Enterprise AI - An Application Perspective

Version One – Oct 2019

By

Ajit Jaokar and

Cheuk Ting Ho

Available exclusively on Data Science Central with free access

https://www.datasciencecentral.com/profiles/blogs/free-ebook-enterprise-ai-an-applications-perspective

The book is used as a reference for Ajit and Cheuk's new course on Implementing Enterprise AI

About the authors

Based in London, Ajit's work spans research, entrepreneurship and academia relating to Artificial Intelligence (AI) and the Internet of Things (IoT). Ajit works as a Data Scientist (Bioinformatics and IoT domains). He is the course director at Oxford University for the “Data Science for Internet of Things” course. Besides Oxford University, Ajit has also conducted AI courses in LSE, UPM and is part of the Harvard Kennedy Future society research on AI.
https://www.linkedin.com/in/ajitjaokar/

Cheuk Ting Ho is a Data Scientist in Machine Learning and Deep Learning. She regularly contributes to the Data Science community by being a public speaker, encouraging women in technology, and actively participates to Python open source projects.
https://www.linkedin.com/in/cheukting-ho/

Contents

Introduction 4

Machine Learning, Deep Learning and AI 4

The Data Science Process 7

Categories of Machine Learning algorithms 11

How to learn rules from Data? 14

An introduction to Deep Learning 15

What problem does Deep Learning address? 17

How Deep Learning Drives Artificial Intelligence 18

Deep Learning and neural networks 20

Perceptrons – an artificial neuron 20

MLP - How do Neural networks make a prediction? 22

Spatial considerations - Convolutional Neural Networks 25

Temporal considerations - RNN/LSTM 31

The significance of Deep Learning 32

Deep Learning provides better representation for understanding the world 33

Deep Learning a Universal Function Approximator 34

What functionality does AI enable for the Enterprise? 35

Technical capabilities of AI 35

Functionality enabled by AI in the Enterprise value chain 36

Enterprise AI applications 39

Creating a business case for Enterprise AI 39

Four Quadrants of the Enterprise AI business case 41

Types of AI problems 46

Enterprise AI – Deployment considerations 48

A methodology to deploy AI applications in the Enterprise 48

DevOps and the CI/CD philosophy 53

Conclusions 56

Introduction

Enterprise AI: An applications perspective takes a use case driven approach to understand the deployment of AI in the Enterprise. Designed for strategists and developers, the book provides a practical and straightforward roadmap based on application use cases for AI in Enterprises. The authors (Ajit Jaokar and Cheuk Ting Ho) are data scientists and AI researchers who have deployed AI applications for Enterprise domains. The book is used as a reference for Ajit and Cheuk's new course on Implementing Enterprise AI

After reading this book, the reader should be able to

- Understand Enterprise AI use cases
- De-mystify Enterprise AI
- Understand what problems Enterprise AI solves (and does not solve)

The term 'Enterprise' can be understood in terms of Enterprise workflows. These are already familiar through deployments of ERP systems like SAP or Oracle Financials. We consider both core Enterprise and the Wider enterprise workflows (including supply chain). The book is thus concerned with understanding the touchpoints which depict how Enterprise workflows could evolve when AI is deployed in the Enterprise.

Machine Learning, Deep Learning and AI

In this book, we will use the following definitions:

- **Machine learning** is the act of Computers learning from experience (typically by using past data).
- **Deep learning** is a set of techniques that involve automatic feature detection (explained below).
- **Artificial intelligence** uses many Deep Learning techniques to create machines that can reason.

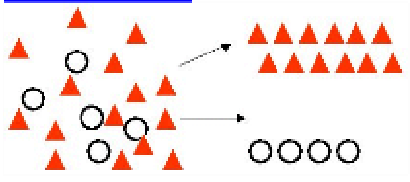
The process of model building involves machine learning and deep learning. Tom Mitchell defines **Machine Learning** as: “*The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience*.”. Formally, this definition is expressed as:
“A computer program is said to learn from experience (E) with respect to some class of tasks (T), and performance
measure (P), if its performance at tasks in T, as measured by P, improves
with experience (E).”

Machine Learning is used in a range of applications such as Spam Detection, Credit Card Fraud Detection, Speech recognition, Face Detection, Medical Diagnosis and Customer Segmentation

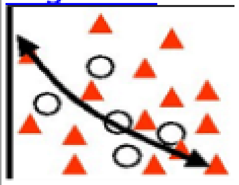
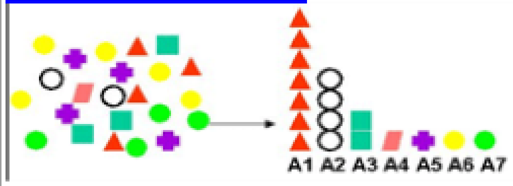
Types of Problems addressed by machine learning include:

- **Classification:** Data is assigned to a class - for example spam/no-spam or fraud/no-fraud etc.
- **Regression:** A value of data is predicted – for example predicting stock prices, house prices etc
- **Clustering:** Data is not labelled but can be divided into groups based on similarity and other measures of structure inherent in the data.
- **Rule Extraction:** Involves the extraction of rules in the form of IF-THEN, i.e. antecedent/consequent rules.

We could visually represent these as below:

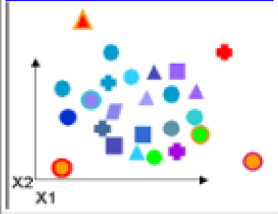
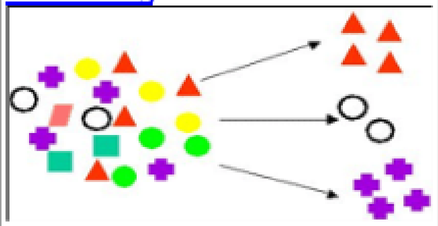
Technique	Applicability	Algorithms
Classification 	Most commonly used technique for predicting a specific outcome such as response / no-response, high / medium / low-value customer, likely to buy / not buy.	Logistic Regression — classic statistical technique but now available inside the Oracle Database and supports text and transactional data Naive Bayes — Fast, simple, commonly applicable Support Vector Machine— Next generation, supports text and wide data Decision Tree — Popular, provides human-readable rules

Source: Oracle

Regression 	Technique for predicting a continuous numerical outcome such as customer lifetime value, house value, process yield rates.	Multiple Regression — classic statistical technique but now available inside the Oracle Database and supports text and transactional data Support Vector Machine — Next generation, supports text and wide data
Attribute Importance 	Ranks attributes according to strength of relationship with target attribute. Use cases include finding factors most associated with customers who respond to an offer, factors most associated with healthy patients.	Minimum Description Length— Considers each attribute as a simple predictive model of the target class

Source: Oracle

re

Anomaly Detection 	Identifies unusual or suspicious cases based on deviation from the norm. Common examples include health care fraud, expense report fraud, and tax compliance.	One-Class Support Vector Machine — Trains on "normal" cases to flag unusual cases
Clustering 	Useful for exploring data and finding natural groupings. Members of a cluster are more like each other than they are like members of a different cluster. Common examples include finding new customer segments, and life sciences discovery.	Enhanced K-Means— Supports text mining, hierarchical clustering, distance based Orthogonal Partitioning Clustering— Hierarchical clustering, density based Expectation Maximization— Clustering technique that performs well in mixed data (dense and sparse) data mining problems.

The Data Science Process

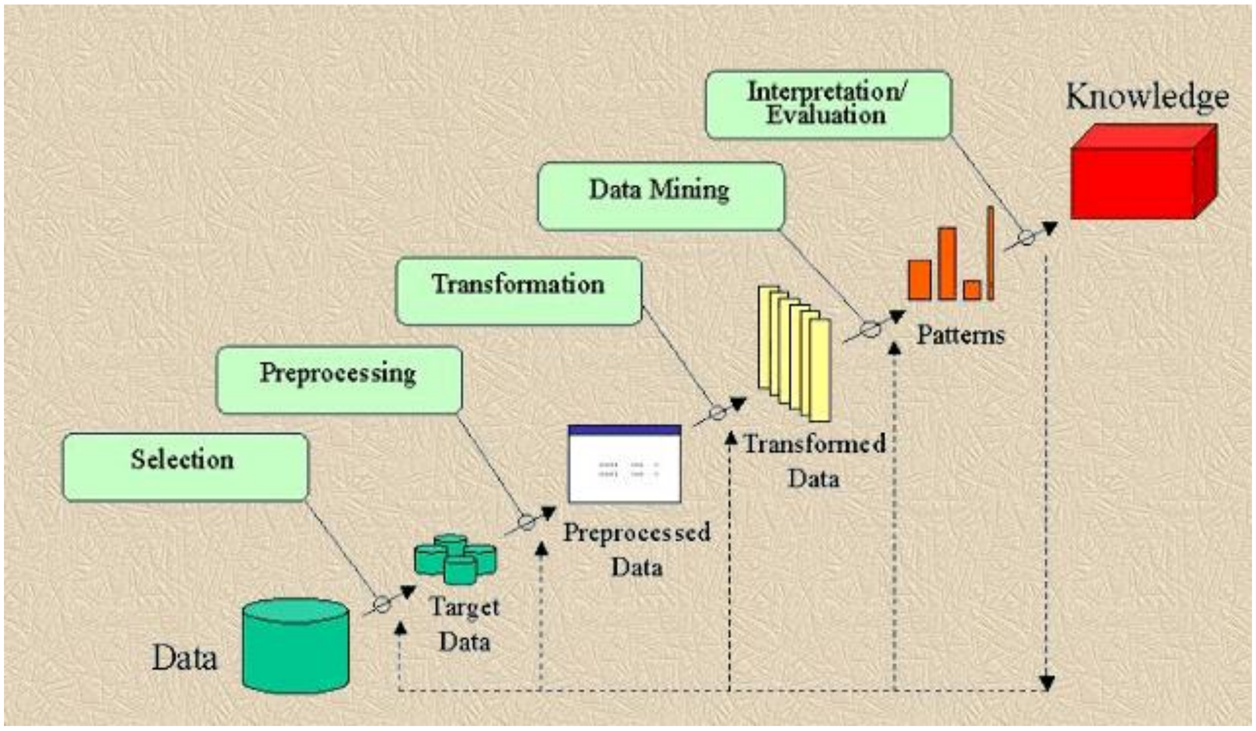
What exactly does a Data Scientist Do?

Let us first try and understand the functions a Data Scientist performs.

Data science is a cross-functional domain spanning Statistics, Computer Science and specific domain knowledge (for example knowledge of application areas like Finance, Healthcare where data science is applied). Over time, Data engineering and DevOps[1] has also played an essential part in the deployment of data science. A data scientist is involved in the

[1] https://aws.amazon.com/devops/what-is-devops/

programmatic/algorithmic aspects, i.e. **a data scientist is concerned primarily with building models from data**. The model is used to gain new insights on unseen data. The roots of Data Science can be traced to early initiatives like Knowledge Discovery in Databases(KDD). KDD involves finding knowledge in data and emphasises the high-level application of data mining methods. KDD incorporates a range of techniques like statistics, pattern recognition, databases and machine learning. The unifying goal of the KDD process is to extract knowledge from data in the context of large databases. The KDD process can be visualised as below [2]



These steps are similar to a typical Data Science process being used today. With the development of the underlying infrastructure, their implementation has become scalable.

[2] http://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1_kdd.html

The process of building a model involves seven steps [3]

1. **Data exploration and hypothesis formulation**: Before creating a hypothesis, we gain a better understanding of the Data using Exploratory Data Analysis (EDA) techniques - explained in later sections. From the EDA analysis, we establish what your model should predict and how.

2. **Load and transform relevant data**: We then collect the data corresponding to your hypothesis and convert it to fit the model's framework including the technical frameworks ex: databases used. After this, the data is split into two groups: one for training or building the model and another for testing the accuracy of its predictions.

3. **Identify features**: Feature engineering is the process of using domain knowledge of the data to create features that help machine learning algorithms perform optimally. Feature engineering is a critical component of machine learning, and it is both challenging and expensive.[4]

4. **Build your model**: Next, we select one model (or try out multiple models) which suit the data. We use the training set and the test set to build the model.

5. **Evaluate your model**: We next validate the accuracy of the model using techniques like cross-validation and receiver operating characteristic (ROC) curve analysis to verify that the model will generalise well to brand new data.

6. **Deploy your model**: Once the model is ready, it is implemented into production. Typically, in a complex environment, the stage of deploying the model into production is handled by the DevOps engineers.

7. **Monitor your model**: Ongoing tuning, retraining and evolving the model.

The implementation of data science models may involve **Big Data techniques**. The term ‘Big Data’ is defined in several ways and includes many sub-components. However, the central themes underlying Big Data are:

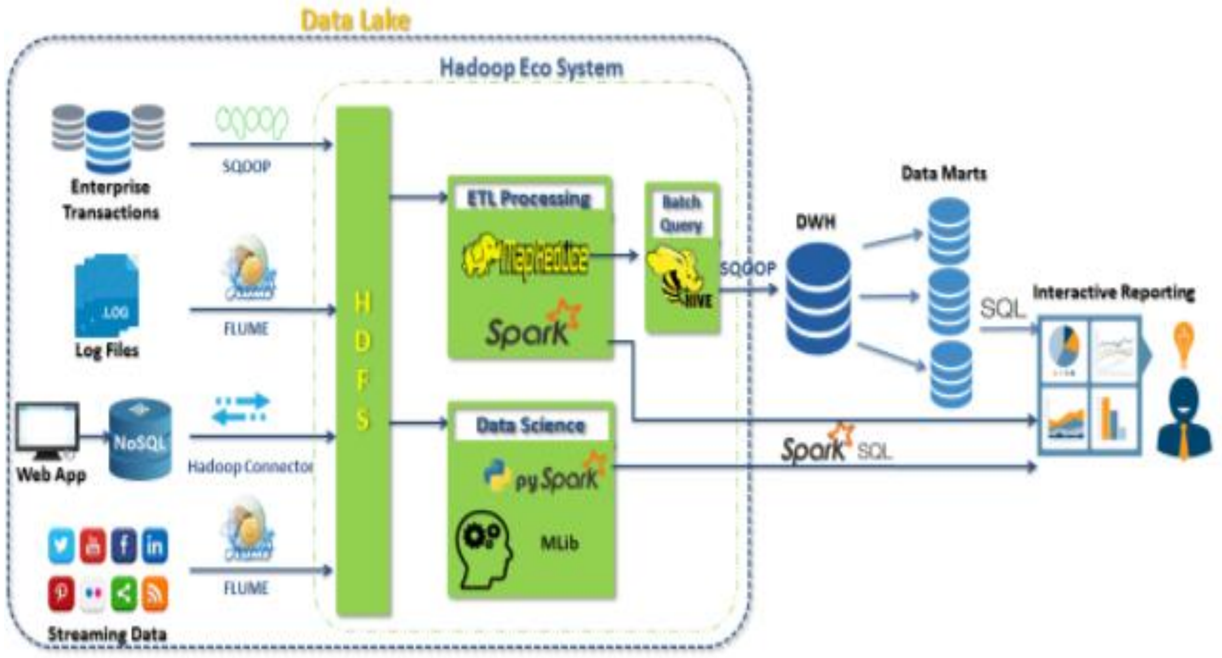
a) The Data is processed in a distributed manner, i.e. across multiple nodes and

[3]Adapted from https://www.datascience.com/blog/predictive-data-models-from-data-cleaning-to-model-deployment.

[4] https://en.wikipedia.org/wiki/Feature_engineering

b) Some variant of Hadoop[5] implementation underpins the deployment of Big Data

A typical flow of information in a Big Data architecture is as shown below. Considering our focus on analytics, we do not cover Big Data technologies in this book.



Source Data Science central[6]

In large enterprises, these seven steps often span three job roles, i.e. **the data engineer, the data scientist and the DevOps engineer.** In this book, we focus on the analytic capabilities as the primary function of a data scientist. In other words, the data scientist is mainly responsible for building the algorithms. She is assisted by the data engineer. Data engineers prepare data which is analysed by data scientists. Data engineers integrate data from a variety of sources using various big data tools and technologies. The role of the data engineer is similar to the ETL (Extract Transform Load)[7] function in Data Warehousing. In addition to the data engineer, the data scientist also engages with the DevOps engineer. The need for DevOps arises because today, software is not written monolithically. Due to the many open source components,

5 https://en.wikipedia.org/wiki/Apache_Hadoop
6 https://www.datasciencecentral.com/profiles/blogs/beyond-datawarehouse-the-data-lake
7 https://en.wikipedia.org/wiki/Extract,_transform,_load

software is now created by integrating other open source products through APIs (Application Programming Interface)[8] and deployed across a diverse set of ecosystems through virtualised mechanisms like Docker[9] and Kubernetes[10]. In real life, these roles may overlap especially for smaller organisations.

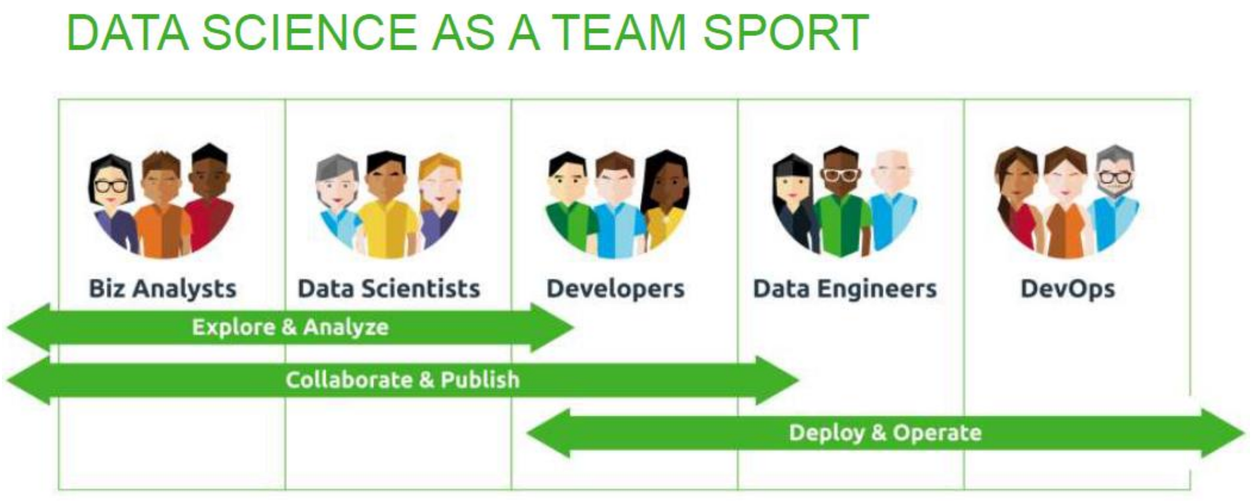


Image source Anaconda/Continuum analytics[11]

Categories of Machine Learning algorithms

Regarding how algorithms learn, we can classify machine learning algorithms into four classes Supervised learning, Unsupervised learning, Semi-supervised learning (ex: GANs) and Reinforcement learning. Note that Deep Learning is not a single category and combines supervised and unsupervised learning strategies. Firstly, we have two major classes of algorithms: Supervised learning and Unsupervised learning. Supervised learning algorithms are trained on labelled datasets. Unsupervised machine learning algorithms do not use labelled

[8] https://en.wikipedia.org/wiki/Application_programming_interface
[9] https://en.wikipedia.org/wiki/Docker_(software)
[10] https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
[11] https://www.slideshare.net/continuumio/the-five-dysfunctions-of-a-data-science-team

data. Instead, the algorithm must determine feature importance based on patterns inherent in the data itself.

In **supervised learning**, the labelled data represents the ‘ground truth,' i.e. the prior knowledge on which the algorithm is trained. The goal of supervised learning is to learn a functional mapping between the inputs and the outputs. Supervised learning algorithms are typically employed in tasks involving classification. They include algorithms like logistic regression, support vector machines and random forests.

In the case of **unsupervised learning**, we wish to learn the inherent structure of our data without using explicitly-provided labels. Unsupervised learning includes activities like clustering, representation learning, and density estimation. It includes techniques like k-means clustering, principal component analysis, and autoencoders. Unsupervised learning is typically used in exploratory analysis and dimensionality reduction.

Deep Learning techniques can be seen to combine unsupervised and supervised techniques.
Deep Learning aims to model complex, hierarchical features in data through a hierarchy of representations.

Semi-supervised learning – Most deep learning algorithms need a large amount of labelled data. To overcome this limitation, a technique called semi-supervised learning is used. Semi-supervised learning uses a small amount of labelled data along with a large amount of unlabeled data. This approach gives a better performance. GANs (Generative Adversarial Networks) is an example of this technique.
In the following section, we will describe GANs and Reinforcement learning in more detail.

<u>Generative Adversarial Networks(GANs)</u>

GANs are gaining in popularity recently. GANs belong to a set of algorithms called generative models. Typically, GANs are used in cases where we generate realistic data from a given sample – typically image data.

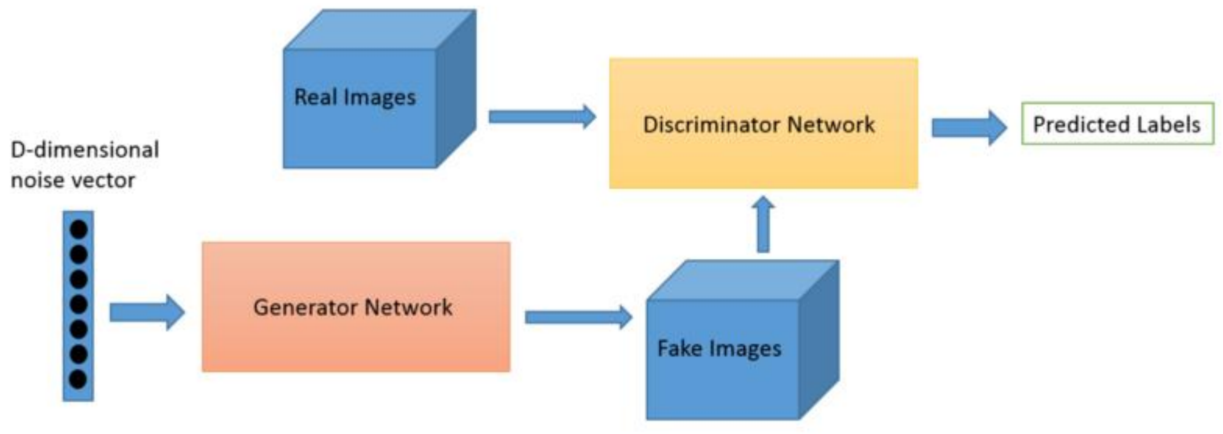


Image source [12]

Generative Adversarial Networks are composed of two models: The Generator and the Discriminator. The analogy of a forger and an art expert is often used to illustrate how GANs work. The Generator acts like a human art forger which creates fake works of art. The discriminator serves as the art expert who tries to detect these counterfeit works. If the input data belongs to the original dataset it is deemed as 'real' or else, it is generated (and hence fake)

Reinforcement learning

Reinforcement learning is a type of Machine Learning where an agent learns how to behave in an environment by performing actions and seeing the results.

With the impressive strides from DeepMind (ex in AlphaGo), Reinforcement learning has been getting much traction. Reinforcement Learning involves the use of an agent that learns from

[12] https://towardsdatascience.com/generative-adversarial-networks-gans-a-beginners-guide-5b38eceece24

the environment by interacting with it. The agent receives rewards for its actions, and the goal is to maximise the cumulative reward. This RL loop outputs a sequence of **state, action and reward** with the intent to maximise the expected cumulative reward. In doing so, the agent makes better decisions with each iteration.

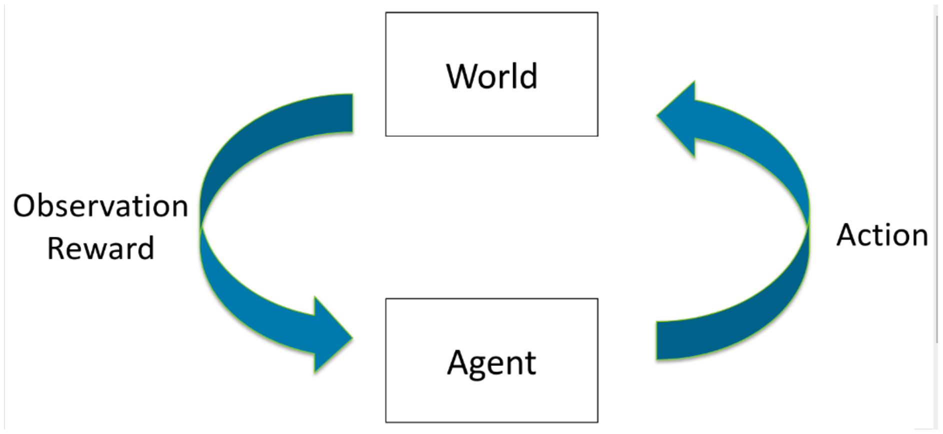


Image source [i]

RL differs from supervised learning because, in addition to the cost function, we also have the reward function to optimise.

How to learn rules from Data?

The process of Machine Learning involves learning rules from Data.

Every machine Learning algorithm has three components: **representation, evaluation and optimisation.**[13] The process of representation involves setting up the data in such a way that it can be used by the system to build models. Once the system sees the data, the next step is the evaluation. In the evaluation stage, we use the data to learn the problem. Evaluation is

[13] https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

followed by optimisation where we try to minimise or maximise the evaluation function (typically a loss function).

If we explore further the idea of machine learning, there are two ways to train an algorithm: We could take a **top-down approach**, i.e. start with the Rules and apply them to Data OR we could undertake a **bottom-up approach,** i.e. start with the data and find the rules from the Data. In **a rule-based approach** (top-down), we could manually write rules for all possible scenarios. However, this approach is limited by the number of rules we can define. The bottom-up approach is more complicated than the top-down approach. Potentially, in the data, there is no model (schema), no linearity(sequence), no hierarchy. The output may be unknown (non-deterministic problems). The problem domain may not be finite.

Chess is a finite-domain problem because there are only 64 squares in chess and each piece has a defined action. In the game of chess, a computer can determine all the rules because it is a finite domain problem. In contrast, recognising an image of a dog from a picture is easy for a child but not for a computer. Identifying pictures of dogs is not a finite-domain problem since there are many types of dogs in many different configurations in images (ex: with a collar, with their tail cropped etc.)

Deep Learning techniques are used to solve the problems described above (i.e. which are not finite domain problems). Andrew Ng and his team used 10 million images from YouTube videos to train a Deep Learning algorithm to recognise pictures of Cats - without telling the computer features which make up the idea of a cat. The algorithm detects features that comprise a cat. The same technique can be applied to real problems like tumour detection. Deep learning techniques are used in Siri(Apple), face detection and other related ideas.

An introduction to Deep Learning

A machine learning algorithm is an algorithm that can learn from data. The definition of 'learning' in this context is provided by Mitchell, i.e. learning from experience based on Data. To learn about the problem, first, we need to understand the characteristics of the problem. Hence, we capture pieces of information about the process which function as data inputs for the algorithm. These are known as **features**. Selecting good features is critical for the performance of AI algorithms. However, for many tasks (especially non-finite domain tasks) it is difficult to know what features to consider. One solution is: instead of hand designing the features, we use machine learning techniques to identify relevant features. This strategy is called **Representation learning**, and it is one of the primary foundations of deep learning, i.e. the ability to detect features automatically from a dataset without human intervention. Representation learning gives deep learning (and AI) the power to be disruptive. Not only can we learn representations through deep learning, but we can also learn complex representations through a hierarchy of more straightforward representations. Representation learning is a compelling idea, and it underpins both deep learning and AI. The best-known examples of a deep learning network is the multi-layer perceptron(MLP)[14](explained in later sections). The MLP comprises of multiple layers of neurons that use a supervised algorithm called **backpropagation** for training. We can think of MLP as a mathematical function mapping some set of inputs to some set of outputs. However, this single function can be seen as a hierarchy comprising of multiple sub-functions. Each layer can be seen as implementing a sub-function, and the subsequent layers build upon the outputs of the previous input. The technique is called deep learning because of the presence of multiple layers. We specify the inputs and the outputs, but the architectures of the interim layers are not defined. The machine learns the structure of the intermediate layers as part of the training process.

Hence, we can see deep learning as a specific kind of machine learning that achieves impressive results by learning the representation of concepts through a hierarchy of sub-concepts. In other words, the concept is supervised only at the higher level (ex: cat or dog), but the sub-concepts and their hierarchy are learned on their own. For example, a cat has whiskers, fur etc.

[14] https://en.wikipedia.org/wiki/Multilayer_perceptron

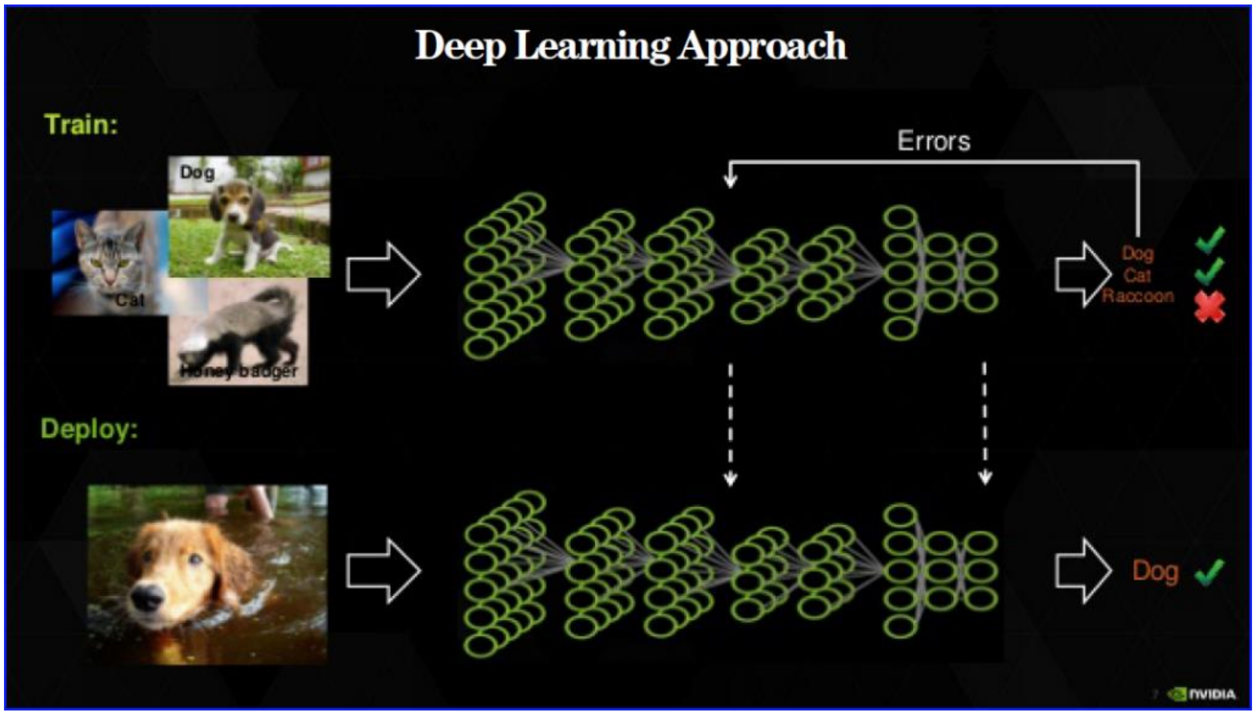


Image source: Nvidia

What problem does Deep Learning address?

Deep learning refers to artificial neural networks that are composed of many layers. The 'Deep' refers to multiple layers. In contrast, many other machine learning algorithms like SVM are shallow because they do not have a Deep architecture through multiple layers. The Deep architecture allows subsequent computations to build upon previous ones. We currently have deep learning networks with 10+ and even 100+ layers.

The presence of multiple layers allows the network to learn more abstract features. Thus, the higher layers of the network can learn more abstract features building on the inputs from the lower layers. A Deep Learning network can be seen as a Feature extraction layer with a Classification layer on top. The power of deep learning is not in its classification skills, but instead in its feature extraction skills. Feature extraction is automatic (without human intervention) and multi-layered.

The network is trained by exposing it to a large number of labelled examples. Errors are detected and the weights of the connections between the neurons adjusted to improve results. The optimisation process is repeated to create a tuned network. Once deployed, unlabeled images can be assessed based on the tuned network.

Feature engineering involves finding connections between variables and packaging them into a new single variable is called a feature. Deep Learning performs automated feature engineering. Automated feature engineering is the defining characteristic of Deep Learning especially for unstructured data such as images. This ability matters because the alternative is engineering features by hand. This is slow, cumbersome and depends on the domain knowledge of the people/person performing the Engineering. Deep Learning suits problems where the target function is complex, and datasets are significant but with examples of positive and negative cases. Deep Learning also suits problems that involve Hierarchy and Abstraction.

Abstraction is a conceptual process by which general rules and concepts are derived from the usage and classification of specific examples. We can think of an abstraction as the creation of a ‘super-category' which comprises of the common features that describe the examples for a specific purpose but ignores the ‘local changes' in each example. For example, the abstraction of a ‘Cat' would comprise fur, whiskers etc. For Deep Learning, each layer is involved with the detection of one characteristic, and subsequent layers build upon previous ones. Hence, Deep Learning is used in situations where the problem domain comprises abstract and hierarchical concepts. Image recognition falls in this category. In contrast, a Spam detection problem that can be modelled neatly as a spreadsheet probably is not a complex problem to warrant Deep Learning

How Deep Learning Drives Artificial Intelligence

Improvements in **Deep Learning algorithms** drive AI. Deep Learning algorithms can detect patterns without the prior definition of features or characteristics. They can be seen as a hybrid form of supervised learning because you must still train the network with a large number of examples but without the requirement for predefining the characteristics of the examples (features). Deep Learning networks have made vast improvements both due to the algorithms themselves but also due to better hardware (specifically GPUs)

The term **Artificial Intelligence (AI)** implies a machine that can Reason. A more complete list or AI characteristics (source David Kelnar) is

1. **Reasoning:** *the ability to solve problems through logical deduction*
2. **Knowledge:** *the ability to represent knowledge about the world* (the understanding that there are certain entities, events and situations in the world; those elements have properties; and those elements can be categorised.)
3. **Planning**: *the ability to set and achieve goal*s (there is a specific future state of the world that is desirable, and sequences of actions can be undertaken that will effect progress towards it)
4. **Communication:** *the ability to understand written and spoken language.*
5. **Perception:** *the ability to deduce things about the world from visual images, sounds and other sensory inputs*

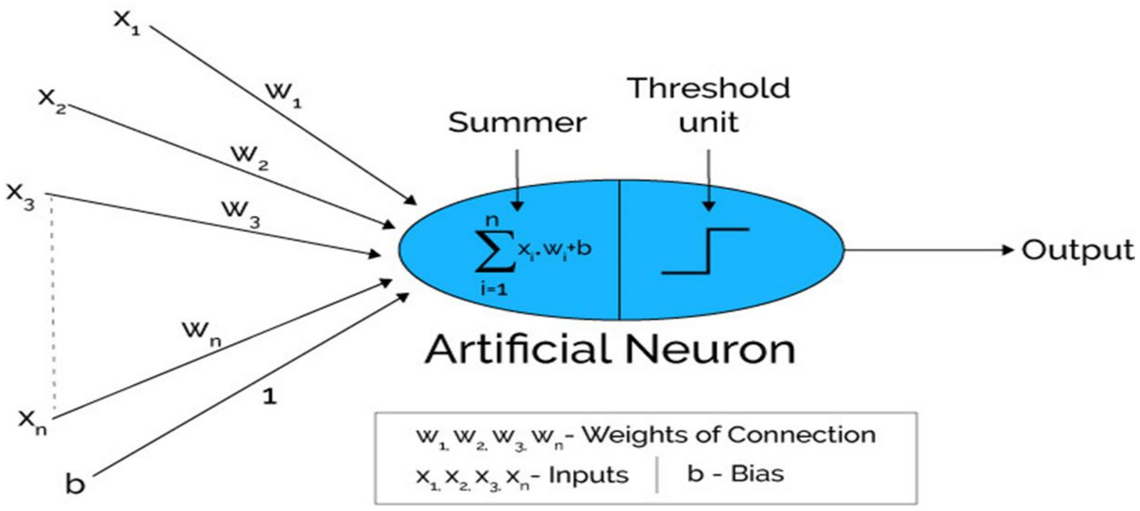
The holy grail of AI is artificial general intelligence (aka like Terminator!) that allows machines to function independently in a typical human environment. What we see today is mostly narrow AI (ex like the NEST thermostat). AI is evolving rapidly. A range of technologies drive AI currently. These include image recognition and auto labelling, facial recognition, text to speech, speech to text, auto translation, sentiment analysis, and emotion analytics in image, video, text, and speech. AI Apps have also reached accuracies of 99% in contrast to 95% just a few years back.

However, the overall trend of better Deep Learning will lead to richer AI in all areas including the Enterprise.

Deep Learning and neural networks

Perceptrons – an artificial neuron

Note: This section is relatively complex, but it's worth understanding the basics of neural networks if you want to understand AI.



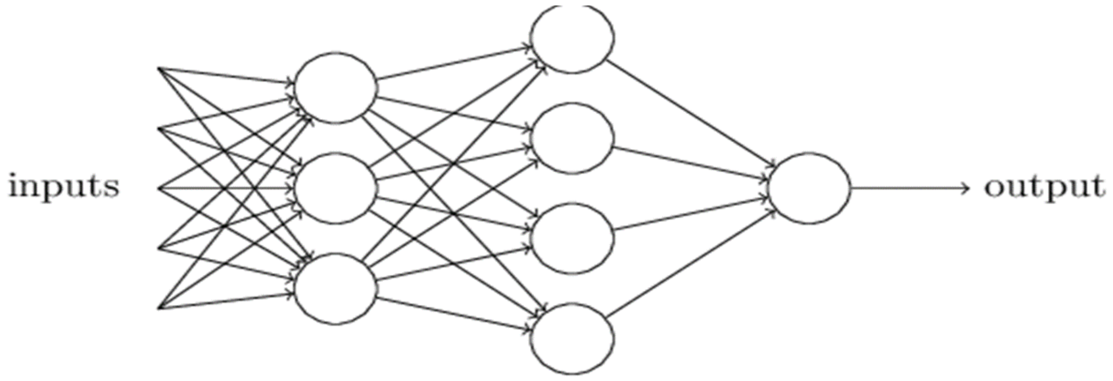
x_1
x_2
x_3
x_n
b
w_1
w_2
w_3
w_n
1
Summer
Threshold unit
$\sum_{i=1}^{n} x_i \cdot w_i + b$
Output
Artificial Neuron
w_1, w_2, w_3, w_n - Weights of Connection
x_1, x_2, x_3, x_n - Inputs
b - Bias

Image source: Xenon stack [15]

Neural networks are modelled on the working of the Brain by emulating the function of Neurons. A **perceptron** is modelled as an artificial neuron. A perceptron takes several inputs, x1,x2,…x1,x2,…, and produces a single binary output (0 or 1). If x1 ,x2, x3 are the inputs to the perceptron and w1,w2, w3 are the co-responding weights for those inputs then the output (either 0 or 1) is determined by whether the weighted sum ∑jwjxj is greater than some threshold value. If the weighted sum is greater than the threshold value, the perceptron fires (output is 1). This process mirrors the working of a biological neuron.

[15] https://www.xenonstack.com/blog/data-science/artificial-neural-networks-applications-algorithms/

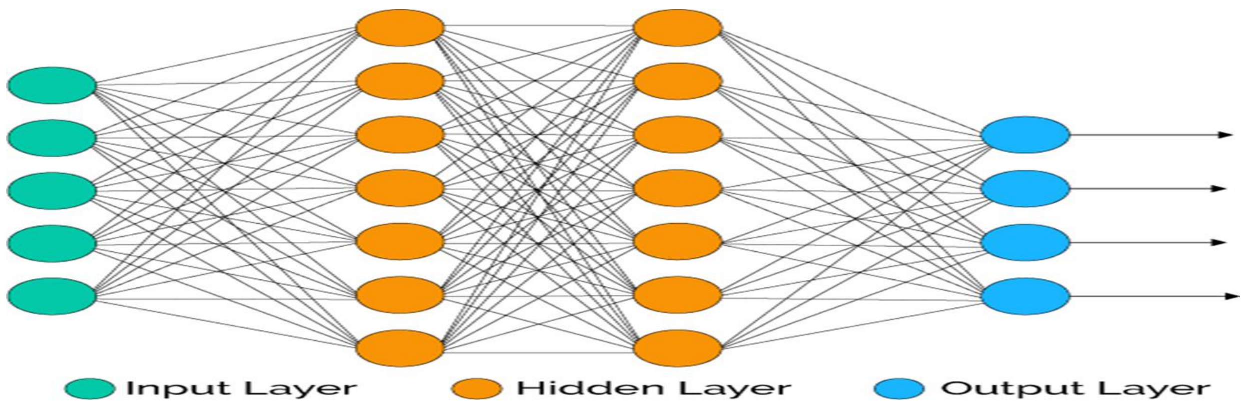
Another way to think of this is to consider the idea of ‘weighing up evidence’. Each perceptron weighs up evidence for each input. In a layer of perceptrons, this decision-making process is hierarchal, i.e. decisions made by previous perceptrons are used as building blocks for decisions made in the future.



In the network above, the first layer of perceptrons makes three decisions. Building on decisions from the first layer, the second layer makes decisions and so on. The decisions in the higher layers are this complex and hierarchal. This idea can be expressed mathematically as

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

A Neural Network contains a large number of artificial neurons called units arranged in a series of layers. In typical Artificial Neural Network, comprises different layers:



Input layer - It contains those units (**Artificial Neurons**) which receive input from the outside world on which network will learn, recognise about or otherwise process.

Output layer - It contains units that respond to the information about how it's learned any task.

Hidden layer - These units are in between input and output layers. The job of the hidden layer is to transform the input into something that output unit can use in some way.

In its simplest configuration (Multi-Layer Perceptron), Neural Networks are fully connected i.e. each hidden neuron is fully linked to every neuron in its previous layer(input) and to the next layer.

The weighted inputs are summed and passed through an activation function. An activation function is a mapping of the summation of the weighted inputs to the output of a neuron. The activation function governs the threshold at which the neuron is activated and the strength of the output signal. Traditionally non-linear activation functions are used. This allows the network to combine the inputs in more complex ways and thus model more complex functions. Non-linear functions like the sigmoid function were used that output a value between 0 and 1 with an s-shaped distribution, and the hyperbolic tangent function (tanh) that outputs the distribution over the range -1 to +1. More recently the ReLu (Rectified Linear Unit) activation function performs best.

MLP - How do Neural networks make a prediction?

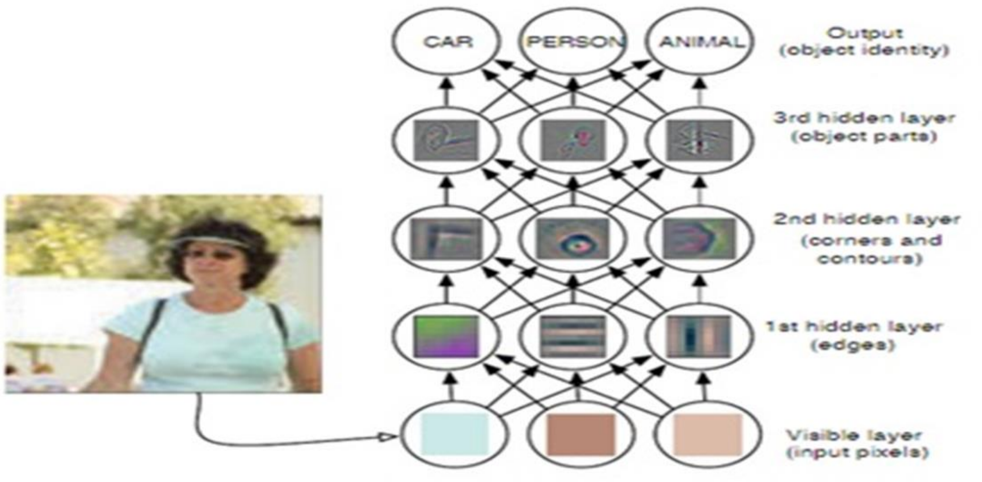

CAR
PERSON
ANIMAL
Output (object identity)
3rd hidden layer (object parts)
2nd hidden layer (corners and contours)
1st hidden layer (edges)
Visible layer (input pixels)

Image source [16]

In the above image, the lowest layer is responsible for identifying the pixels. The layer above it identifies the edges (based on the inputs from the previous layer, i.e. the pixels). The next layer identifies the contours by the same theme. And so on.

A neural network generates a prediction after passing all the inputs through its layers. This process is called ***feedforward***. We "feed" the network with the input, ***x***, and pass it through layer by layer via the activation function until it reaches the output layer. The cost function measures how good our model approximates the real label. Thus, training the neural network involves ***minimising the cost functio***n. We can define our cost function as follows:

$$J(w,b) = \frac{1}{n}\sum \frac{1}{2}(Y - Y^{hat})^2$$

Mean Squared Error

So, the objective is to find some combination of weights(w) and biases(b) that could make our cost **J**, as small as possible. To do this, we'll rely on two critical algorithms which are **Gradient**

[16] https://www.deeplearningbook.org/

Descent and Backpropagation. Note – the bias term is the threshold value when the weights are 0. It is the equivalent of the offset in a linear equation y = mx + b

The backpropagation algorithm involves

a) Propagating the total error backwards through the connections in the network layer by layer
b) calculate the contribution (gradient) of each weight and bias to the total error in every layer
c) then use gradient descent algorithm to optimise the weights and biases,
d) Doing so will eventually minimise the total error of the neural network.

For more details see this reference [17]

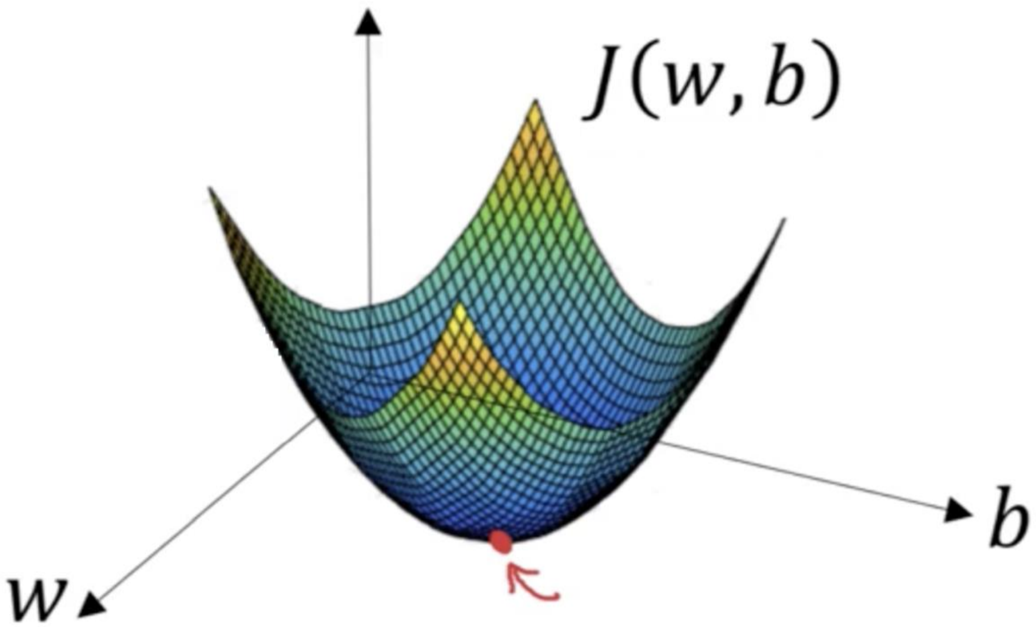


Image source: [18]

[17] https://www.linkedin.com/pulse/gradient-descent-backpropagation-ken-chen/
[18] https://machinelearning-blog.com/2018/02/28/gradient-descent/

We can visualise gradient descent as above. The objective is to find some combination of ***w's*** and ***b*** that could make our cost **J**, as small as possible. To do this, we'll rely on two algorithms which are Gradient Descent and Backpropagation. In the diagram above, the horizontal axes represent our space of parameters, weights and biases, while the cost function, ***J (w, b)*** is then some surface above the horizontal axes. To minimise the cost, we now know that we have to go to the steepest path down the bowl. The direction of the gradient vector, at a given point (the derivative), will point in the steepest direction. Therefore, we'll use the gradient of our cost function w.r.t our weights and our biases to determine the direction to move.

Spatial considerations - Convolutional Neural Networks

Once we understand the basic working of a Multi-Layer Perceptron – we can understand two essential variants. Note that there are many other Neural network architectures and they are evolving rapidly. However, for this discussion, we need an understanding of these three architectures.

We saw previously that for a Multi-Layer Perceptron, the Neural Network is fully connected, i.e. every neuron in a layer is connected to every neuron in adjacent layers, and each connection has its own weight. This connection pattern is generic and makes no assumptions about the features in the data. It is also costly regarding memory (weights) and computation (connections). More importantly, it makes no considerations for spatial similarities in the data. Images have many characteristics that depend on proximity, i.e. local connections. This connection pattern is useful where the data can be interpreted in a spatial manner and where the features to be extracted are spatial. Neural networks that exploit spatial considerations (typically for images) are called Convolutional Neural Networks(CNN). By exploiting spatial considerations, the neural networks are cheaper (more efficient) regarding memory and power. Convolutional neural networks are characterised by locally connected shared weight layers. Consider an image with 28x28 square of neurons

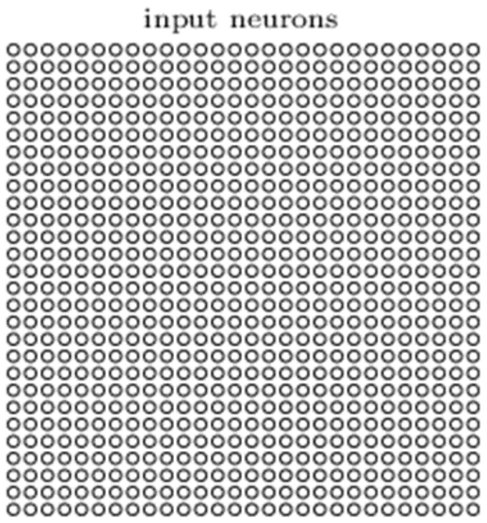


Image source http://neuralnetworksanddeeplearning.com/chap6.html

Instead of connecting every input pixel to every hidden neuron (like the MLP), we connect neurons only from a small localised region of the input image. Hence, each neuron in the first hidden layer will be connected to a small region of the input neurons, say, for example, a 5×5 region, corresponding to 25 input pixels.

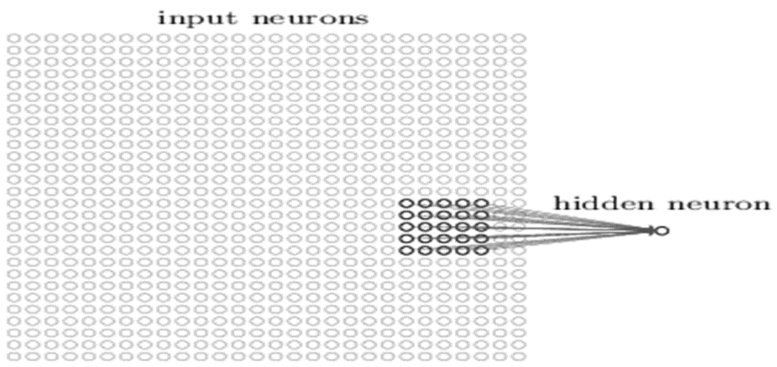


Image source http://neuralnetworksanddeeplearning.com/chap6.html
That region in the input image is called the *local receptive field* for the hidden neuron. It's a little window on the input pixels. The hidden neuron learns to analyse its local receptive field.

We then slide the local receptive field across the entire input image. For each local receptive field, there is a different hidden neuron in the first hidden layer. To illustrate this concretely, let's start with a local receptive field in the top-left corner:

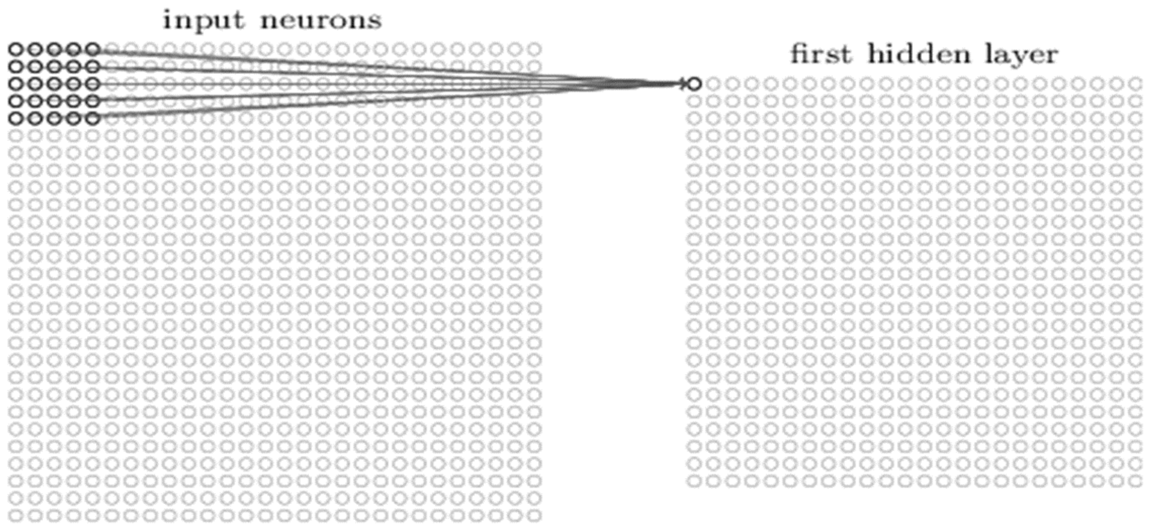


Image source http://neuralnetworksanddeeplearning.com/chap6.html

Then we slide the local receptive field over by one pixel to the right (i.e., by one neuron), to connect to a second hidden neuron:

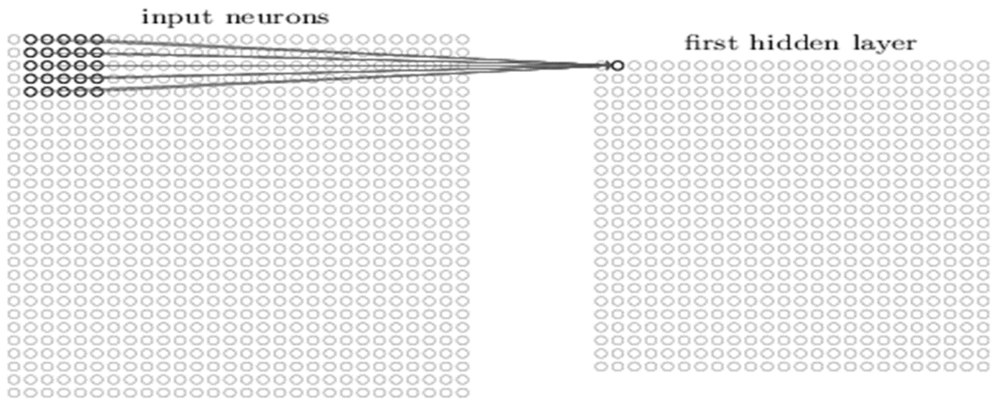


Image source http://neuralnetworksanddeeplearning.com/chap6.html

Sometimes a different *stride length* is used instead of one pixel. **Shared weights and biases:** Each hidden neuron has a bias and 5×5 weights connected to its local receptive field. We can use the *same* weights and bias for each of the 24×24 hidden neurons. This means that all the neurons in the first hidden layer detect the same feature, just at different locations in the input image (ex an edge). Note re terminology:

The map from the input layer to the hidden layer is called a *feature map*.

The weights defining the feature map are called the *shared weights*.

The bias defining the feature map is called the *shared bias*.

The shared weights and bias are often said to define a *kernel* or *filter*. The network structure described so far can detect just a single kind of localised feature (ex an edge). To do image recognition, we need more than one feature map. Hence, a complete convolutional layer consists of several different feature maps. In addition to the convolutional layers, convolutional neural networks also contain ***pooling layers***. Pooling layers are usually used immediately after convolutional layers. Pooling layers simplify the information in the output from the convolutional layer. A pooling layer takes each feature map output from the convolutional layer and prepares a condensed feature map. For instance, each unit in the pooling layer may summarise a region of (say) 2×2 neurons in the previous layer. A common procedure for pooling is known as *max-pooling*. In max-pooling, a pooling unit outputs the maximum activation in the 2×2 input region, as illustrated in the following diagram:

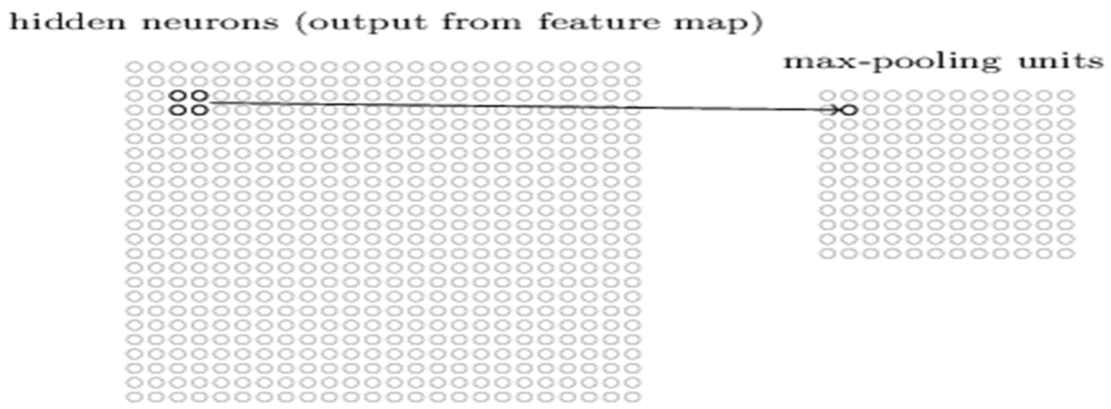
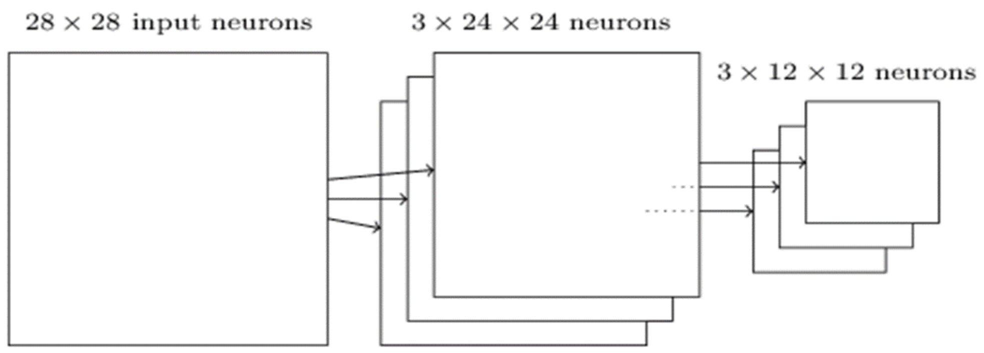


Image source http://neuralnetworksanddeeplearning.com/chap6.html

So, in the case of three feature maps, the combined convolutional and max-pooling layers would be:


28×28 input neurons
$3 \times 24 \times 24$ neurons
$3 \times 12 \times 12$ neurons

Putting it all together: for an output layer of 10 output neurons, corresponding to the 10 possible values for MNIST [19]digits ('0', '1', '2', *etc*):

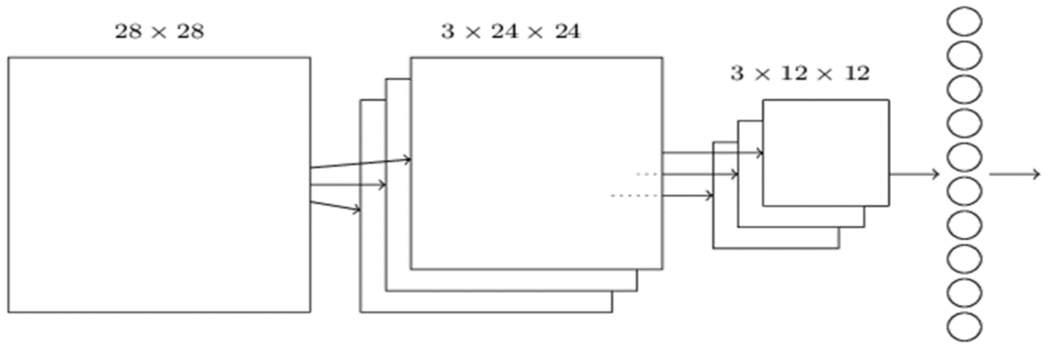


Image source http://neuralnetworksanddeeplearning.com/chap6.html

1) The network begins with 28×28 input neurons, which are used to encode the pixel intensities for the MNIST image.
2) This is then followed by a convolutional layer using a 5×5 local receptive field and 3 feature maps. The result is a layer of 3×24×24 hidden feature neurons.

[19] https://en.wikipedia.org/wiki/MNIST_database

3) The next step is a max-pooling layer, applied to 2×2 regions, across each of the 3 feature maps.
4) The result is a layer of 3×12×12 hidden feature neurons.
5) The final layer of connections in the network is a fully-connected layer. That is, this layer connects *every* neuron from the max-pooled layer to every one of the 10 output neurons.

From a mathematical perspective, “Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers." (reference Bengio – Goodfellow) i.e. instead of doing matrix-matrix multiplication between two layers, the convolution operation can be understood as a "kernel" or "filter" (a small matrix, e.g., 3x3) that slides over an input feature space (e.g., image, or the hidden layer units). Here, one sliding step does an element-wise multiplication between the "kernel" (e.g., 3x3) and an, e.g. 3x3 image patch resulting in an 3x3 output, which is summed up to a single number. This is repeated by sliding over the image then to obtain an output "image".

To summarise, [20]

- A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a CNN typically comprise of convolutional layers, pooling layers, fully connected layers and normalisation layers.
- Convolutional layers apply a convolution operation to the input, passing the result to the next layer.
- Each convolutional neuron processes data only for its receptive field.
- Although fully connected feedforward neural networks can be used to learn features as well as classify data, it is not practical to apply this architecture to images.
- Pooling layers combine the outputs of neuron clusters at one layer into a single neuron in the next layer. For example, max pooling uses the maximum value from each of a

[20] Adapted from http://neuralnetworksanddeeplearning.com/chap6.html

cluster of neurons at the previous layer. Average pooling, which uses the average value from each of a cluster of neurons at the prior layer

- shared weights – same filter used for each receptive field in the layer; reduces memory footprint and improves performance.

Temporal considerations - RNN/LSTM

In the previous section, we saw how CNNs manage spatial relationships. In this section, we will see how Recurrent Neural Networks (RNNs) work with temporal relationships in the data. In a traditional neural network model(MLP), we assume that all inputs (and outputs) are independent of each other. However, that is not true for some applications. For example, MLPs lose spatial information (which is catered to by CNNs). However, they also lose temporal information, i.e. sequence data. For example, in sequence prediction and language translation, the order of the words is important if you want to predict the next word. To manage this, we need a type of neural network called Recurrent neural networks. In recurrent neural networks(RNN), the RNN performs the same task for every element of a sequence and the output at each element also depends on the previous computations. Because of this characteristic (the ability to use the outputs from previous calculations), RNN can be considered to have a “memory”.

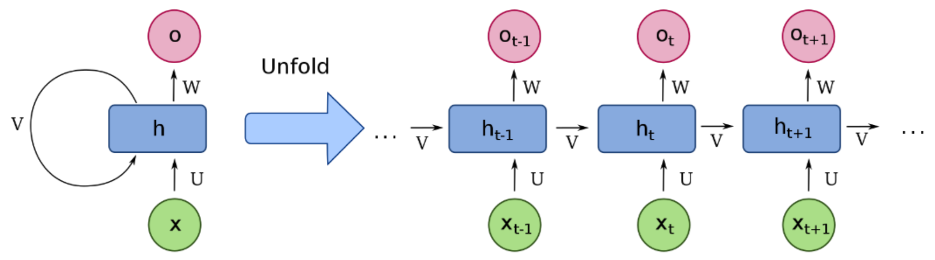
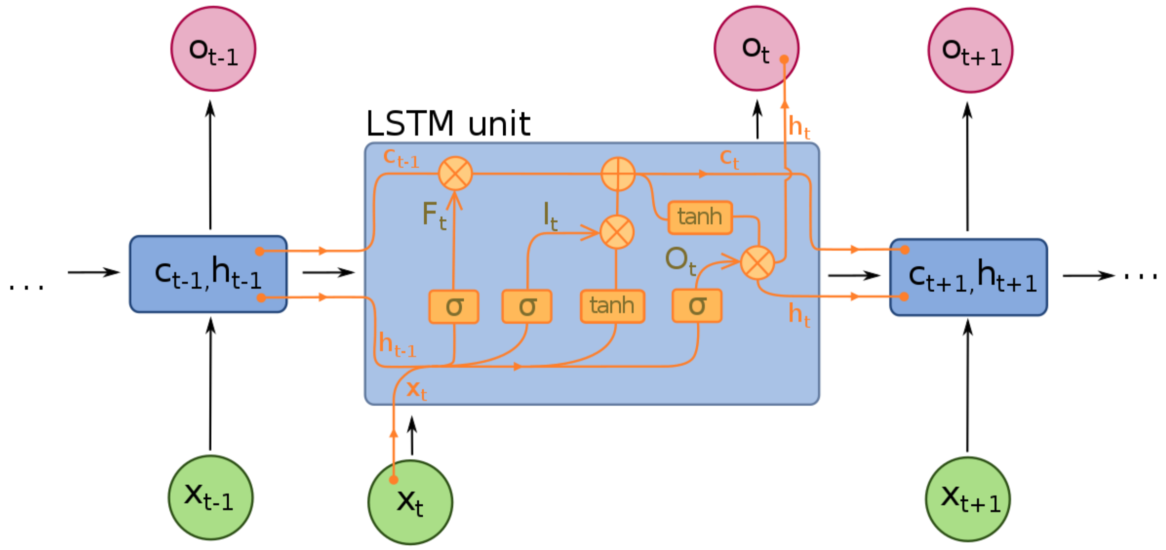


Image source: Wikipedia[21]

[21] https://en.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Recurrent_neural_network_unfold.svg

The most commonly used type of RNNs are LSTMs (Long Short-Term Memory), which are much better at capturing long-term dependencies in sequences. Training a RNN is similar to training a traditional Neural Network through backpropagation. However, RNNs share parameters across all time steps in the network. Hence, instead of using Backpropagation, we use a technique called Backpropagation through time.



LSTM – image source Wikipedia [22]

LSTM networks are a type of RNNs which use a different function to compute the hidden state. The memory in LSTMs are called *cells* which function as black boxes that take as input the previous state. LSTMs are designed to manage long sequences.

The significance of Deep Learning

Is there something special about Deep Learning algorithms? In this section, we explore this idea further by considering two concepts: the notion of **representation learning and** the concept of Deep learning as a **universal function approximator**.

[22] https://en.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Long_Short-Term_Memory.svg

Deep Learning provides better representation for understanding the world

In the seminal paper Representation Learning: A Review and New Perspectives by Yoshua Bengio, Aaron Courville and Pascal Vincent[23] , the authors make a case for representation learning when they say that "***An AI must fundamentally understand the world around us, and we argue that this can only be achieved if it can learn to identify and disentangle the underlying explanatory factors hidden in the observed milieu of low-level sensory data***." Representation learning is one of the critical differentiators for AI. The performance of machine learning methods is heavily dependent on the choice of data representation (or features) on which they are applied. Hence, data scientists often spend a large part of their time in the preprocessing stage which is designed to create an appropriate representation of the data such that it can be most effectively used by the algorithm. Data preprocessing is expensive, labour intensive and depends on prior knowledge.

In his 2006 paper, a breakthrough idea was suggested by Geoff Hinton that proposed a central notion that unsupervised pre-training, was needed to learn a hierarchy of features one level at a time. Finally, the set of layers could be combined with a neural network classifier. This idea forms the central theme of deep learning and makes Deep Learning is unique because it helps to understand the features that comprise an environment without explicitly engineering those features. Hence, if representation leaning is essential – then what makes a good representation?

Smoothness: We can see the Deep Learning process as the ability to learn a function. The function to be learned must be smooth. A smooth function is a function that has continuous derivatives over the area of interest.

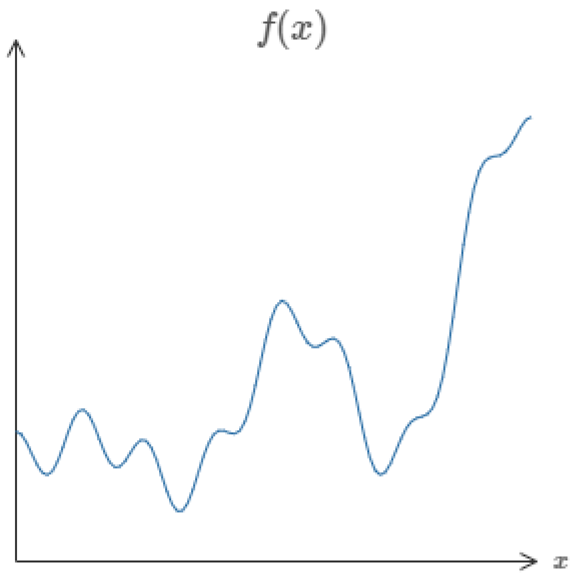
Distributed representations: Good representations, i.e. a reasonably-sized learned representation can capture a huge number of possible input configurations. The distribution must be able to unentangle factors of variation (such as shadows in images)

[23] https://arxiv.org/pdf/1206.5538.pdf

A hierarchical organisation of explanatory factors: Deep Learning suits problems where the features are hierarchical, i.e. abstract concepts are described in terms of other simpler concepts.

Deep Learning a Universal Function Approximator

A very interesting property of Deep Learning is that **deep learning can be seen as a universal approximator to represent any function.** Feedforward networks can provide a universal system for representing functions. Hence, for a given function, there exists potentially a feedforward network that approximates the representation of the function. The universal approximation theorem [24] proposes that regardless of what function we are trying to learn, we know that a large MLP (coupled with sufficient computing resources) will be able to represent this function. For example, a straight line is represented by the equation y = max +c. In contrast, the squiggly function is not easily representable.



[24] https://en.wikipedia.org/wiki/Universal_approximation_theorem

Image source: [25]

The significance of deep learning is that any such function can be represented given enough depth and resources. This means, that whatever function we want to compute, there exists potentially a neural network which can do the job. This is true even if we have just a single hidden layer.

What functionality does AI enable for the Enterprise?

Technical capabilities of AI

As we have seen previously, Artificial Intelligence (AI) is based on Deep Learning. In this book, we first explored Deep Learning technologies and what they can do. We now explore how these technologies apply to the Enterprise. AI is a buzzword and is often used loosely to apply to many things. When applied to the Enterprise, Deep Learning based technologies are typically used along with Machine learning technologies. In a purist sense, AI would only involve Deep Learning. In a pragmatic sense, if a task can be achieved by Machine Learning, it makes more sense to apply Machine Learning than Deep Learning. Firstly, let us see how features do Deep Learning technologies enable within the Enterprise. [26]

Gesture Recognition	The ability to analyse human actions and extract real-time information from video streams. Applies to Robots, Fall Detection, In-car monitoring, Security
Voice Biometrics or Authentication	The ability to use voice as an authentication method and identify individual speakers with speech recognition. alternatively, use voice as the authentication method
Machine vision	Used in robotics, face recognition, content moderation etc.
Video surveillance	Use of AI in images from surveillance camera – often on the Edge, i.e. at the device itself. as digital brains to match their eyes

[25] http://neuralnetworksanddeeplearning.com/chap4.html
[26] This section adapted from http://web.appliedai.de/use-case-library-v1/ AI use case library.

Handwriting Recognition	Recognise and extract handwritten text
Text recognition and generation	Generating text – i.e. transforming data into a written narrative
Entity Recognition	Detect objects in images
Natural Language Processing	Text analysis and the ability to extract information about people, places, events from text documents. Also includes sentiment analysis, content classification and language translation
Movement Prediction	Prediction of movement based on images.
Creating images from text	The ability to create images from text
Attention, Sentiment Tracking	tracking of emotional response based on facial movement and other techniques.

Functionality enabled by AI in the Enterprise value chain

Based on the technical capabilities which AI brings to the Enterprise, AI can impact a range of functions in the value chain. Examples of these include:

Sales and marketing

Demand prediction	Demand prediction and planning

Production

Quality control	Using Artificial Intelligence to Improve Quality Control using Computer vision
Collaborative Robots	Robots which can work with humans to augment human capacity
Robot Training	The ability to train a robot by demonstration by a human operator. This enables the robot to perform the same task by using technologies like computer vision and reinforcement learning.

Procurement

Stock forecasting (warehouse)	Forecasting stock levels
Supply chain management	AI can be used in Supply chain management in many ways. For example, use of Chatbots in procurement, Machine Learning for Supply Chain Planning and in warehouse management. In future, autonomous vehicles and drones could be used in supply chain management. Autonomous Robots are already a key part of supply chain management.
Sourcing Analytics	Use of AI and ML for automating the processes of collecting, cleaning, classifying and analysing expenditure data in an organisation.
Supply Chain Analytics	The Industry 4.0 initiative [27]Drives Supply chain analytics across the value chain and aims to automate many of the existing manufacturing and distribution processes. Many 'autonomous' technologies have a role to play here including autonomous robots and drones.
Smart Contract Management	The ability of AI to empower legal teams and create a better workflow for contracts

Legal

Theft Detection	Automated detection of theft from CCTV images ex: potential shoplifters
Violence Detection	Using Drone-based surveillance for violence detection in crowds
IP infringement detection	Ability to spot IP infringement from images ex the misuse of Logo and Brand

[27] https://en.wikipedia.org/wiki/Industry_4.0

Intrusion detection	Using video surveillance for intrusion detection
Malware detection	Ability to detect malware in an Enterprise by comparing against rapidly changing patterns

Customer support

Cross-selling	Cross-selling involves combining data from multiple sources (often in near real time) to make an offer to a customer. For example, from the CRM system we extract the customer data; from the transactional system, we extract the history of previous transaction amounts etc.
Recommender and Personalization	The ability to recommend the latest offers
Churn prevention	Forecast customer churn and suggest measures to counter churn.
Dynamic pricing	Now commonly used by online retailers and other customers, i.e. offers that vary rapidly on changing customer behaviour but at the same time also consider the company's inventory. Dynamic pricing is especially applicable to businesses with high fixed costs and low margins.
Predictive Lead Scoring	Lead Scoring – the ability to qualify leads/prospects
Influencer discovery	The ability to discover social media influencers online
Next Best Action	Determine the next best action for each customer – as the customer’s behaviour changes, the recommended action also changes
PR analytics	Ability to write reports for the media
Social Media Analysis	Create engaging content for your brand
Chatbots	The quintessential application for AI in customer services, Chatbots are quickly becoming popular across many areas of the Enterprise to engage with customers

Creating a business case for Enterprise AI

Firstly, let us consider the business case for Enterprise AI. To formulate the ideas in this section, we studied the criteria used by VC firms to evaluate AI start-ups and then applied these to the deployment of AI in large Enterprises. The VC references used are listed at the end of this section.

Before we discuss the considerations for building the AI model, let us first outline the measures of success, i.e. how does the AI model create business value, Creation Value by AI is subjective, but some considerations apply:

- Is the amount reflected in business metrics like conversion, churn and cost savings?
- Do we get significantly improved performance over existing machine learning or rule-based algorithms?
- Does AI improve business processes and thereby create new value?
- Is there a cost-benefit regarding optimising employee costs?
- Can the AI identify the hidden rules/hierarchy?
- Can AI provide near-human or ideally better-than-human, levels of performance?
- Does training data exist? Is it labelled?
- Does the application require a high level of trust (ex: self-driving cars)?
- Does the application need a high level of control? (human intervention as required – for example in some medical applications)
- Domain complexity – for example, is there a need for extensive feature engineering
- Do we need to develop proprietary algorithms?
- Impact of regulation on business models including GDPR
- Regulatory transparency – ex: explainable AI
- Risk of adoption especially in the non-consumer space. Many existing applications of AI are in the consumer space where the risks are relatively lower (ex: chatbots). As AI expands into Enterprise and Healthcare domains, the risk of failure and liability increase substantially.

Building on the above considerations, we now consider specific data considerations for building an Enterprise AI business case. **"Data is the new oil"** is a phrase commonly used but Oil is scarce – and (not all) data is limited. So, the analogy does not directly apply. However, Crude oil is less valuable. In that sense, the analogy does apply, i.e. raw data is less valuable (ex: aggregators like Nielsen are valued less than companies that build products on Data and Algorithms – such as Netflix).

A better analogy is that of a Data moat. More generally, at least the following considerations for Data apply to build a business case for Enterprise AI based on a defensive (Data moat) strategy

- Data availability (ideally accessed by an API)
- Data readiness validated through Data quality rules
- Data governance – supported by Governance rules
- Model governance – delivered through an API / container and refreshed
- The idea of Vertical AI comprising Full stack products, Subject matter expertise, Proprietary data, AI offers core value
- Performance threshold and the minimum performance of the algorithm
- Stability threshold and data decay - Machine learning models train on examples taken from the real-world environment they represent. If conditions change over time, gradually or suddenly, and the model doesn't change with it, the model will decay. In other words, the model's predictions will no longer be reliable.
- Data dimensions: Accessibility; Time to acquire' Cost to acquire including licenses; uniqueness; Dimensionality of the data;
- Cost of inclusion of all edge cases
- Perishability of data
- Virtuous loop – access to data to improve the algorithm over time
- Prediction horizon: Algorithms making predictions with long time horizons are challenging to evaluate and develop. Most algorithms model dynamic systems and return a forecast for a human to act upon.

- Hire people to train the algorithms, either as full=time employees or via Mechanical Turk
- Deployment scalability
- Impact of confounding variables - In statistics, a confounding variable is a variable that influences both the dependent variable and independent variable causing a spurious association. Confounding is a causal concept, and as such, cannot be described regarding correlations or associations. Models trained on only co-related variables (vs predictive variables) are likely to be impacted by the presence of confounding variables downstream.
- Cost of data integration from data in different formats
- Cost of data integration arising from data in various sources

The above discussion provides a framework to create an AI business case. Data is a critical component of the Enterprise AI business case. However, as we see above, not all data is created equal -and many Data components play a role in the Enterprise AI business case.

References used in this section

the-mmc-ventures-ai-investment-framework

ai-entrepreneurs-3-things-to-check

ai-adoption-is-limited-by-incurred-risk-not-potential-benefit/

data-rights-are-the-new-ip-rights/

finding-the-goldilocks-zone-for-applied-ai/

data-is-not-the-new-oil/

what-makes-a-successful-ai-company

Four Quadrants of the Enterprise AI business case

Based on the above discussion, let us consider the incremental deployment of AI in the Enterprise. Since Enterprise AI is complex and impacting many areas, we believe that the business case will be evolutionary in most cases.

In progressive orders of complexity (and opportunity) the **four quadrants for the Enterprise AI business case** are:

- **Experiment driven**: Machine Learning and Deep Learning
- **Data driven**: Enterprise Platforms and Data
- **Scale driven**: AI Pipeline and Scalability
- **Talent driven**: AI disruption and Stagnation

Enterprise AI is an abstract concept, interdisciplinary and much-hyped concept. However, in any case, the **deployment of AI in the Enterprise cannot be viewed in isolation**. Within the Enterprise, there already exist systems (like ERP and Data Warehousing). The integration of these will have a role to play in any AI deployment. The word 'Enterprise' can be seen in terms of **Enterprise workflows**. We also consider the core Enterprise (a non-manufacturing company ex Insurance) and the Wider enterprise (including supply chain). Hence, Enterprise AI could be understanding how workflows change when AI is deployed in the Enterprise.

The professional deployment of AI in Enterprises differs from the content in a typical training course. In larger organisations, the Data Science function typically **spans three distinct roles: The Data Engineer, the Data Scientist and the DevOps Engineer**. The Data Scientist is primarily responsible for developing the Machine Learning and Deep Learning algorithms. The Data Engineer and The DevOps Engineer roles work in conjunction with the Data Scientist to manage the product/service lifecycle. Hence, in an Enterprise, managing the AI pipeline involves the philosophy of CICD (Continuous Improvement – Continuous Delivery). CI/CD can be seen as an evolution of Waterfall and Agile methodologies. With this background, let us explore the four quadrants of the AI business case.

Four Quadrants of the Enterprise AI business case

3) Scale driven

AI Pipeline and Scalability
competitive advantage through scale
CICD, Pipeline (with Kafka, Spark), AI deployment at the Edge, containerization, high throughput

4) Talent driven
AI disruption (and Stagnation)
AI first organization
Process alignment with AI at the core
IPR
AI and humans working together
Cobots

1) Experiment driven
Machine Learning and Deep Learning
ML + DL
Model accuracy
Single node

2) Data driven
Enterprise Platforms and Data
Platform strategy, Data driven, Regulation, ERP, Data warehouse, Cloud based (potentially), AutoML, Explainable AI

By Ajit Jaokar -https://www.linkedin.com/in/ajitjaokar/

Experiment driven

At the simplest level, we could model the problem as a machine learning or a deep learning problem. At this stage, we are concerned with the accuracy, choice and the efficiency of the model. Hence, the first quadrant is characterised by experimental analysis to prove value. We are also concerned with improving the existing KPIs. For example, if you are working with fraud detection or loan prediction – each of these applications has an existing KPI based on current techniques. The machine learning and deep learning models would be expected to improve the current benchmarks significantly. We are typically working with one node(non-distributed) processing. The Data could be in Time series, Tabular, Textual, Image, Audio or Video based. The applications could involve Computer vision, NLP, Fintech/financial services, Healthcare, Reinforcement learning, Unsupervised learning (ex GANs, VAE), Emotion AI (Affective computing) etc.

Data-driven

The second quadrant is characterised by

1. Managing Data for algorithms using a variety of sources from the Enterprise

2. Integration with existing systems and platforms (ex: ERP and Data Warehousing)
3. Managing regulatory considerations ex GDPR
4. Estimating the costs of resources
5. Working with the Cloud
6. Strategies which simplify AI deployment (ex: AutoML)

Both ERP and Data Warehousing exist in large Enterprises. Apart from the integration with the existing system and with a Cloud strategy, in this quadrant, we have also to consider

1. Regulation – ex GDPR and Payment regulation
2. Explainable AI
3. Strategies like AutoML and Auto-Keras which simplify AI deployment

Marlene Jia creates a landscape of Enterprise AI companies which categorises AI applications in the Enterprise. We note that the problems are the same or similar as before but are solved more optimally using AI by gaining insights from much larger amounts of (often) unstructured data.

The categories of AI applications include:

- BUSINESS INTELLIGENCE ex Ayasdi;
- PRODUCTIVITY ex: virtual scheduling assistants like X.ai;
- CUSTOMER MANAGEMENT ex Inbenta's AI-powered natural language search;
- HR & TALENT Companies ex: Entelo;
- B2B SALES & MARKETING Salesforce's Einstein;
- CONSUMER MARKETING ex: companies like Lexalytics;
- FINANCE & OPERATIONS ex AppZen which is an automated audit platform that can instantly detect fraud and compliance issues;
- DIGITAL COMMERCE ex Sentient Technologies analyses product recommendations for user actions;
- DATA SCIENCE like RapidMiner;
- ENGINEERING companies like Diffbot;
- SECURITY & RISK ex Demisto (incident response);

- INDUSTRIALS & MANUFACTURING ex GE Predix

The above analysis confirms our previous discussion and also demonstrates that AI will impact many areas of the Enterprise, but in this Quadrant, **the emphasis is on evolution rather than revolution** where companies integrate with existing applications and also gain experience in AI. The challenges in this quadrant are mostly data related – especially the problems of finding labelled data to train the algorithm.

Scale Driven

In the third quadrant, the emphasis is **scaling and in handling real-time transactions**. There are a range of technologies which may be involved here – which mostly come under the category of CICD also a variety of initiatives from Enterprise Cloud providers like Azure ML CI/CD. At the simplest level, we can deploy deep learning models using flask, but more complex strategies could come into play, for example, Mlflow from databricks, Kafka to take on some functions of Enterprise service bus, use of Jenkins 2.0 AI pipeline models for continuous delivery and others. We discuss the approach for CI/CD in the next section.

Talent-driven – leading to process change

Quadrant four is the most interesting. **It is driven by AI talent who can think strategically and technically**. At this stage, we are looking for AI already integrated into the Enterprise and how AI can be used for disruption. This calls for an AI first company as outlined by William Vorhies and four major AI strategies for AI, i.e. Data Dominance, Horizontal, Vertical and Systems of Intelligence. Monica Rogati also talks of an AI hierarchy of needs which also resonates with this approach. In this quadrant, we are working with issues like Process alignment with AI at the core, IPR, AI and humans working together, Cobots etc. The work is driven mostly by Masters/PhD often implementing ideas directly from research papers to create new IP. This quadrant also involves a change in company culture and the role of people within a company.

Discussing the evolution of Enterprise AI in the framework of four quadrants allows us to amalgamate evolution and disruption. The disruption from AI for the Enterprise will be evolutionary. The first two quadrants are based on incremental changes. However, the radical disruption is very real. Hence, the last quadrant involves both disruption and stagnation. AI is a winner takes all game. An alternate working definition of AI is 'processes which improves with experience'. In that sense, the early adopters who will learn from AI will be the future market disruptors because they will create processes which have had the most time to evolve (much like we see Amazon or Google today).

Types of AI problems

We now discuss the types of AI problems

1) Domain expert: Problems which involve Reasoning based on a complex body of knowledge

This includes tasks which are based on learning a body of knowledge like Legal, financial etc. and then formulating a process where the machine can simulate an expert in the field

2) Domain extension: Problems which involve extending a complex body of Knowledge

Here, the machine learns a complex body of knowledge like information about existing medication etc. and then can suggest new insights into the domain itself – for example, new drugs to cure diseases.

3) Complex Planner: Tasks which involve Planning

Many logistics and scheduling tasks can be done by current (non-AI) algorithms. However, increasingly, as the optimisation becomes complex AI could help. One example is the use of AI techniques in IoT for Sparse datasets AI techniques help on this case because we have large and complex datasets where human beings cannot detect patterns, but a machine can do so easily.

4) Better communicator: Tasks which involve improving existing communication

AI and Deep Learning benefit many communication modes such as automatic translation, intelligent agents etc

5) New Perception: Tasks which involve Perception

AI and Deep Learning enable newer forms of Perception which allow new services such as autonomous vehicles

6) Enterprise AI: AI meets Re-engineering the corporation

While autonomous vehicles etc. get a lot of media attention, AI will be deployed in almost all sectors of the economy. In each case, the same principles apply, i.e. AI will be used to create new insights from automatic feature detection via Deep Learning - which in turn help to optimise, improve or change a business process (over and above what can be done with traditional machine learning).

7) Enterprise AI adding unstructured data and Cognitive capabilities to the Enterprise

Majority of the data available to most organisations is unstructured – call logs, emails, transcripts, video and audio data etc. AI can be used to gain insights from this information using techniques like CNNs

8) Problems which impact domains due to second order consequences of AI

'Second-order consequences' cover the broader impact of changing a specific process through AI. For example, introducing AI in healthcare impacts insurance

9) Problems in the near future that could benefit from improved algorithms

A catch-all category for things which were not possible in the past could be possible in the near future due to better algorithms or better hardware. For example, in Speech recognition, improvements continue to be made and currently, the abilities of the machine equal that of a human.

10) Super Long sequence pattern recognition

The application of AI techniques to sequential pattern recognition.

An example of the application of AI in the fintech industry is as below.[28]

- **Customer Service and Engagement:** Conversational Interfaces**,** Virtual Advisors, Customer Service, Marketing, Smart Spending, Passion Fields, Client Segmentation, Sales
- **Investment and Trading:** Investment Strategies, Investment Sentiment Analysis, Investment Reporting, Quantitative Trading, Investment Research, Investment Risk Management, Knowledge Platform, A.I. Trading
- **(Cyber) Risk and Security:** Cyber Incident Investigation, Intrusion Prevention or Detection, Payment Fraud Detection, Authentication, Source Code Scanning, Data Loss Protection, Surveillance, Forensics
- **Regulatory and Compliance (RegTech):** AML, Compliance Advisory, Rogue Trading Prevention, Automated Compliance Monitoring, KYC, Contract Due Diligence, Information Governance
- **Operations:** Recruiting**,** Spend Analysis, Autonomous Documentation, Credit Risk, Automated Reporting, Invoice Processing, Vendor Management, IT Support and Infrastructure Management
- **Others:** Employee Services**,** Expertise Network, Streamlined Mailing, Core Banking, Automated Scheduling

Enterprise AI – Deployment considerations

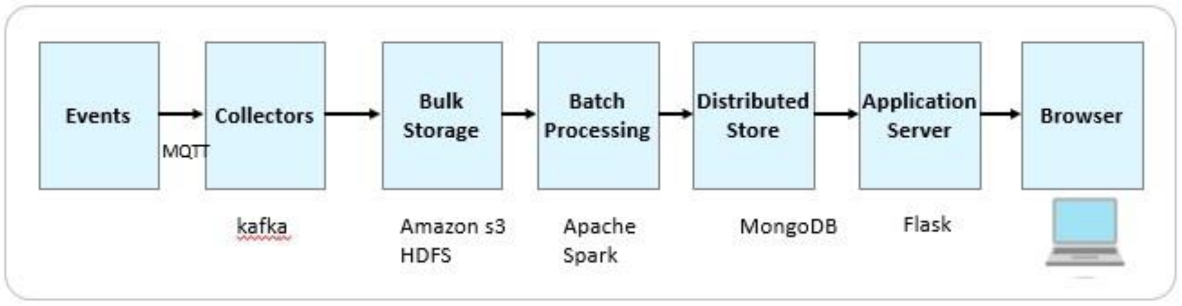
In this section, we consider deployment considerations for AI in large Enterprises.

A methodology to deploy AI applications in the Enterprise

In Enterprises, the deployment of Machine Learning often involves full-stack capabilities.

[28] Banking: why UBS is interested in AI and other fintech innovations ANNIKA SCHRÖDER, UBS AG

A big picture view of such an application is shown below:[29]



We have the following components

Events: represents an occurrence with a relevant timestamp. Events can represent various things (ex: logs from the server).

Collectors are event aggregators who collect events from various sources and queue them for action by real-time workers. Typically, Kafka or Azure event hub may be used at this stage.

Bulk storage – represents a file system capable of high I/O – for example S3 or HDFS

Distributed document store – ex MongoDB

A **web application server** – ex: flask, Node.js

In the above example, the machine learning is implemented in Spark – ex PySpark. In any case complex AI applications need a methodology for deployment in an Enterprise.

In this final section, we explore a methodology to deploy AI applications in large enterprises. The ability to implement AI in a sophisticated setting forms a key component of Enterprise AI and a significant differentiator from traditional machine learning. The Team Data Science Process (TDSP) [30] is an agile, iterative data science methodology to deliver predictive analytics solutions and intelligent applications efficiently. Created by the Microsoft Azure team, TDSP helps improve team collaboration and learning**.**

TDSP comprises of the following key components:

- A **data science lifecycle** definition

[29] Image source: Agile Data Science, 2.0 by Russell Jurney

[30] **https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/overview**

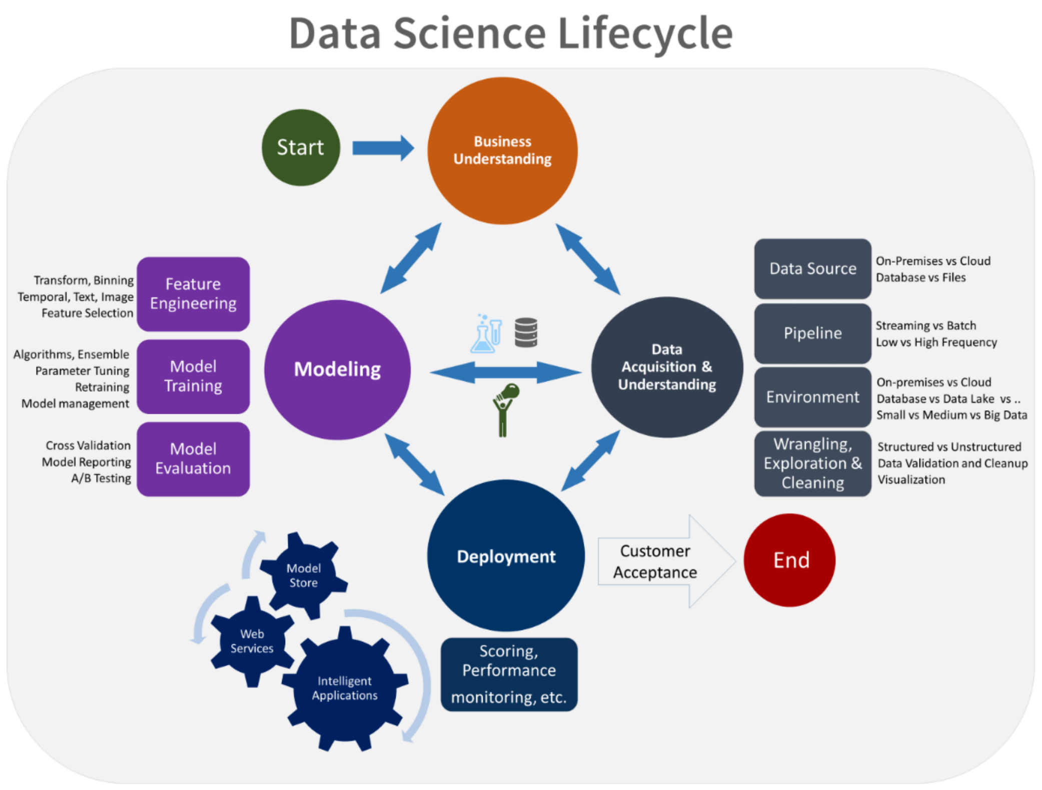
- A **standardised project structure**
- **Infrastructure and resources** for data science projects
- **Tools and utilities** for project execution

A methodology is needed because introducing processes in most organisations is challenging. The methodology along with tools to implement it, reduce the barrier to adoption and provide a consistent approach across the organisation. Team Data Science Process (TDSP) is loosely modelled on traditional data science lifecycle processes like CRISP-DM. TDSP also draws on software code management (SCM) conventions and definitions.

The lifecycle phase of the methodology outlines the significant stages that projects typically execute, often iteratively:

- **Business Understanding**
- **Data Acquisition and Understanding**
- **Modelling**
- **Deployment**
- **Customer Acceptance**

Shown below as



Source: TDSP / Azure

TDSP **project roles** include:

- Solution architect
- Project manager
- Data scientist
- Project lead

According to their documentation: *"TDSP also includes a standardised project structure. Having all projects share a directory structure and use templates for project documents makes it easy for the team members to find information about their projects. All code and documents are stored in a version control system (VCS) like Git, TFS, or Subversion to enable team collaboration. Tracking tasks and features in an agile project tracking system like Jira, Rally, and Azure DevOps allows closer tracking of the code for individual features. Such tracking also enables teams to obtain better cost estimates. TDSP recommends creating a separate repository for each project on the VCS for versioning, information security, and collaboration. The* *standardised structure for all projects helps build institutional knowledge across the organisation."*

There are four distinct roles for the team personnel:

1. ***Group Manager***. Group Manager is the manager of the entire data science unit in an enterprise.
2. ***Team Lead***. A team lead is managing a team in the data science unit of an enterprise. A team consists of multiple data scientists.
3. ***Project Lead***. A project lead manages the daily activities of individual data scientists on a specific data science project.
4. ***Project Individual Contributor***. Data Scientist, Business Analyst, Data Engineer, Architect, etc. An individual project contributor executes a data science project.

The **workflow** includes steps that can be grouped into three activities:

- Sprint planning (Project Lead)
- Developing artefacts on git branches to address work items (Data Scientist)
- Code review and merging branches with master branches (Project Lead or other team members)

In the TDSP sprint planning framework uses familiar development terminology modelled on conventional SCM(Software Code Management) processes.

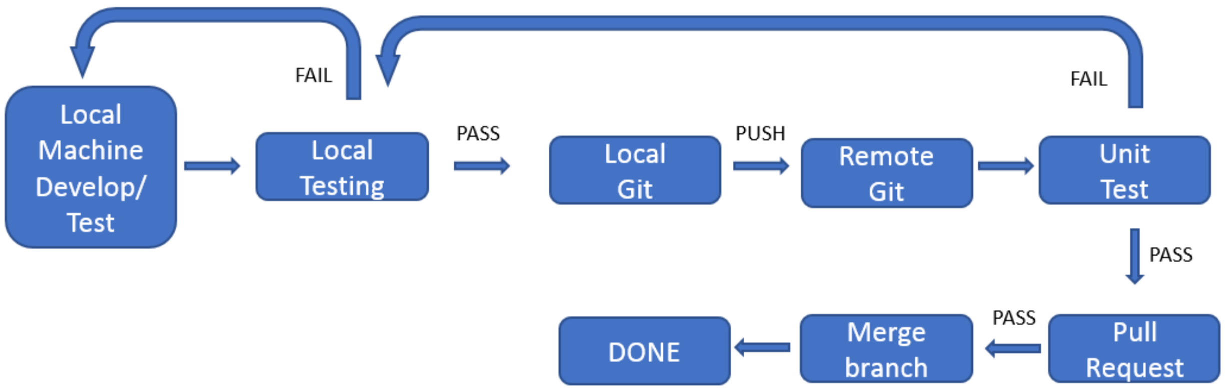
For example, Stories are different work items that are needed to complete a feature (project) end-to-end. Examples of stories include:

- Getting Data
- Exploring Data
- Generating Features
- Building Models
- Operationalising Models
- Retraining Models

Note

Concepts are borrowed of features, stories, tasks, and bugs from software code management (SCM) to be used in data science. They might differ slightly from their conventional SCM definitions.

The overall workflow of testing code in a data science project looks like this:



DevOps and the CI/CD philosophy

Previously, we mentioned that there are three roles in an AI pipeline: The Data Engineer – The Data Scientist and the Devops Engineer. When an AI application is built, there are two main streams of work: firstly, the App developers and Data Engineers who make the application which faces the end customer and secondly the Data Scientists who build machine learning models. The models are encapsulated in the App and use the data acquired from the data engineering phase. The third phase (DevOps) involves the task of deploying the app often in a CI/CD mode (continuous improvement – continuous delivery).

Continuous integration (CI) and continuous delivery (CD) embodies a philosophy [31]based on a set of operating principles and practices that enable application development teams to deliver code changes more frequently and reliably. The CI/CD pipeline is typically implemented by the DevOps team. CI/CD applies to the deployment of AI applications in the Enterprise.

[31] https://www.infoworld.com/article/3271126/ci-cd/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html

CI/CD enables development teams to implement small changes and check-in code to version control repositories frequently. The CI/CD approach is necessary because modern applications require developing code in diverse platforms and tools. These applications need a mechanism to integrate the incremental changes from individual developers reliably. In this context, CI/CD aims to establish a consistent and automated pipeline to build, package, test and deploy changes. With a consistent mechanism in place, the collaboration between team members improves and frequency of code commits increase – leading to better software quality.

CI/CD includes the following steps:

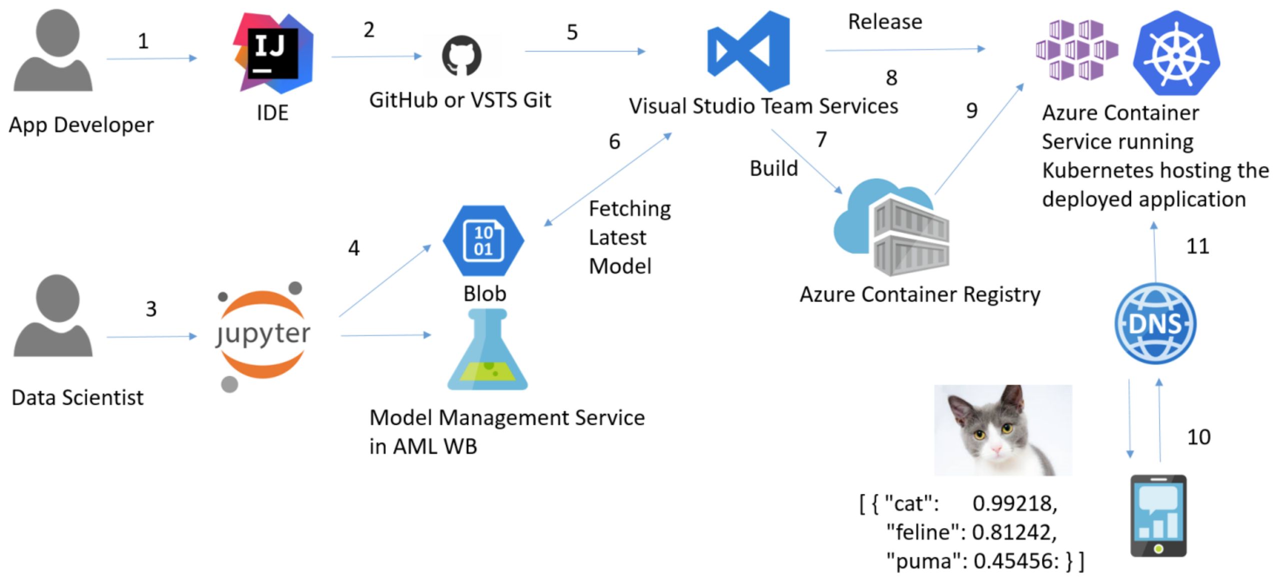
Automating the build process: The build process is automated by packaging all the software, database, and other components. The automation will also execute unit tests.

Continuous integration typically involves developers committing their code into the version control repository frequently. It is easier to identify defects and other software quality issues on smaller code segments.

Continuous delivery (CD) automates the delivery of applications to specific environments (ex: production, development and testing). CD also manages all the ancillary tasks in this process ex: starting web servers, databases etc. A CI/CD tool such as Jenkins or Travis is in this process.

Continuous testing: Implementing a set of automated regression, performance, and other tests that are executed as part of the CI/CD pipeline.

Container management: CI/CD pipelines on cloud environments also typically involve management using containers such as Docker and Kubernetes.
The process is illustrated using azure DevOps as below


App Developer
1
IDE
2
GitHub or VSTS Git
5
Visual Studio Team Services
Release
8
9
Azure Container Service running Kubernetes hosting the deployed application
7
Build
Azure Container Registry
6
Fetching Latest Model
Blob
4
Data Scientist
3
Model Management Service in AML WB
11
DNS
10
[{ "cat": 0.99218, "feline": 0.81242, "puma": 0.45456: }]

Steps of the CI/CD pipeline

1. Developer work on the IDE of their choice on the application code.
2. They commit the code to source control of their choice
3. Separately, the data scientist work on developing their model.
4. Once happy, they publish the model to a model repository.
5. A build is kicked off in Azure DevOps based on the commit in GitHub.
6. Azure DevOps Build pipeline pulls the latest model from Blob container and creates a container.
7. Azure DevOps pushes the image to a private image repository in Azure Container Registry
8. On a set schedule (nightly), release pipeline is kicked off.
9. Latest image from ACR is pulled and deployed across Kubernetes cluster on ACS.
10. Users request for the app goes through DNS server.
11. DNS server passes the request to load balancer and sends the response back to user.

Conclusions

To conclude, Enterprise AI is a rapidly moving goalpost. It is a complex and cross-functional domain. However, we believe that the deployment of AI in the Enterprise will be more structured. It will consider existing systems in the Enterprise (like ERP) and will follow the 'quadrants' approach we have listed in this book. All areas of the Enterprise will be affected. AI will incorporate both machine learning and Deep Learning. Ultimately, it will affect jobs and roles of people in large and small companies.

AI is not a panacea. AI comes with a cost (skills, development, and architecture) but provides an exponential increase in performance. Hence, AI is ultimately a rich company's game. However, AI is also a 'winner takes all' game and therefore provides a competitive advantage. The winners in AI will take an exponential view addressing very large-scale problems, i.e. what is possible with AI which is not possible now?
We live in interesting times!

Enterprise AI - An Application Perspective

Version One – Oct 2019

By

Ajit Jaokar and

Cheuk Ting Ho

Available exclusively on Data Science Central with free access

https://www.datasciencecentral.com/profiles/blogs/free-ebook-enterprise-ai-an-applications-perspective

The book is used as a reference for Ajit and Cheuk's new course on Implementing Enterprise AI

[i] http://web.stanford.edu/class/cs234/index.html