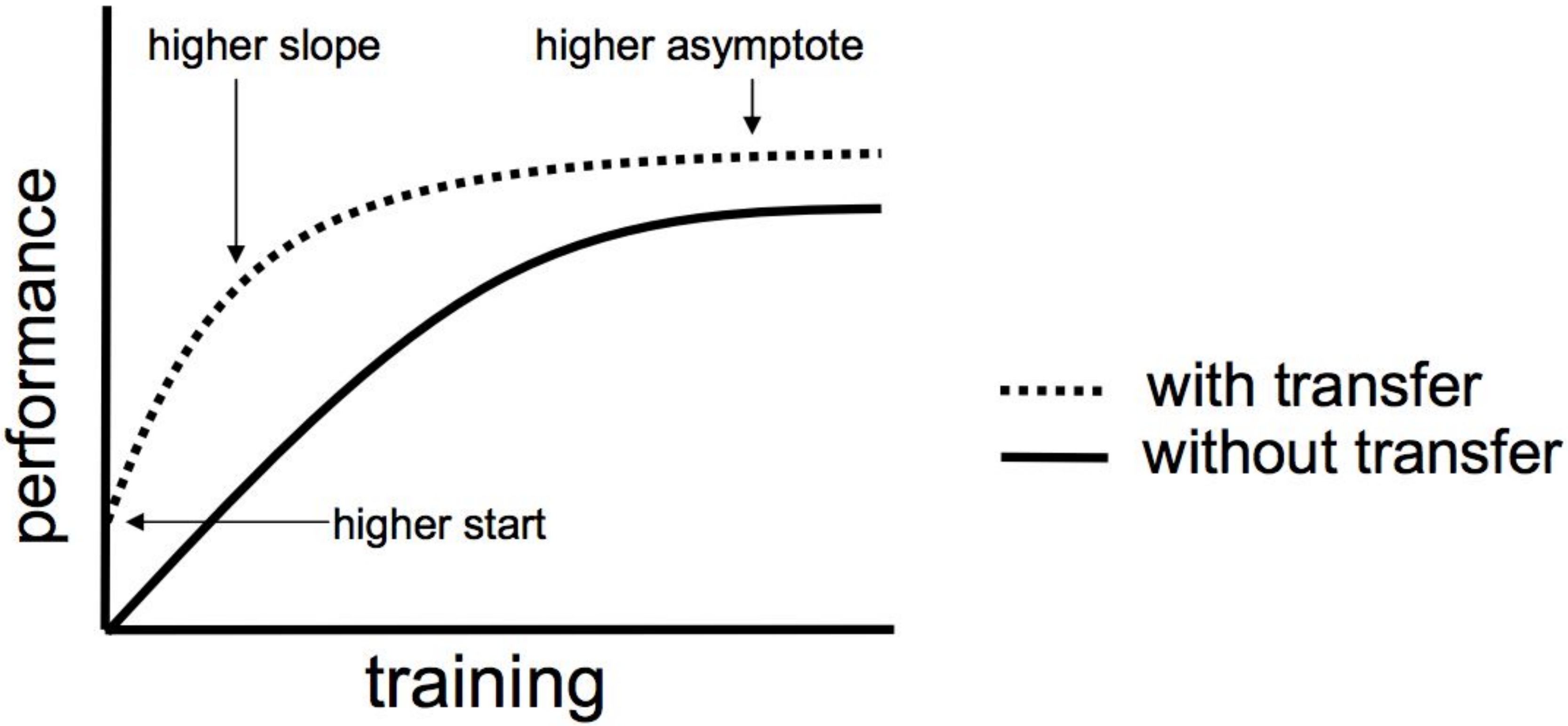
Underfitting;
If training and validation are both low; review all network settings

If an inflection point
when training goes above the validation; use early stopping.



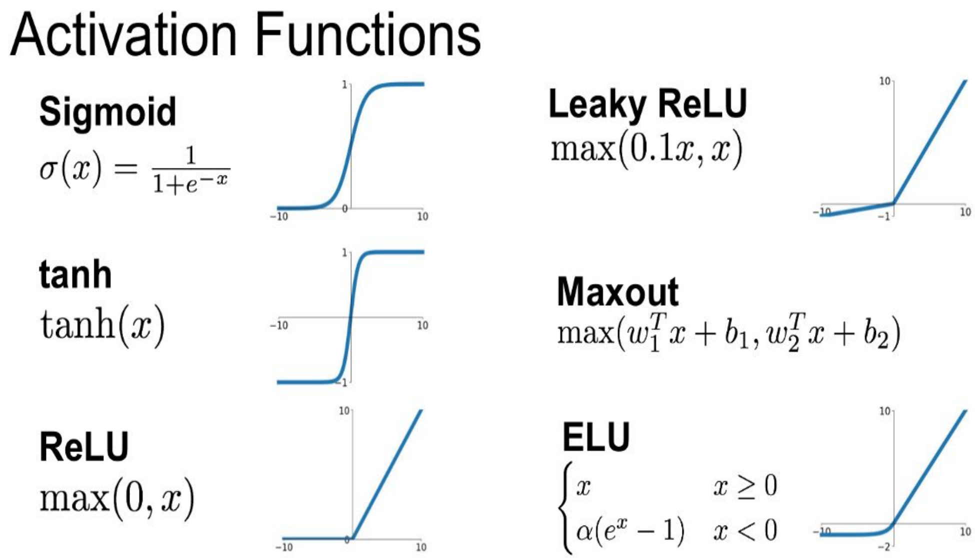
Plot of Model Accuracy on Train and Validation Datasets

Machine Learning Intro (Cont.)



Activation Function

- exponential
- get
- hard_sigmoid
- linear
- relu
- selu
- serialize
- sigmoid
- softmax
- softplus
- softsign
- tanh



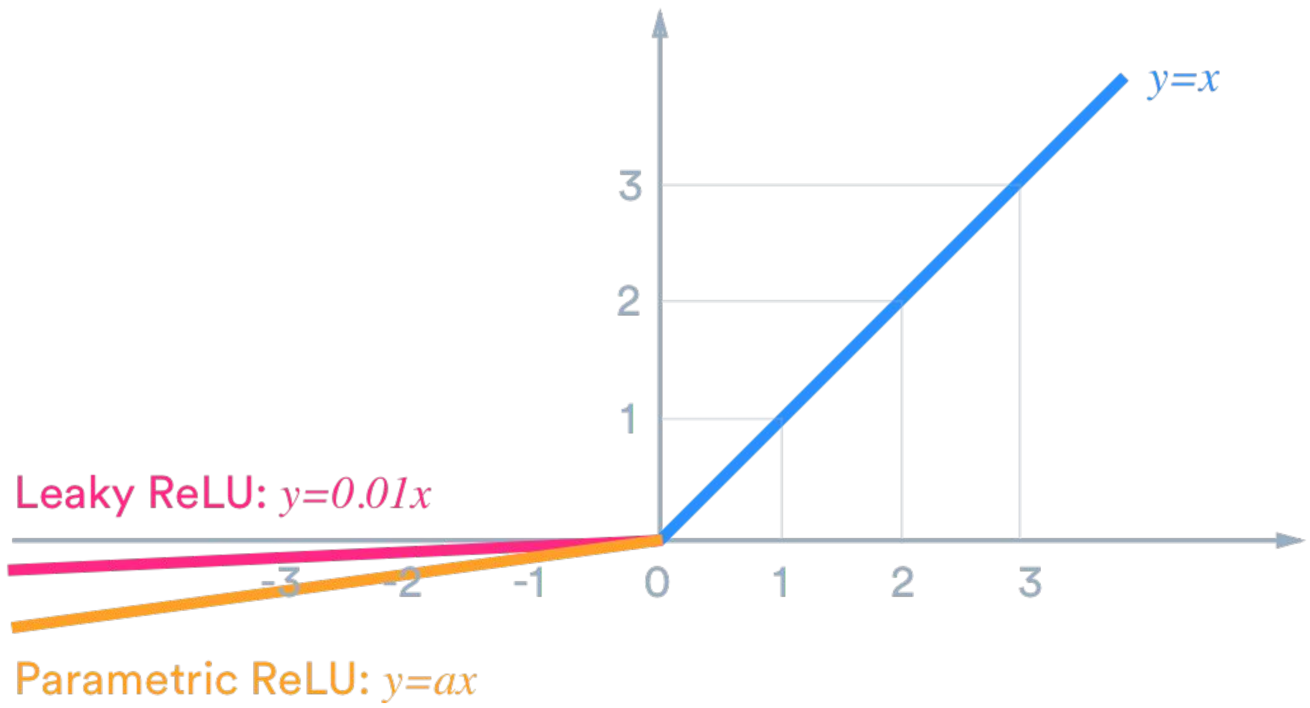
Activation Function

Choosing the right Activation Function :

- The softmax function is often in the final layer of a deep learning.
- ReLU function is only in the hidden layers (intermediate layers)
- ReLU function is a general activation function in most cases.
- In presence of dead neurons; the leaky ReLU function is the best choice
- Always begin with using ReLU function and then move over to other activation functions in case ReLU doesn’t provide with optimum results)

More info here :
https://www.analyticsvidhya.com/blog/2021/01/fundamentals-deep-learning-activation-functions-when-to-use-them/
https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7
https://datascience.stackexchange.com/questions/5706/what-is-the-dying-relu-problem-in-neural-networks

Activation Function



Batch size

- Larger batch sizes result in faster progress in training, but don't converge as fast.
- Smaller batch sizes train slower, but can converge faster.
- Large batch size ignore details in training while small batch size go to details. It's definitely problem dependent.
- Large batch size is good at very distinguable problems and requires enough memory size while small batch works perfect for very complex and similar classes.

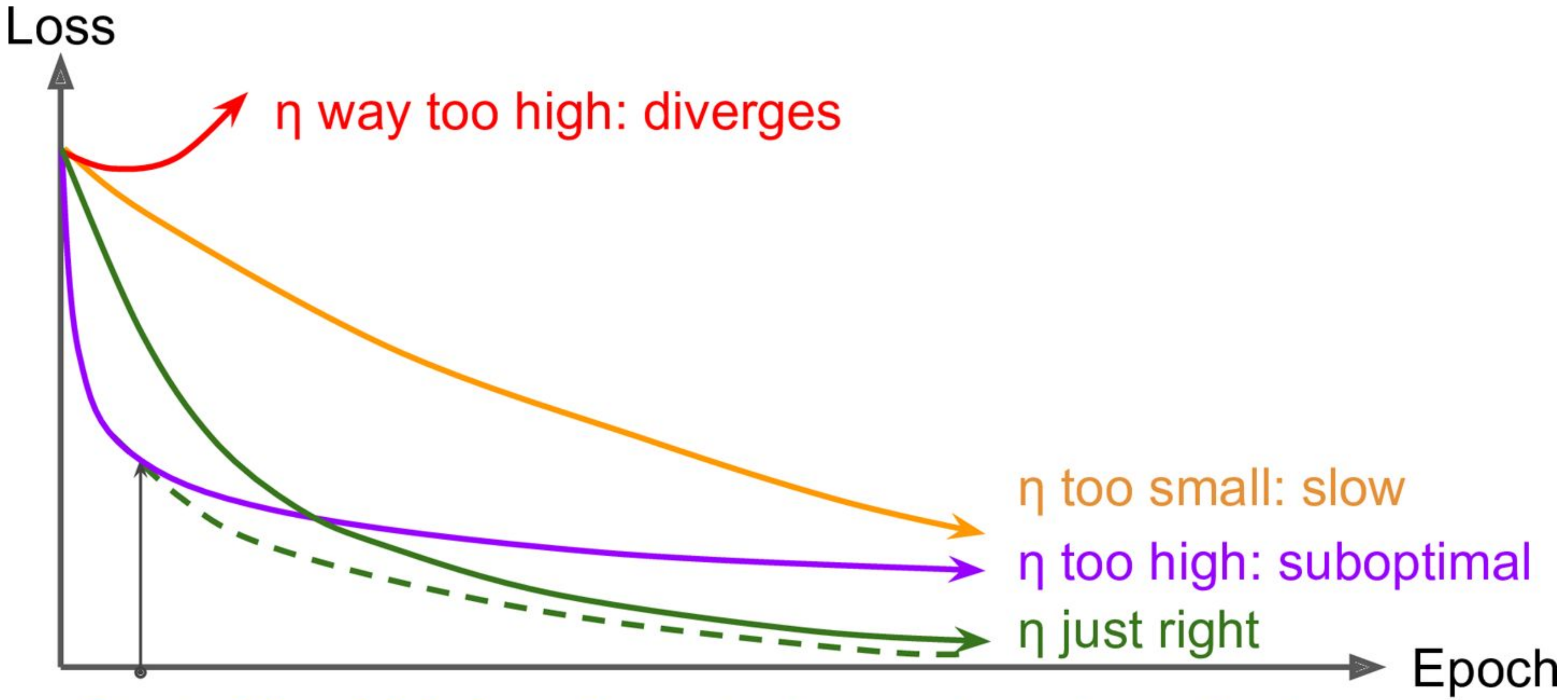
Be safe using a batch size of 8,16,32,64 which are pretty standard

More info here:
https://stackoverflow.com/questions/35050753/how-big-should-batch-size-and-number-of-epochs-be-when-fitting-a-model-in-keras#:~:text=In%20general%3A%20Larger%20batch%20sizes,of%20training%2C%20to%20a%20point.

Optimizer

Class	Convergence speed	Convergence quality
`SGD`	*	***
`SGD(momentum=...)`	**	***
`SGD(momentum=..., nesterov=True)`	**	***
`Adagrad`	***	* (stops too early)
`RMSprop`	***	** or ***
`Adam`	***	** or ***
`Nadam`	***	** or ***
`AdaMax`	***	** or ***

Learning Rate



Start with a high learning rate then reduce it: perfect!

Learning Rate

Learning Rate Schedule For Training Models

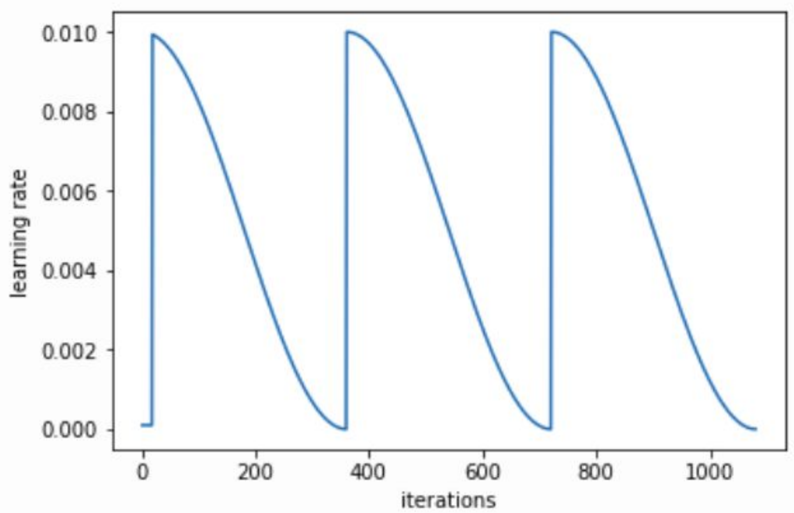
Decrease the learning rate gradually based on the epoch. (argument called decay)

Decrease the learning rate using punctuated large drops at specific epochs.

Cyclical Learning Rates

More info here :

https://github.com/mhmoodlan/cyclic-learning-rate
https://brandonmorris.dev/2018/06/24/mastering-the-learning-rate/



Four Parameters

Learning rate is coupled with :

Number of training epochs

Batch size and

Optimization method.

Learning Methods (Cont.)

- **Some Deep Learning Methods :**

  - *Image processing :*
    - Convolutional Neural Network
    - Inception ResNet
    - Fast R-CNN
    - U-Net

  - *Text Mining :*
    - HAN
    - FastText
    - *Transformers* (Google BigBird , BERT, GPT-3, RoBERTa, XLM, DistilBert, XLNet…)

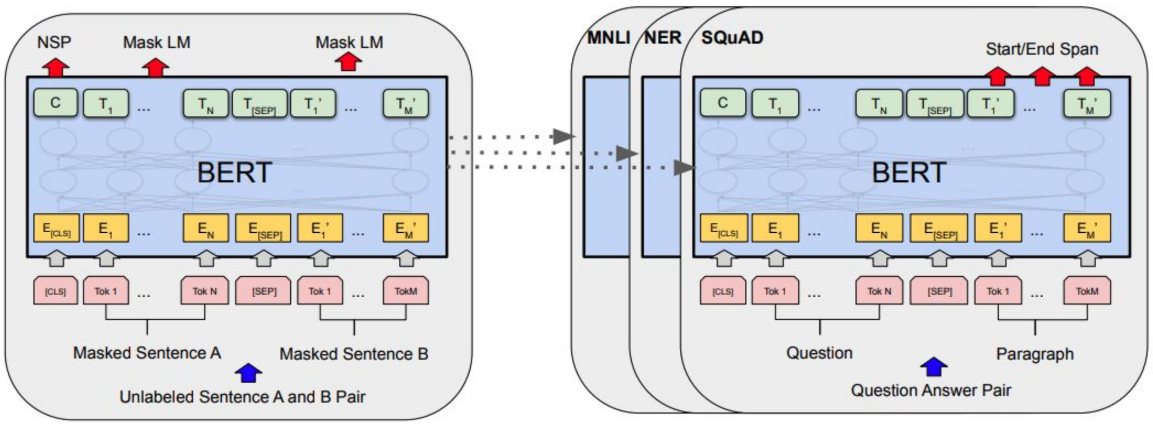
  - *Numerical Data :*
    - *Time2Vec*

- **Transformer**

BERT in details :

https://towardsdatascience.com/understanding-bert-is-it-a-game-changer-in-nlp-7cca943cf3ad

https://neptune.ai/blog/bert-and-the-transformer-architecture-reshaping-the-ai-landscape

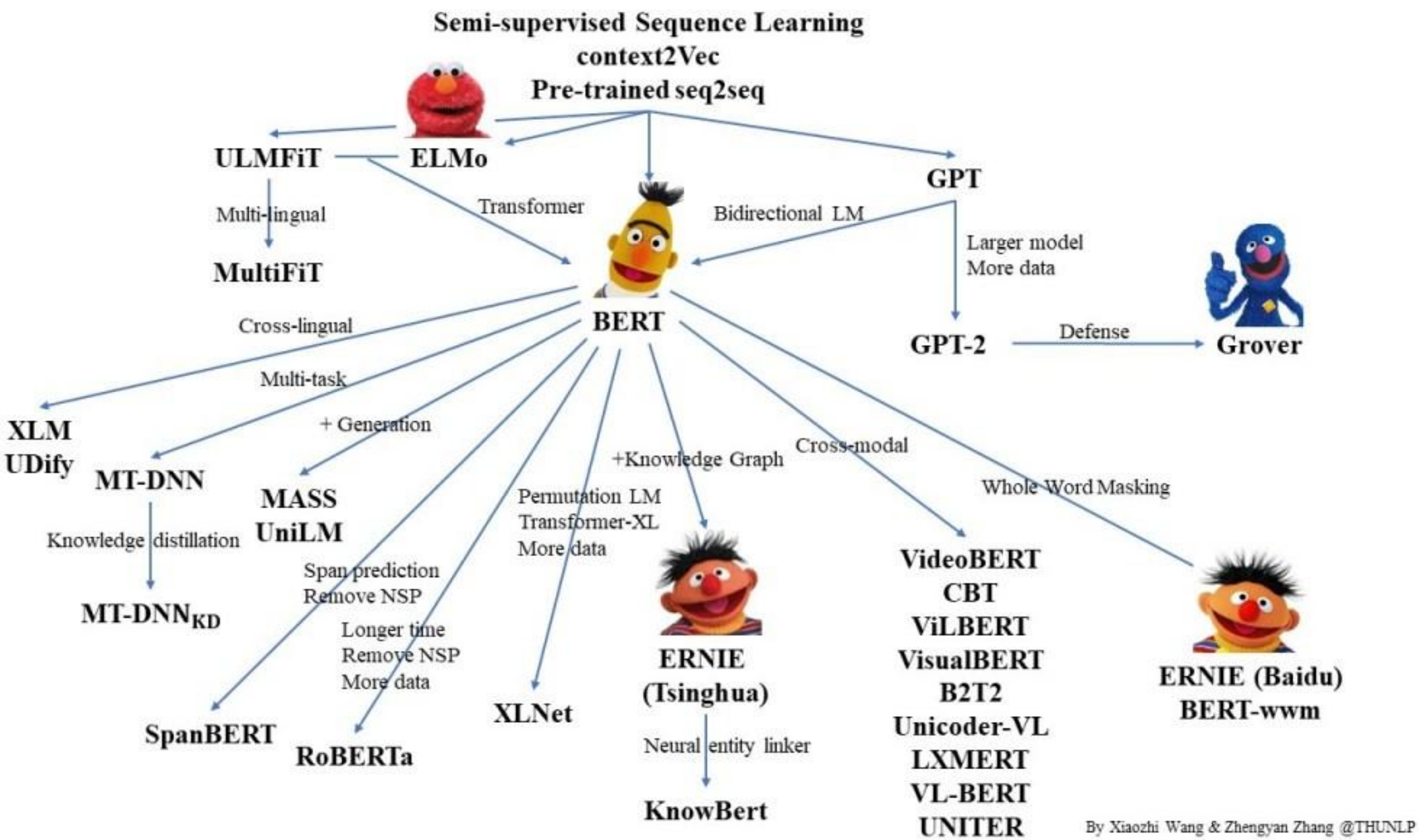


Pre-training

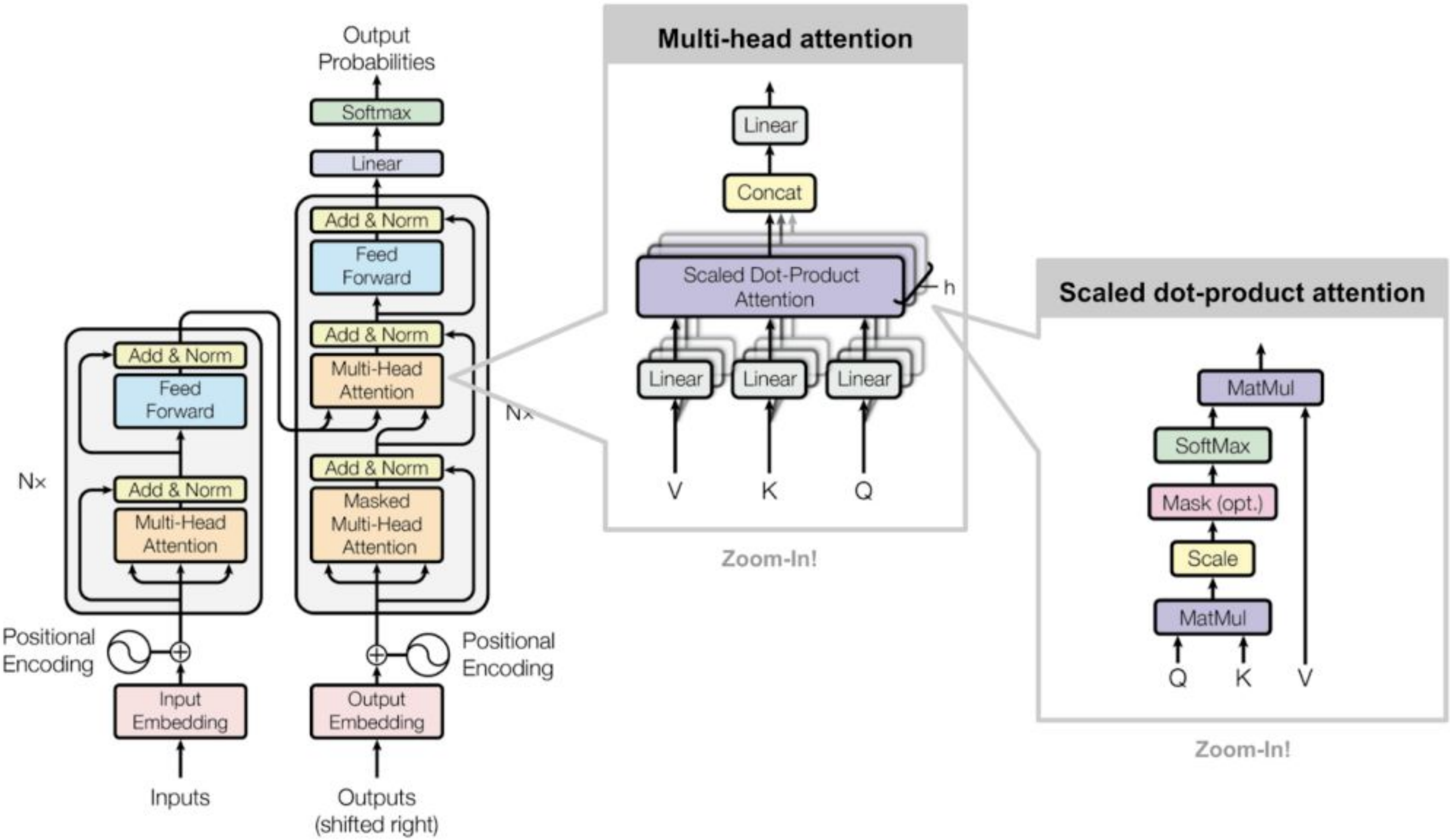
Fine-Tuning

Deep Learning Methods (Cont.)

- **Transformer**



BERT


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)
Multi-head attention
Linear
Concat
Scaled Dot-Product
Attention
h
Linear
Linear
Linear
V
K
Q
Zoom-In!
Scaled dot-product attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Zoom-In!

Industrial Package @ Repository

Spacy : https://spacy.io/
HayStack : https://github.com/deepset-ai/haystack
H2O :
https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html

RAPIDS : https://rapids.ai/

Simple Transformer : https://simpletransformers.ai/
Transformer : https://huggingface.co/
TensorFlow pre-trained model : https://tfhub.dev/

Machine Learning Steps

- **Pre Processing Steps (IMPORTANT STEP):**

Image processing :

- ❑ Channel Selection (RGB, HSV , YIQ , ….)
- ❑ Noise Removal & Image Enhancement
- ❑ Image Augmentation (for Deep Learning solutions)

Text Mining :

Language Detection

Gibberish detection (No- sense phrase, un-related topics ,…)

Expanding Contractions

Remove special characters(Symbols, HTML tags ,...)

Text Augmentation (Deep Learning – <u>hot topic !! , GPT3 OpenAI)</u>

Machine Learning Steps (Cont.)

- **Pre Processing Steps (IMPORTANT STEP):**

  - Numerical Data :

    - Outlier Detection
    - Missing Values
    - Analysis of Relevance and Redundancy (Efficient Feature Selection)
    - Categorical conversion
    - Convert a Time Series
    - Rescale Your Data (Fit for activation functions)
    - Transform Your Data (skewed Gaussian, exponential distribution)

Machine Learning Accuracy

- **Improve Learning Precision (IMPORTANT):**
  - Improve Performance With Data.
    - More Data, Invent Data
    - Resampling Methods
  - Improve Performance With Algorithms.
    - Spot-Check Algorithms
    - Steal From Literature

Machine Learning Accuracy (Cont.)

- Improve Performance With Algorithm Tuning**.
  - Diagnostics.
    - Over-fitting , Under-fitting
  - Weight Initialization (Fine- tune; Transfer Learning).
  - Learning Rate.
  - Activation Functions.
  - Batches and Epochs, Early Stopping.
  - Regularization, Optimization and Loss.

- Improve Performance With Ensembles.

Machine Learning Settings

- **Efficient Software Settings:**
  - Python as Programming Language and Anaconda as data science library
  - Linux Ubuntu as an efficient OS platform
- **Machine Learning Packages :**
  - Sklearn , skimage (conventional machine learning)
  - Tensorflow (Mostly, in this course !! Building ML models involves this apporach)
  - Keras, PyTorch , MxNet
- **Efficient Hardware Settings:**
  - Training with GPU support (cuDNN, NVIDIA CUDA)
  - Speed up in certain floating point operations (AVX instruction)
  - Using Bazel a build tool to install TensorFlow package dependencies

Machine Learning Deployment

- **Deploy Models; Production & @Scale : (IMPORTANT)**
  - Docker to Build Source code :
    - Docker image, Docker containers
      - Building Dockerfile, Requirement.txt (OS and code dependencies)
    - Pushing Docker Image to a private repository
  - User interface:
    - Up and running Tensorflow serving (with REST, gRPC)
    - Creating Web APIs (API Gateway, RESTful API with Flask)
  - Multiple concurrent request :
    - Building hosting services with uWSGI application
      - Nginx to handle actual client requests and proxy them to the uWSGI server.
      - Gunicorn (Python WSGI server that runs Python web application)

Machine Learning Measurement

- **Speed Performance**
  - Wall time , CPU time
- **Accuracy performance**
  - BLEU (Bilingual Evaluation Understudy)
  - Confusion Matrix Measures :
    - True Positive ,False Positive
    - False Negative ,True Negative
    - Precision , Recall
    - F-Score
    - Kappa
    - Label ranking average precision (LRAP)

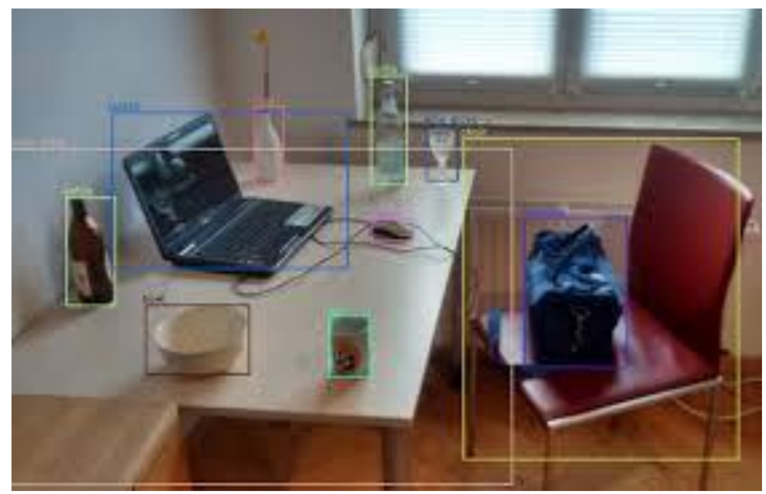
	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

AI in the Real world (Cont.)

- **Numerical Systems; Big Data and Machine Learning**
  - Banknote Authentication and Forgery Detection
  - Financial Fraud Detection
  - Bank Embezzlement & Money Laundering
  - Boost e-commerce Sales
  - Losing From Disgruntled Customers
  - Loan Approval Prediction

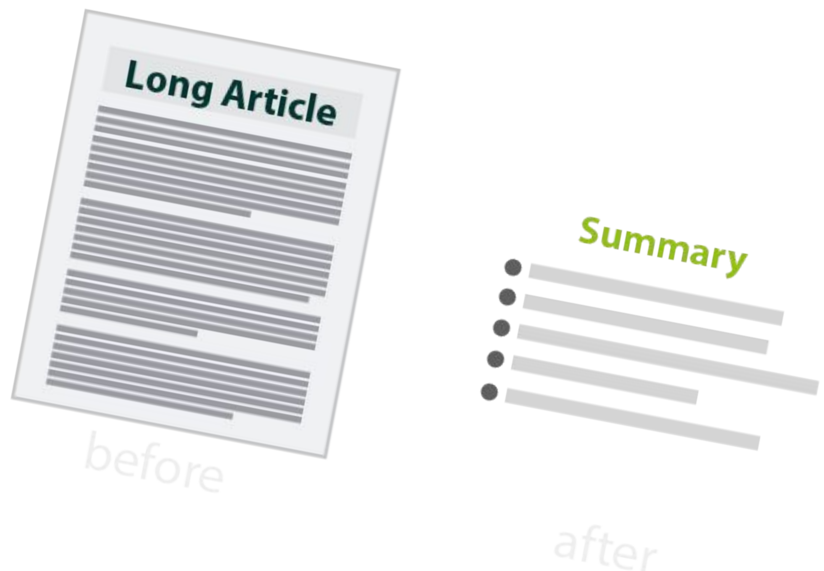
AI in the Real world (Cont.)

- **Image Processing; Big Data and Machine Learning**
  - Image Classification
  - Object Detection
  - Face Detection and Recognition
  - Video Summarization
  - Key Frame Detection in Human Action Videos
  - Image Super-Resolution

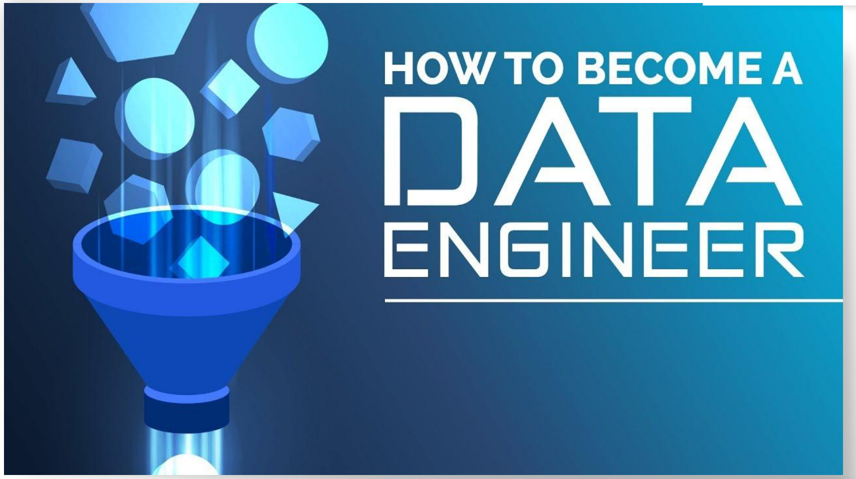


AI in the Real world (Cont.)

- **Text Understanding; Big Data and Machine Learning**
  - Key Word Extraction
  - Polarity Analysis
  - Text classification (Multi classes , Multi Labels)
  - Multi Documents Summarization
  - Topic modeling
  - Named Entity Recognition

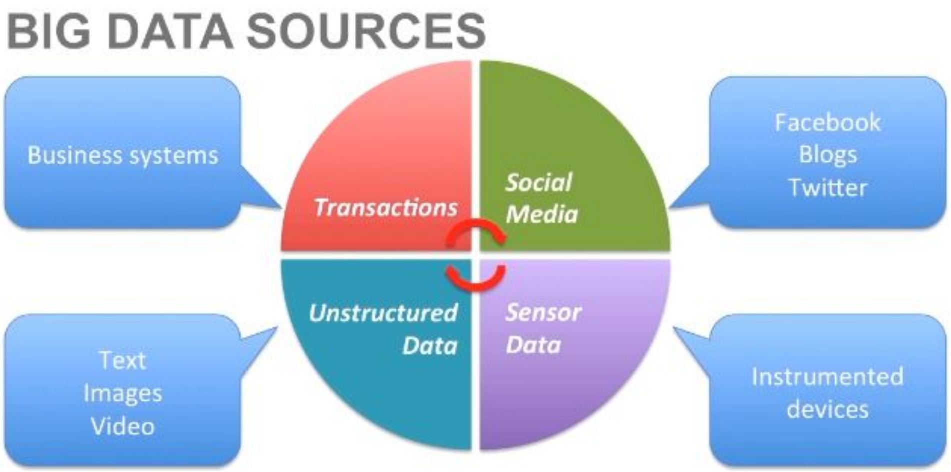


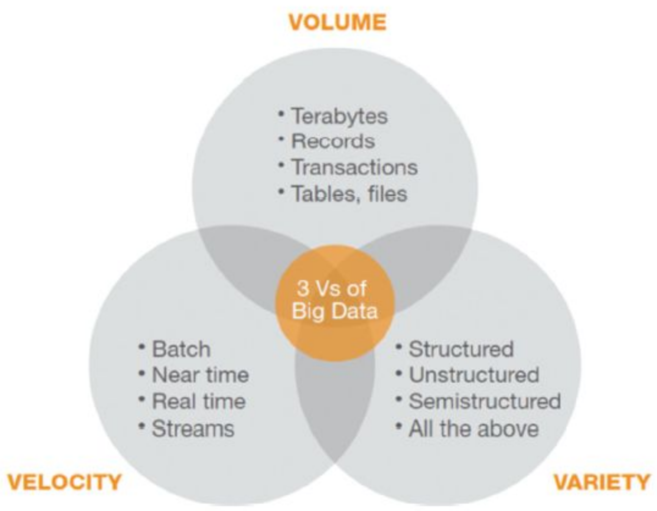
Machine Learning and Big Data



Big Data and Challenges

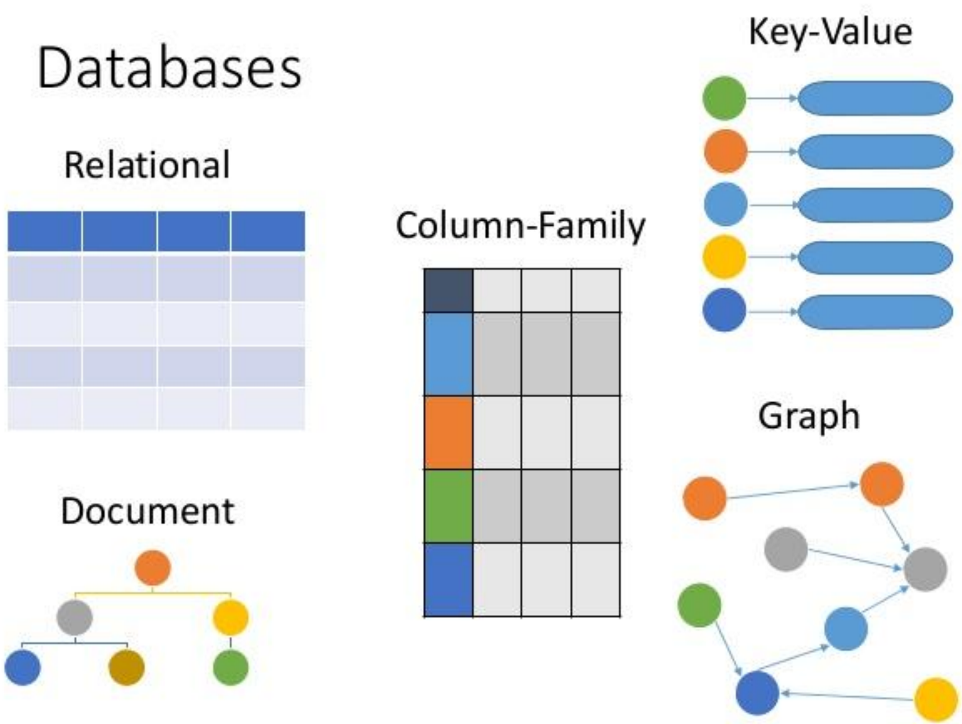
- **Sources and Massive Information**
  - Characteristics and Trends
    - The year 2015 was a big jump in the world of big data.
      - » Adoption of technologies, associated with unstructured data
      - » Ref : http://www.tableau.com/top-8-trends-big-data-2021?





Big Data and NoSQL

- **NoSQL particularly in response to <u>three challenges:</u>**
- Data Volume
- Data Velocity
- Data Variety



Big Data and NoSQL (out of scope!)

- **NoSQL Categories and types :**
  - More info : http://nosql-database.org/
  - List of NoSQL Databases [currently (Nov 2021) >225]
    - Open Source policies leads variety of novelties and names

Type	Example	
Key-Value Store	redis	riak
Wide Column Store	HBASE	cassandra
Document Store	mongoDB	CouchDB relax
Graph Store	Neo4j	InfiniteGraph The Distributed Graph Database

NoSQL Categories (out of scope!)

- **Key-Value Database / Key–Value store**
  - Unique key ; Varying items of data (Tuple, Dictionary)
    - Amazon Aurora , Redis,
- **Wide Column Database/ Column Families store**
  - Very large amounts of data distributed over *Many* machines.
    - Cassandra, Hbase, Scylla, Elassandra

NoSQL Categories (out of scope!)

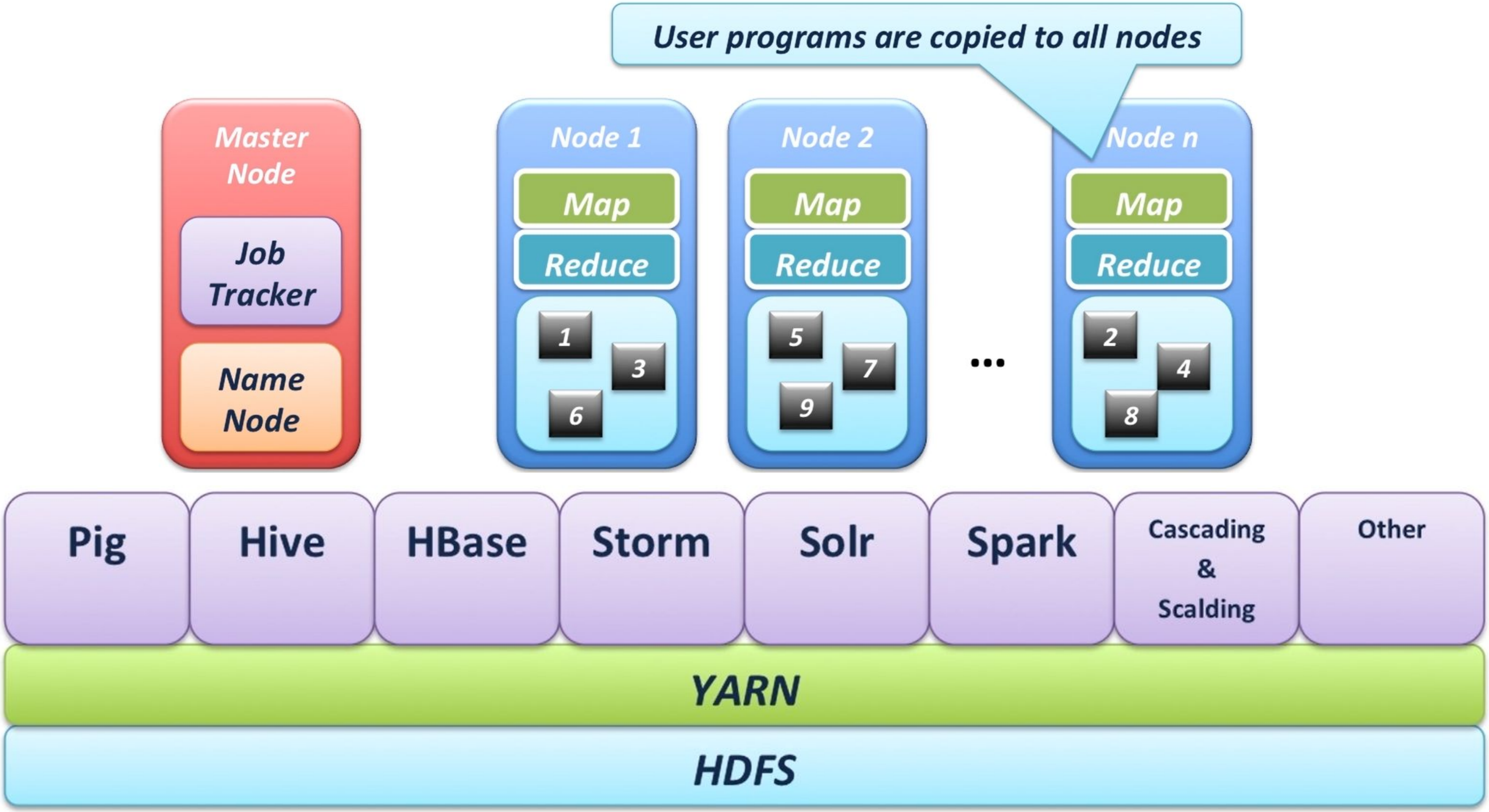
- **Document Databases**
  - Similar to key-value stores,
  - Semi-structured documents are stored in formats like JSON
  - Allowing nested values associated with each key.
  - Document databases support querying more efficiently.
    - CouchDB, MongoDB, ArangoDB

NoSQL Categories (out of scope!)

- **Graph Database**
  - Flexible graph model
    - Instead of tables of rows and columns and the rigid structure of SQL
    - Scale across multiple machines (Scale Out)
    - neo4j, InfoGrid, Infinite Graph, TITAN

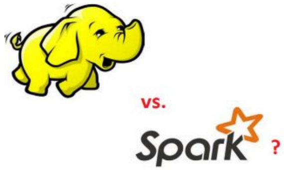
Hadoop Framework

- **Hadoop :**



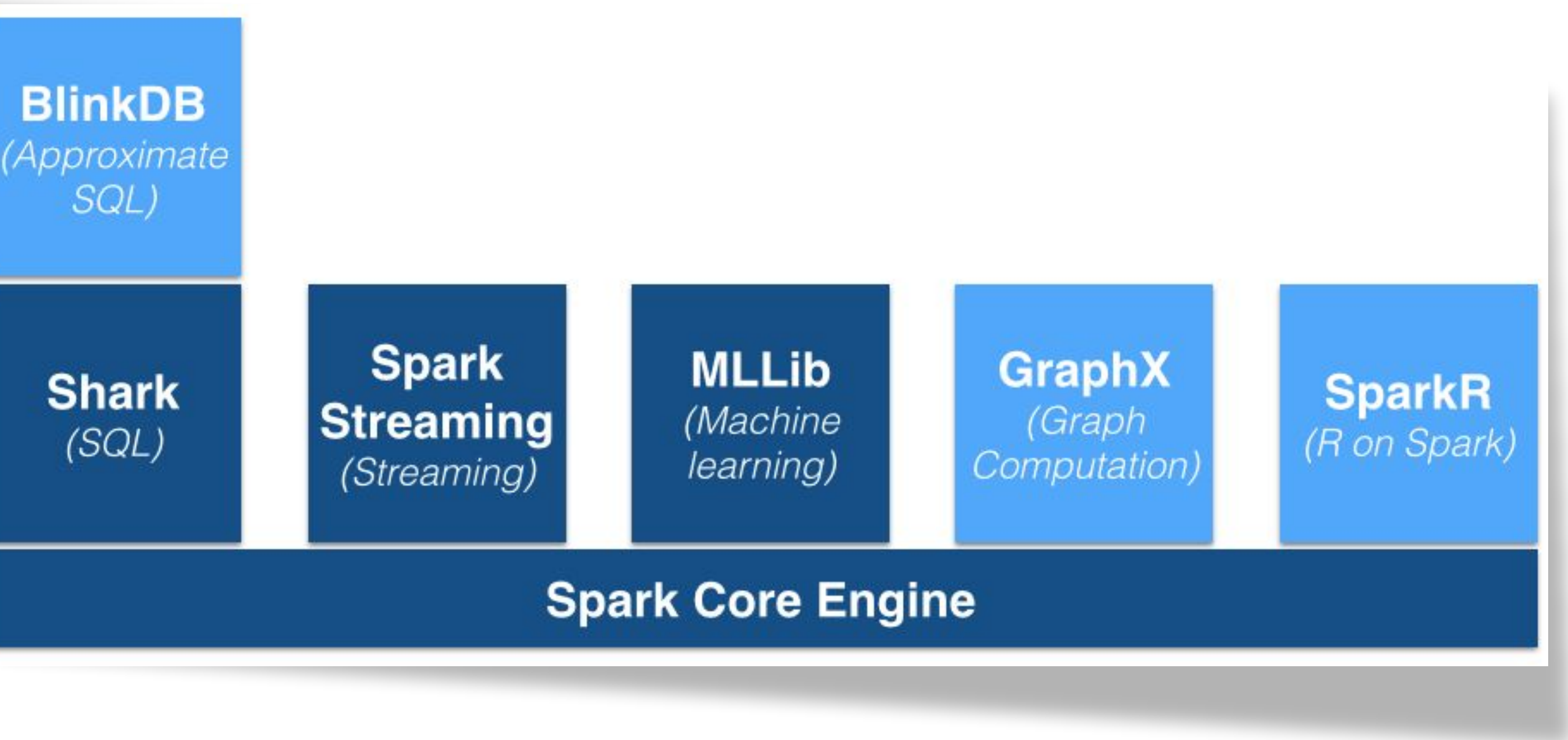
Apache Spark <IMPORTANT>

- **Spark Features (More than Distributed Processing)**
  - Ease of use, and sophisticated analytics
  - In-memory data storage and ALMOST real-time processing
  - Holds intermediate results in memory
  - Store as much as data in memory and then goes to disk
- **Spark & Hadoop**



  - On top of existing HDFS
  - Data sets that are diverse in nature (Text, Videos, …)
  - Variety in source of data (Batch v. real-time streaming data).
  - 100 times faster in memory, 10 times faster when running on disk.

Apache Spark Framework (Cont.)



Apache Spark Framework (Cont.)

- **Compatible with Java, Scala and Python**
- **Perform Data Analytics and Machine Learning**
  - SQL Queries, Streaming Data
  - Machine Learning and Graph Data Processing
    - Spark MLlib, Spark's Machine Learning library
- **On Demand :**
  - Apach Spark and Deep learning (On going Topics!)
  - EMR Service In Cloud !!! *(VERY IMPORTANT)*
    - Elastic Map Reduce
      - Hadoop and Spark in AWS

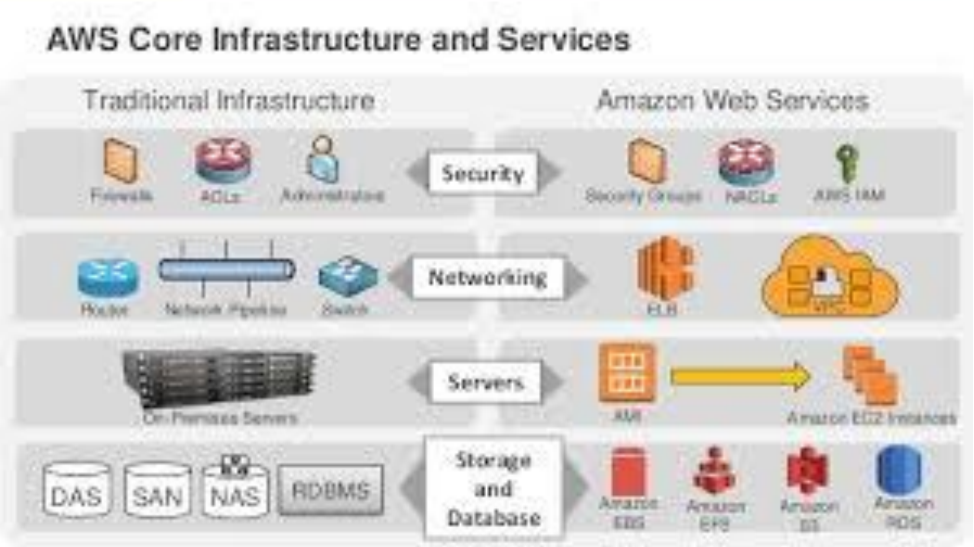
Lightning-Fast Cluster Computing

Big Data and Cloud

- **Cloud Computing Platform & Services**

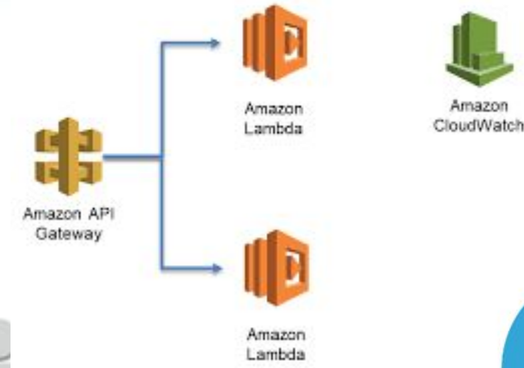
Big Data and Cloud (Cont.)

Big Data and Cloud (Cont.)



Amazon Web Services (AWS)

git + AWS CodeCommit



amazon S3

Cloud9

docker

Amazon SageMaker

AWS Fargate

amazon web services™

Applied Machine Learning

In general, steps in actions would be but not limited to :

1- Search topic with annotated and related topics and words in Google search engine

2- Filter out search into 4 main practical pages (Below Link) with appropriate phrase

3- Review (not details) related article in Google scholar and/or open- access Arxiv page

4- Borrow sample code(s) from GitHub or mostly from 4 main pages (Below Links)

5- Hyper parameter and Fine-tuning for at least two potential solutions (Keep looking and seeking for pre-trained model in near business problem in different resources like TensorFlow Hub , HuggingFace , ...)

Applied Machine Learning

6- Having first intuitive primary version as proof of concept (PoC)

7- Code modification to be in professional format (Class- Function , convenient naming, ...)

8- Code optimization (Appropriate data structure like json, hash data, trained model in Protocol buffer, I/O throughput a large scale "tf.data.Dataset" tf.keras.utils.image_dataset_from_directory, tf.io.gfile.glob,..., install and build packages from source,...)

9 - Deploy model at scale in production (Docker - Flask /FastAPI - Gunicorn - ALB - ELB - TFX - Cloud Computing " GCP . AWS. Azure")

10 - Quality Analysis

Data Science and Applications - Resources :

https://towardsdatascience.com/

https://www.analyticsvidhya.com/

https://medium.com/

https://machinelearningmastery.com/

Contact Info

- Mehdi Zadeh (PhD in Computer Science)
  - Telephone and What'sApp :
    - +1 438 368 3132
  - Email
    - zadeh1980mehdi@gmail.com

LinkedIn :

https://www.linkedin.com/in/mehdihabibzadeh/

Instagram:

https://www.instagram.com/nimahm1980/