AI 深度學習實務

C. Alex Hu, PhD

Summer, 2019

REFERENCE

(1) **FRANÇOIS CHOLLET** (Keras 原創者)

" Deep Learning with Python, " 2018, Manning Publications

http://www.deeplearningitalia.com/wp-content/uploads/2017/12/Dropbox_Chollet.pdf

[Code] : https://github.com/fchollet/deep-learning-with-python-notebooks

(2) **" Neural Networks & Deep Learning, " Courses**
by Andrew Ng (吳恩達)
https://www.youtube.com/watch?v=CS4cs9xVecg&list=PLkDaE6sCZn6Ec-XTbcX1uRg2_u4xOEky0

(3) Deep Learning video clips:

Ch 1 "究竟神經網路是什麼？ I 第一章 深度學習” https://youtu.be/aircAruvnKk

Ch 2 "Gradient descent, how neural networks learn I Chapter 2, deep learning”
https://youtu.be/IHZwWFHWa-w

Ch 3 "What is backpropagation really doing? I Chapter 3, deep learning"
https://youtu.be/Ilg3gGewQ5U

Ch 3.appendix "Backpropagation calculus I Appendix to deep learning chapter 3"
https://youtu.be/tIeHLnjs5U8

1. AI 深度學習簡介

INTRODUCTION — 未來的世界：工業 4.0 時代

❖ **Amazon 無人商店 =>** ***Online vs. Offline***

Industry 4.0 :

- AI
- Big Data
- IoT
- Blockchains

共同之處 — Python 程式設計

NowThis 新聞 — 2018/1/22 （https://goo.gl/Z16fk7）

“ 亞馬遜首家智慧商店正式對公眾開放，排隊結帳很快將成為過去式。“

Global / Taiwan Workforce Crisis

Global/Taiwan： 少子化問題

Global/Taiwan： 搶 高技術人才？

Labor Shortage
（勞動力不足）

+

Skill Mismatch
（技能不符）

Taiwan
高技術人才？
四年級生
五年級生
六年級生
七年級生
八年級生
九年級生
… …

＜TED Talk＞： Rainer Strack (賴內・斯達克):
2030 年勞動力危機令人震驚 — 如何從現在開始解決！

工業 4.0 時代：目前重要的推手

無人汽車 (自動車駕)

Elon Musk, Apple,

Google, … …

無人商店

Jeff Bezos (Amazon),

7-11, 全家, … …

新零售 / 網路銀行

BAT - 馬雲 (Alibaba),

騰訊, 百度, … …

AI + Big Data + IoT + Blockchains

無人工廠

郭台銘 (工業互聯網, FII),

孫正義, BMW, Tesla,…

半導體產業

張忠謀 (TSMC), 三星,

NVIDIA, 日月光, 美光,

紫光, 鴻海,…

精準行銷 / 演算法

Google, Amazon, Netflix,

Mark Zuckerburg

(FB, Instagram),

Stitch Fix, …

… …

工業 4.0 時代：目前重要的推手

無人汽車 (自動車駕)

Elon Musk, Apple,

Google, … …

無人商店

Jeff Bezos (Amazon),

新零售 / 網路銀行

BAT - 馬雲 (Alibaba),

Q：醫療照護 要如何導入 工業 4.0 技術

(AI, Big Data, IoT, Blockchains) 呢？

AI + Big Data + IoT + Blockchains

－ 以 AI 和 “傳統工業機器人” 為例

=> 醫療智慧型機器人

無人工廠

郭台銘 (工業互聯網, FII),

孫正義, BMW, Tesla,…

半導體產業

張忠謀 (TSMC), 三星,

NVIDIA, 日月光, 美光,

紫光, 鴻海,…

精準行銷 / 演算法

Google, Amazon, Netflix,

Mark Zuckerburg

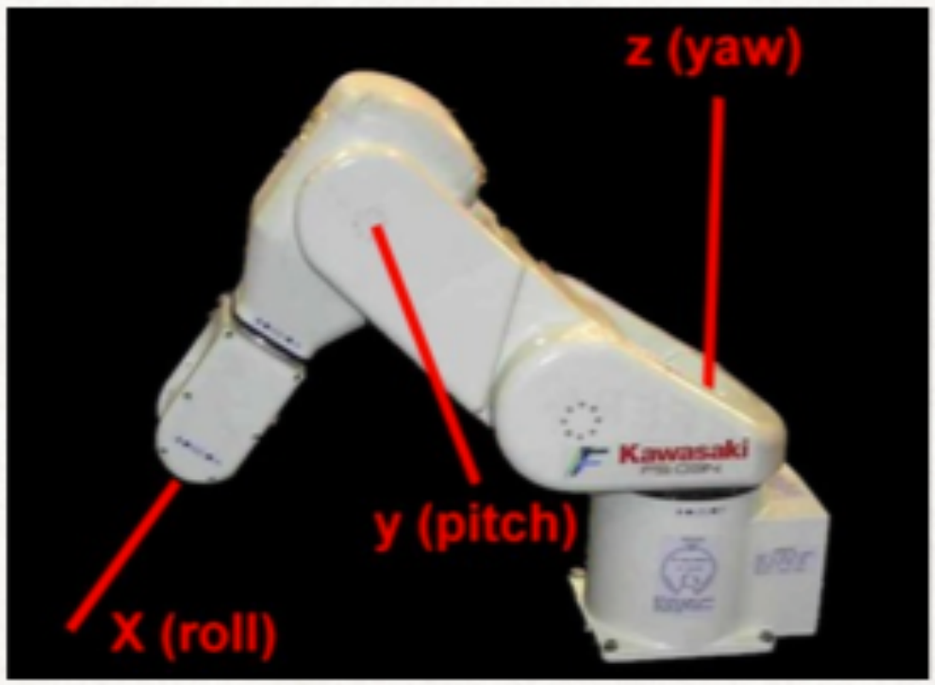
(FB, Instagram),

Stitch Fix, …

… …

AI & Intelligent Robotics (cont'd)

傳統的工業機器人 ： 三軸/六軸機器人 (機械手臂)



https://upload.wikimedia.org/wikipedia/commons/f/fa/Rollpitchyawrob.jpg

ROBOT軸承與馬達自動組裝設備(六軸機械手臂)
https://youtu.be/YZO2di1Is4A

醫療智慧型機器人 – 人機協同作業、機聯網

傳統三軸/六軸
機器人 (機械手臂)

+ AI ?

自然語言處理
(Natural Language Processing)

機器視覺化
(Machine Vision)

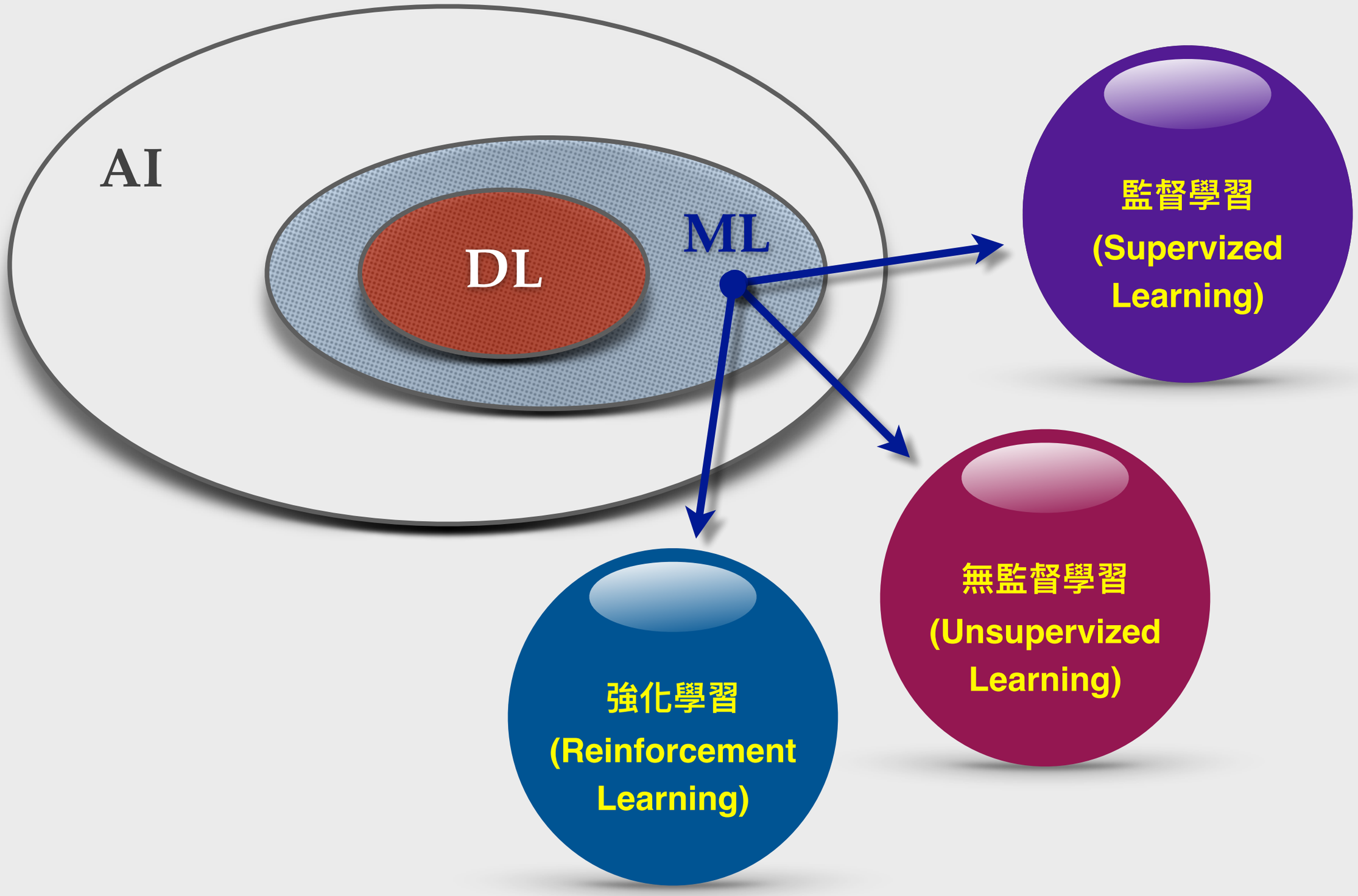
路徑/排程 規劃
(Planning)

機器學習
(Machine Learning)

... ...

智慧機器人範例 – 達文西機器手臂機器操作 (https://youtu.be/eE3oCv1qrh0)

AI, Machine Learning (**ML**) & Deep Learning (**DL**)

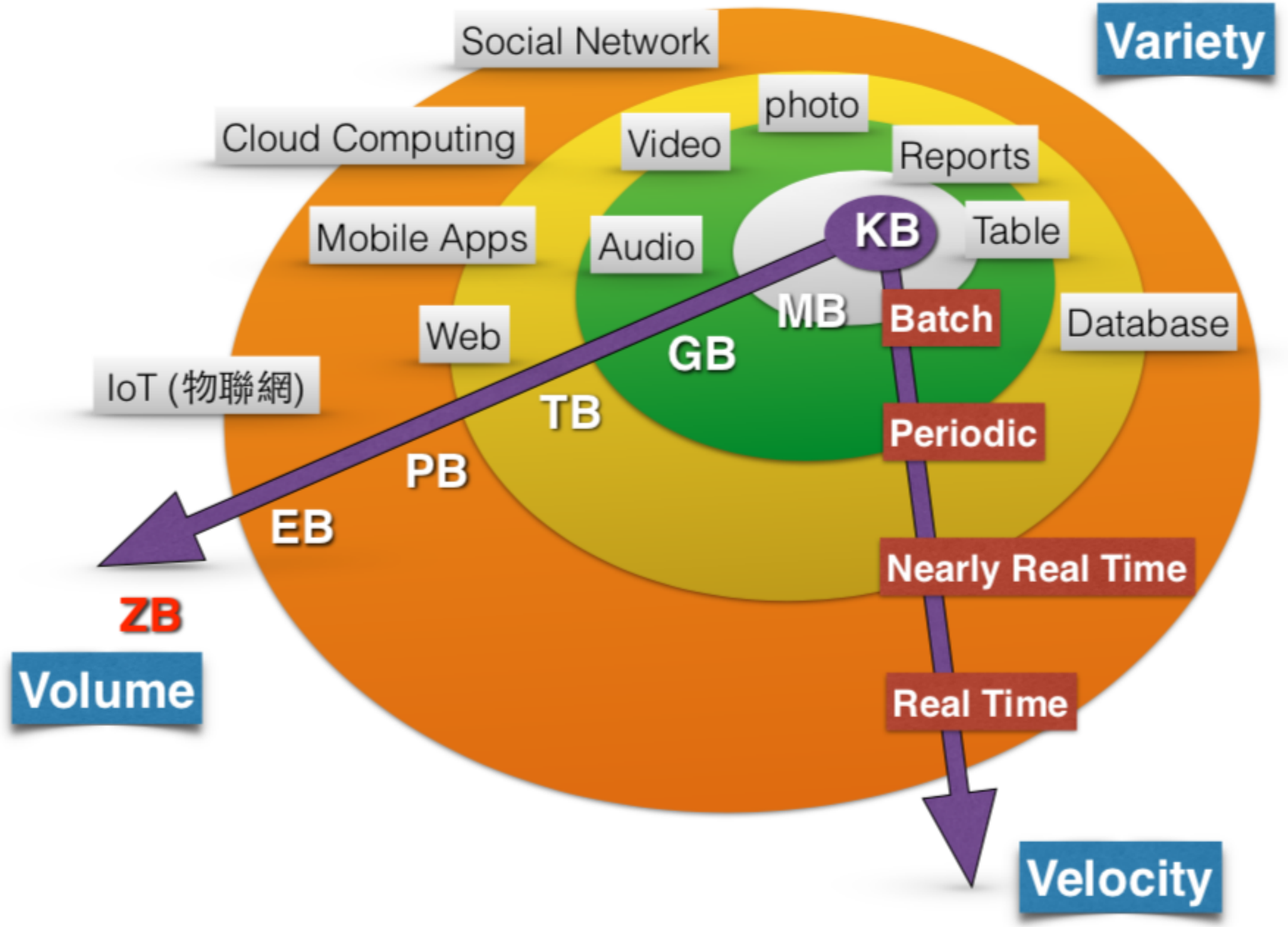


主流人工智慧 ： narrow AI & Big Data

李開復博士 在 “世界經濟論壇” (2018-2-1) 接受專訪時，強調「現在的主流人工智慧是弱人工智慧 (weak AI or narrow AI)」

專訪影片 請連結至 李開復博士 **2018/2/1 Facebook** 網頁: **https://goo.gl/XCp6Vz**

3V's Model of Big Data



AI & Big Data

IPv6 for IoT Devices :

- 2^{128} devices = 2^8 x $(2^{10})^{12}$ ~ 256 x $(10^3)^{12}$

  ~ 10^{38} devices  一百兆兆兆個裝置 !!

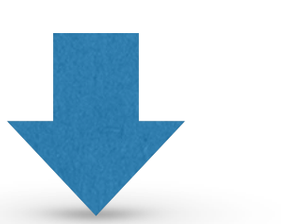
- 若每個 IoT 裝置：每一個月上傳 1GB (10^9 Bytes) 資料量

  則一兆個裝置 (10^{12} devices)： 每一個月將產生 1ZB (10^{21} Bytes)資料量！

Q：什麼時候 “物聯網裝置” 將會普遍使用呢?

A： Blockchains + IoT => AI + Big Data Analytics

AI + Big Data = narrow AI



Deep Learning

原始資料
Raw Data

- Finance
- Med/Health
- Engineering
- … … …

1

資料前置處理
(Data Pre-processing)
☞ 需要 “領域知識”

傳統資料分析
vs.
Big Data 資料分析

2

傳統資料分析流程： R / Python / Scala

資料視覺化
(Data Visualization)

特徵向量擷取
(Feature Vector's Extraction)

資料分析方法：

- 機率模型
- 統計模型
- 資料探勘 (Data Mining)

3

機器學習訓練與測試流程
(Machine Learning Pipeline for Training & Testing)

Big Data 資料分析流程： PySpark, SparkR, Spark/Scala

Data Analytics Project Life Cycle (5 stages)

1

1. Identifying the problem

2. Designing data requirement

3. Preprocessing data

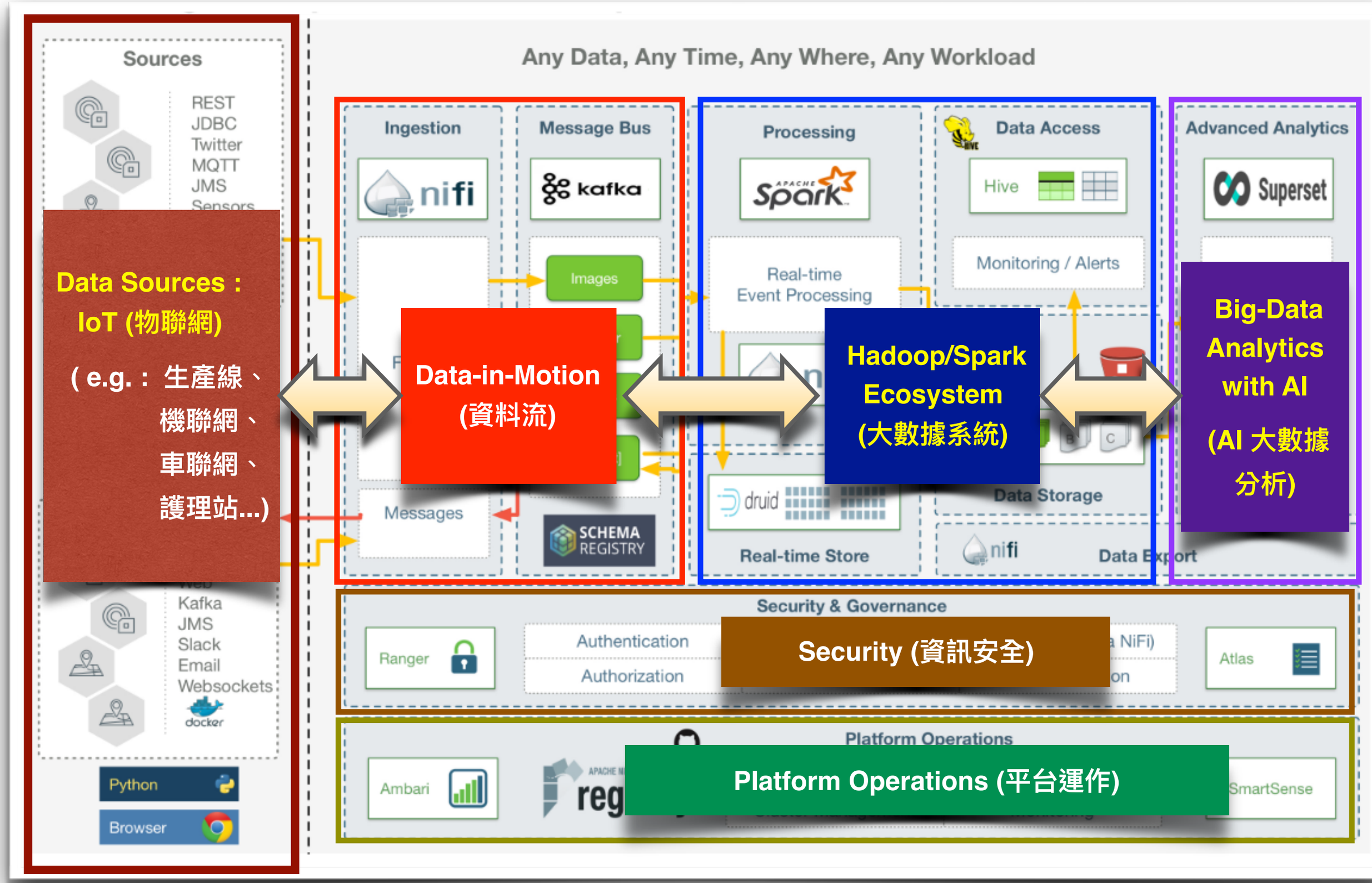
4. Performing analytics over data

5. Visualizing data

Stage 1 ~ 2 :
Data Collection
~ 80% Work-load

Stage 3 ~ 5 :
Data Analysis
~ 20% Work-load

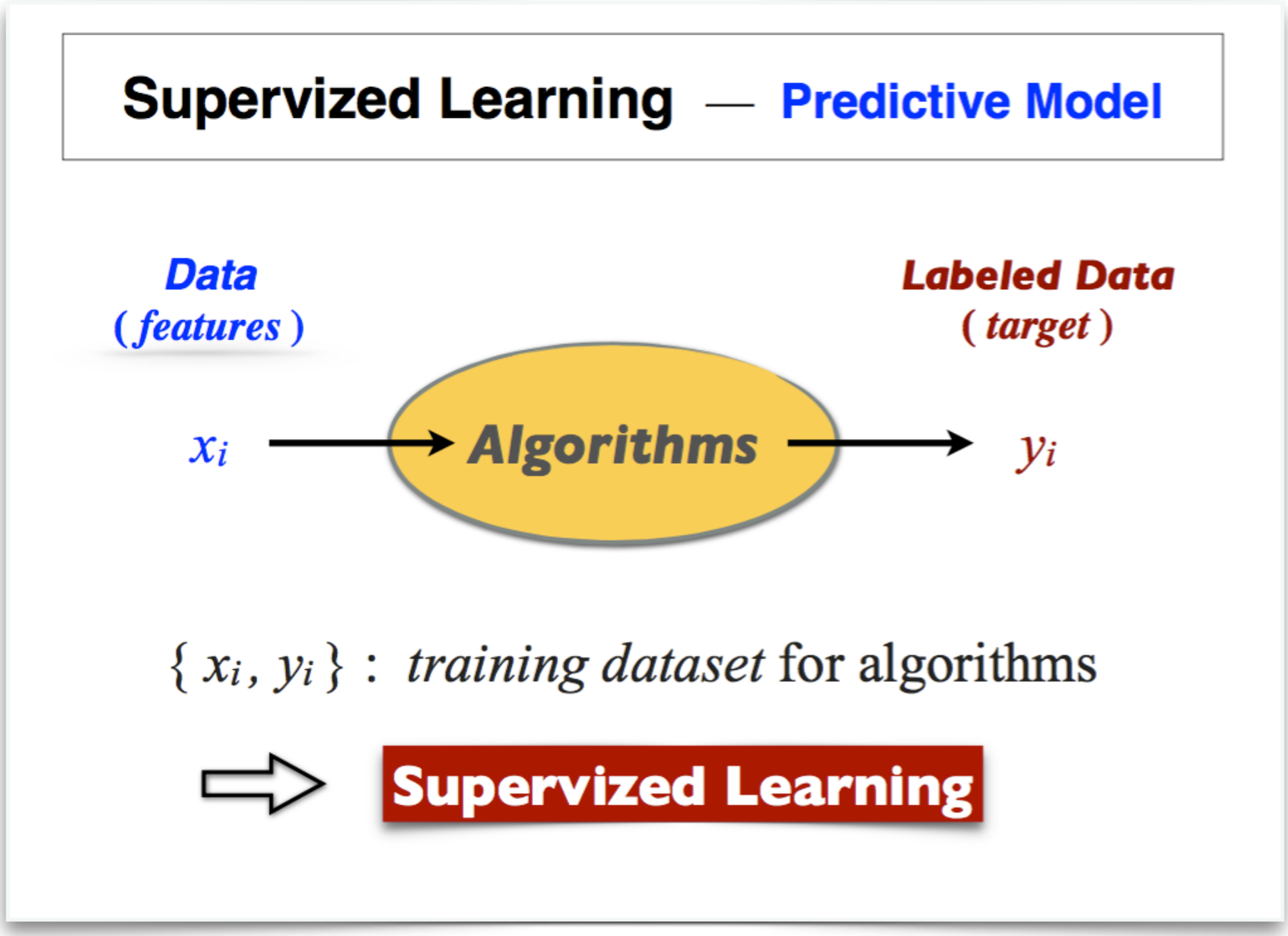
1 Data Engineering



(https://dzone.com/articles/real-time-stock-processing-with-apache-nifi-and-ap)

1. Introduction to Machine Learning

- Supervized Learning
- Unsupervized Learning
- Reinforcement Learning



3 Machine Learning Training & Testing Pipeline

Datasets : Feature + Labeled, (x_i , y)

Training Dataset, *e.g.* 75% ~ 80%

Transformation
(Normalization or Standardization ?)

Algorithm(s)

Evaluation

Testing Dataset, *e.g.* 20% ~ 25%

Training Model

Prediction

3 Machine Learning Training & Testing Pipeline

[需要用到的 Python 技術] ： Scikit-Learn

Q ： 為什麼 Feature Dataset, x_i 要拆解成 75% (or 80%) : 25% (or 20%) ？
可不可以拆解成其他比例呢？ 例如：60% : 40%

Q ： 如何決定該選擇 Normalization 或 Standardization ？

Q ： 如何在 Training 階段 選擇 Algorithms ？
(Validation Curves Diagram — *Under-fit vs. Over-fit*
Learning Curve Diagram — *Accuracy vs. Training Dataset Size*
Training Epochs or Steps — *Convergence of Accuracy*)

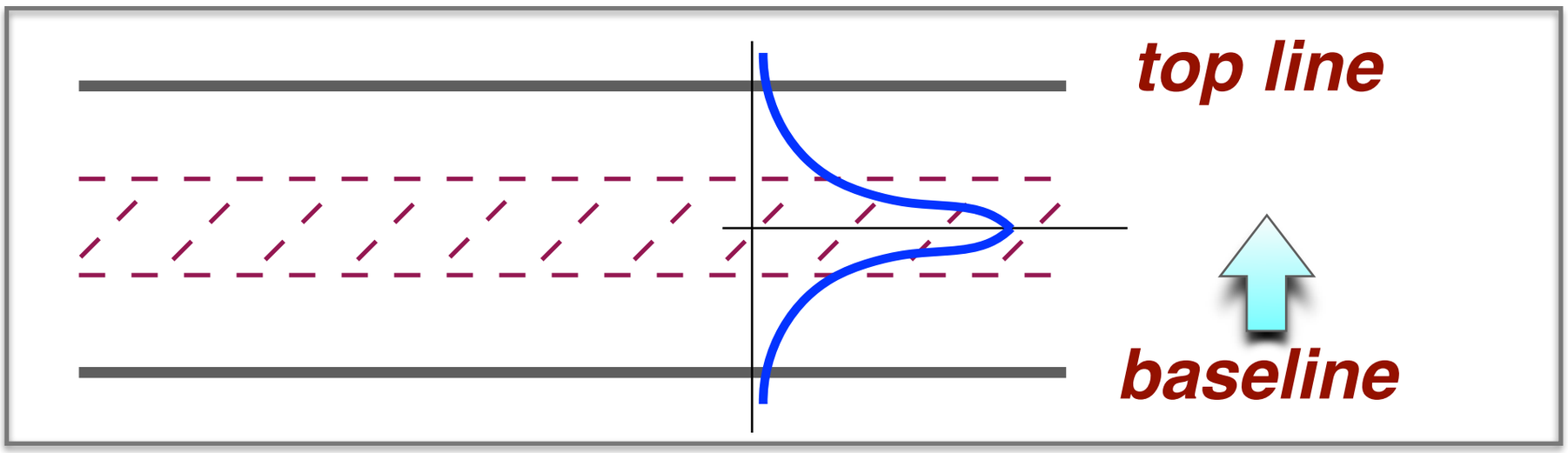
Q ： 如何在 Training 階段，進行 Evaluation (評估) ？ [labeled data]

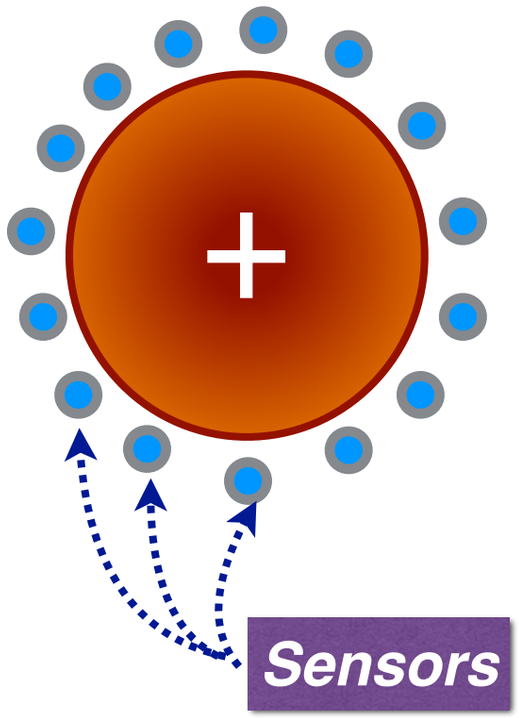
Q ： 如果 Training Model 在 Testing 階段，預測結果表現 “很好” 或
“不好”，該如何處理呢？ (Prediction => Risk Analysis => Strategy)

Data Analysis for **Strategy/Business Model** to Solve Problems

Big Data => Predictive Models => Risk Analysis

$-\nabla T$:
the negative local temperature gradient

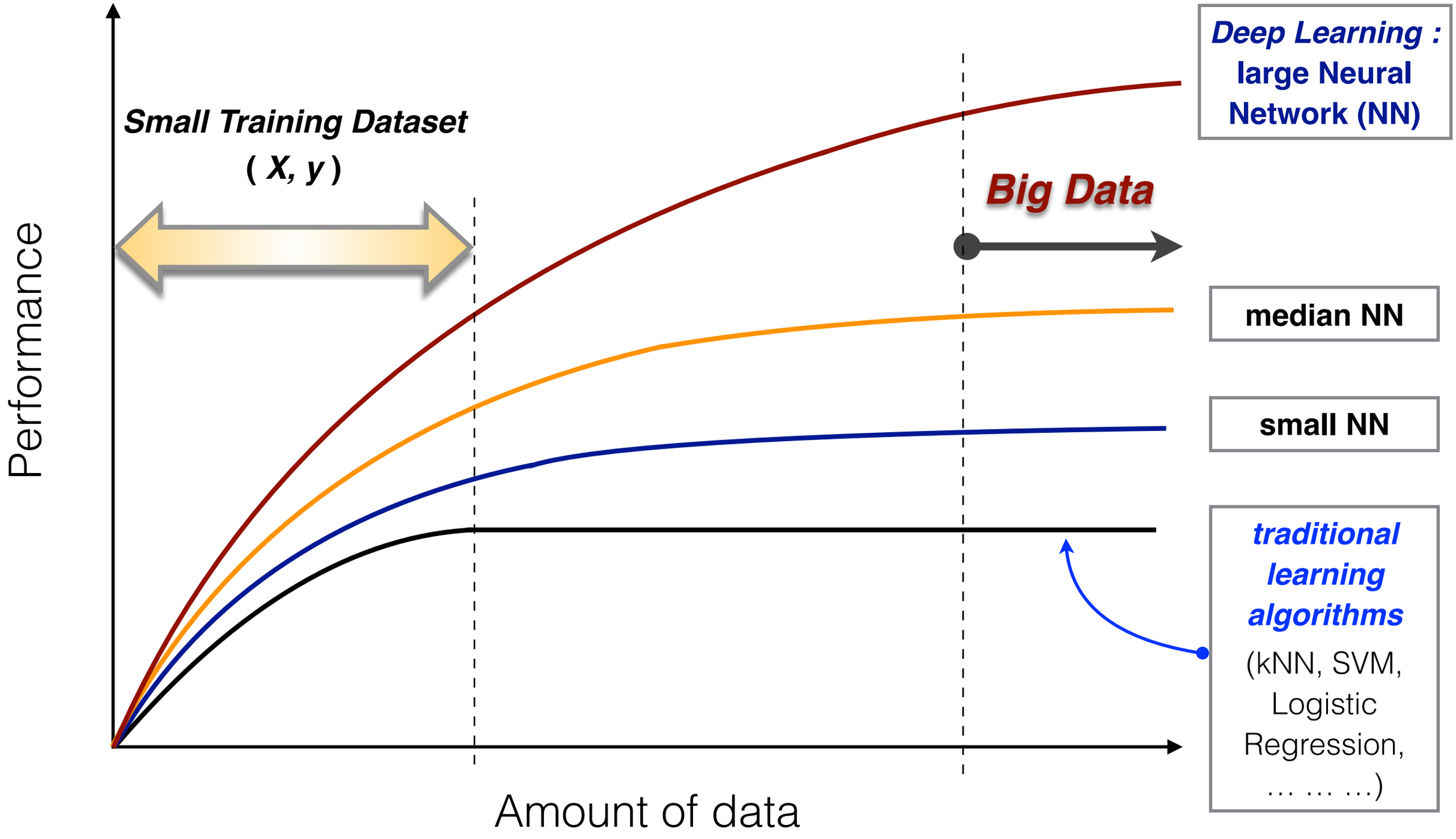




Strategy → **Policy**

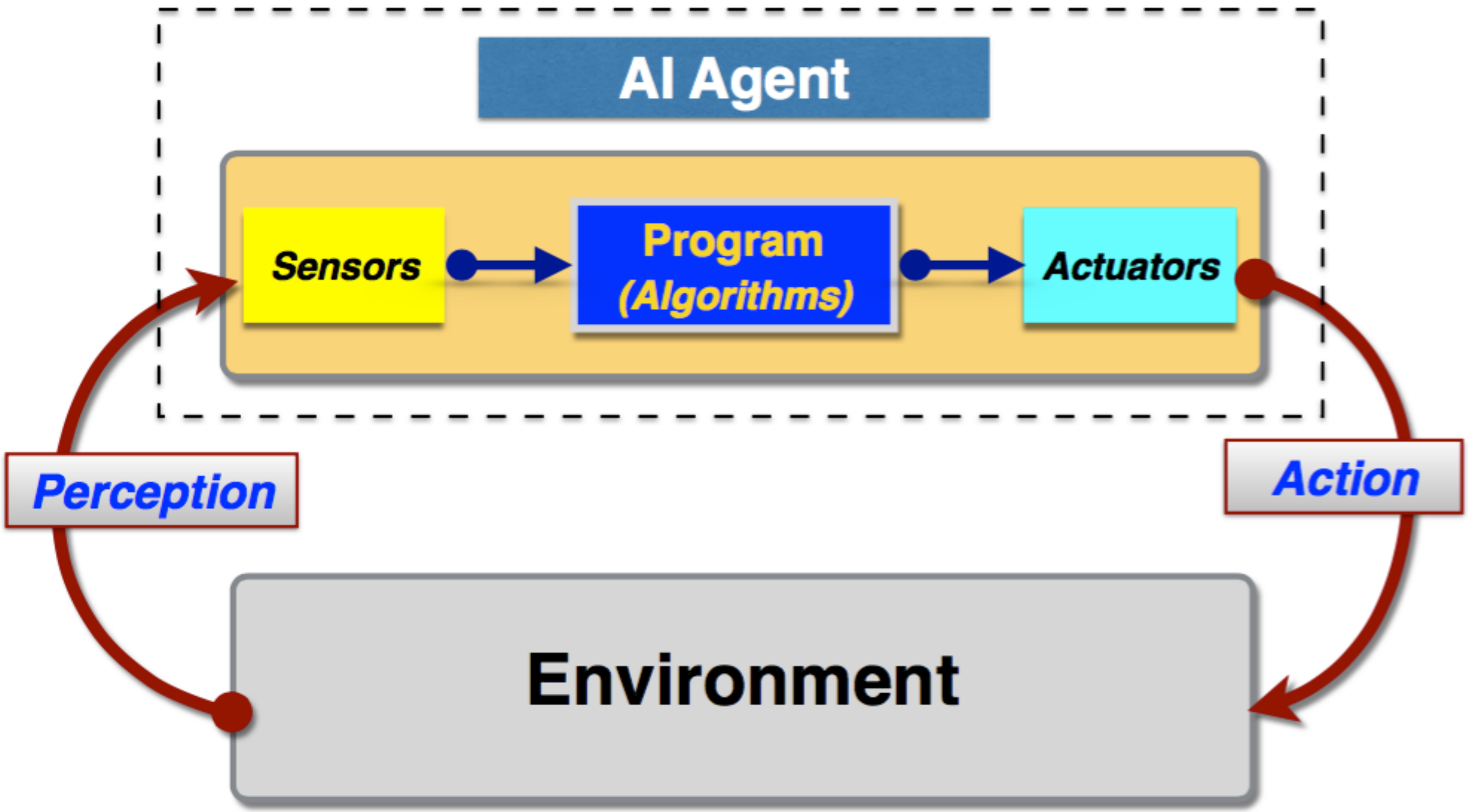
Policy => Execution => Risk Management

After Scikit-Learn … Next, ***Deep Learning.***

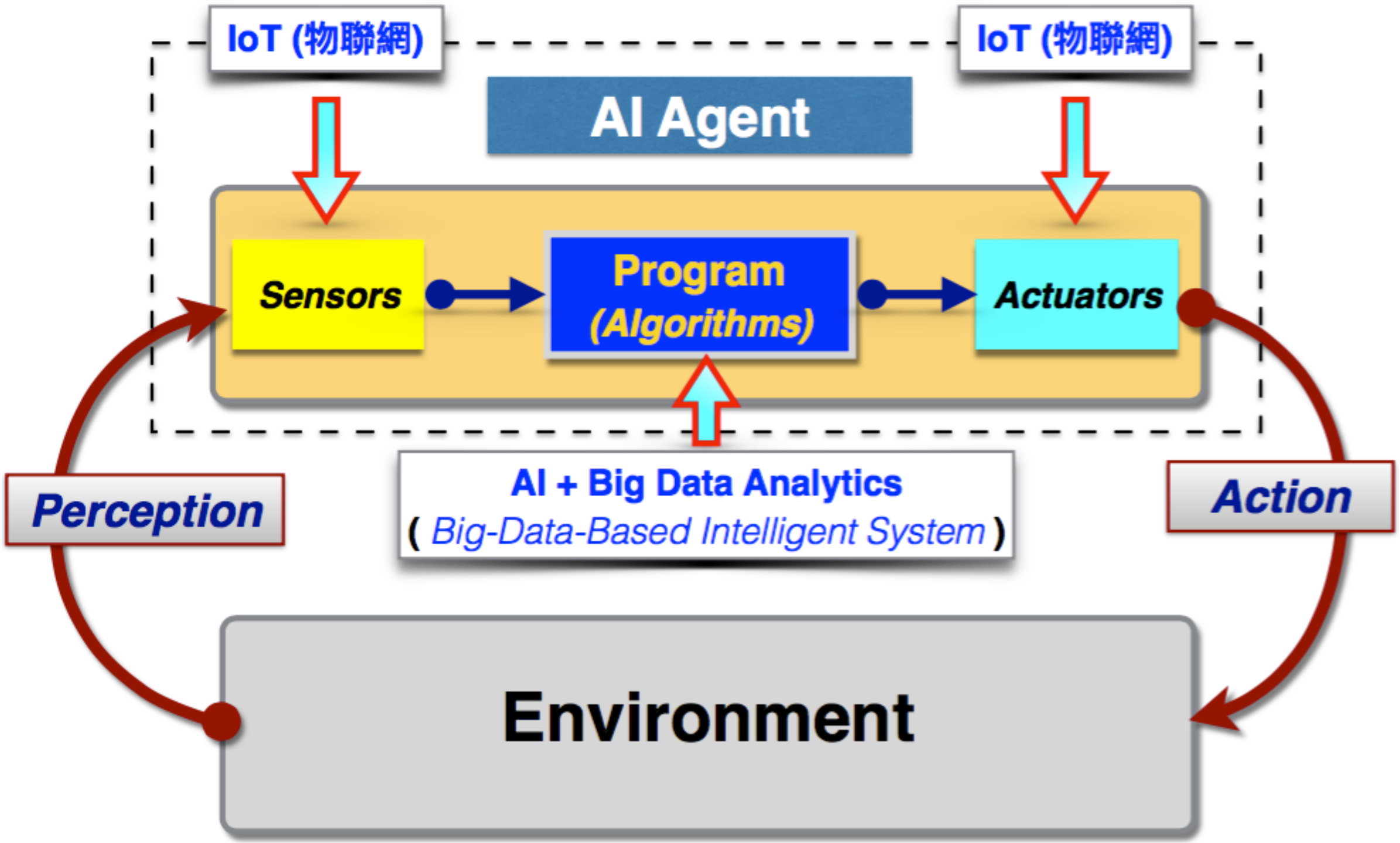


Deep Learning by Andrew Ng (吳恩達) [Full Course] — YouTube Video :
4. Drivers Behind the Rise of Deep Learning (https://youtu.be/j4-QFpTVCtM)

AI Agent


AI Agent
Sensors
Program
(Algorithms)
Actuators
Perception
Action
Environment

AI Agent for Industry 4.0 Solutions



Concluding Remarks

Q1： 誰來分析 Big Data (by ML or DL Algorithms)?

Q2： 誰來實現 Big-Data-Based Intelligent Systems ?

A : Data Scientists.

What is Data Scientist (資料科學家) ?

- 統計學家 (Statistician) vs. **資料科學家 (Data Scientist)**

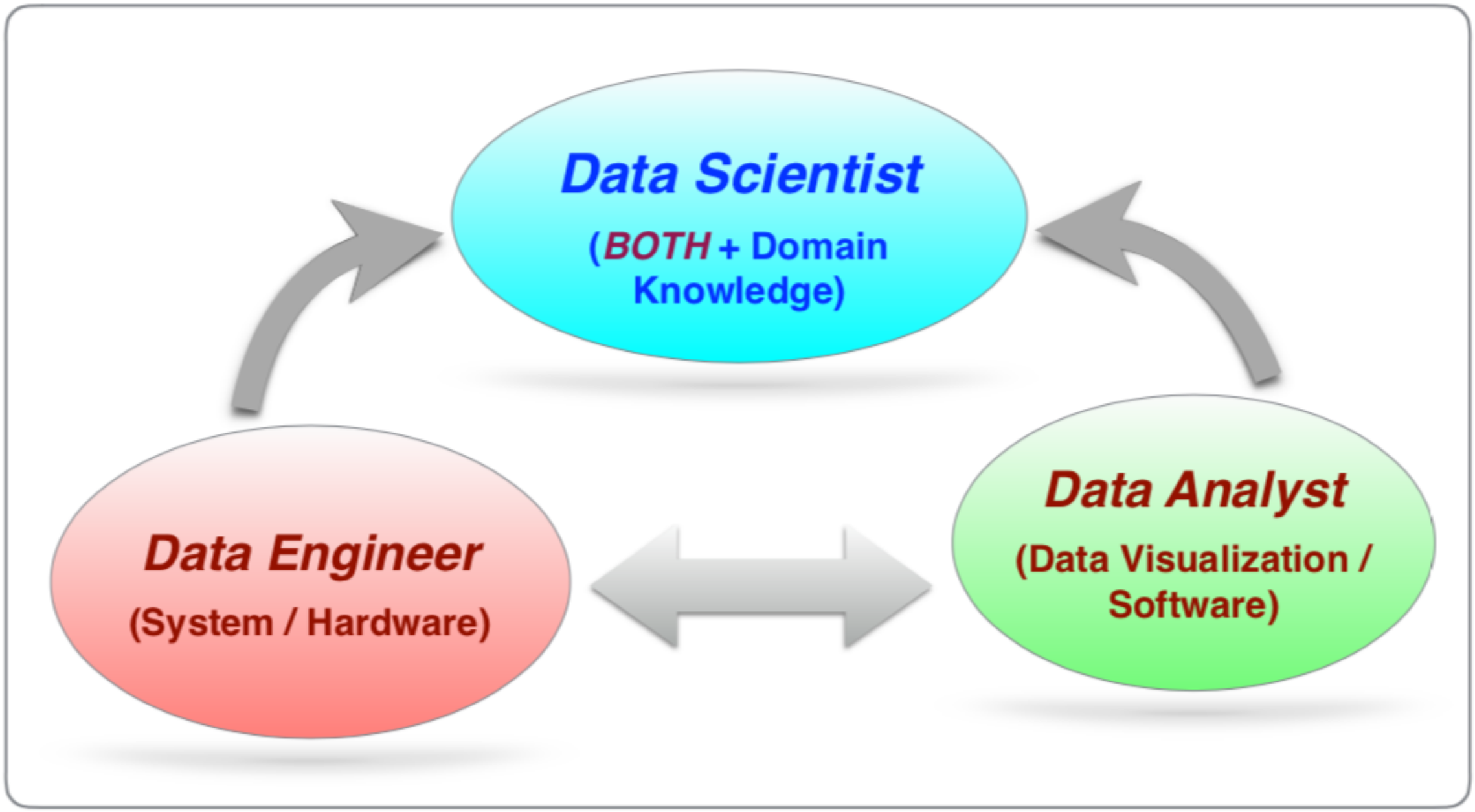
  - 統計學家通常利用三門數學學科的領域知識：

    1. 數據分析（ Data Analysis) — (SPSS, SAS, Excel…)
    2. 機率 (Probability)
    3. 統計 (Statistics)

  - 資料科學家需要的領域知識：

    — **數據分析、機率、統計、程式設計(例：R, Python, Scala…)、平行運算、資料庫、機器學習 (Machine Learning) or 資料探勘 (Data Mining) + 至少一門專業的領域知識** (機械業、製造業、銀行業、股票市場…)

What is Data Scientist ? (cont'd)

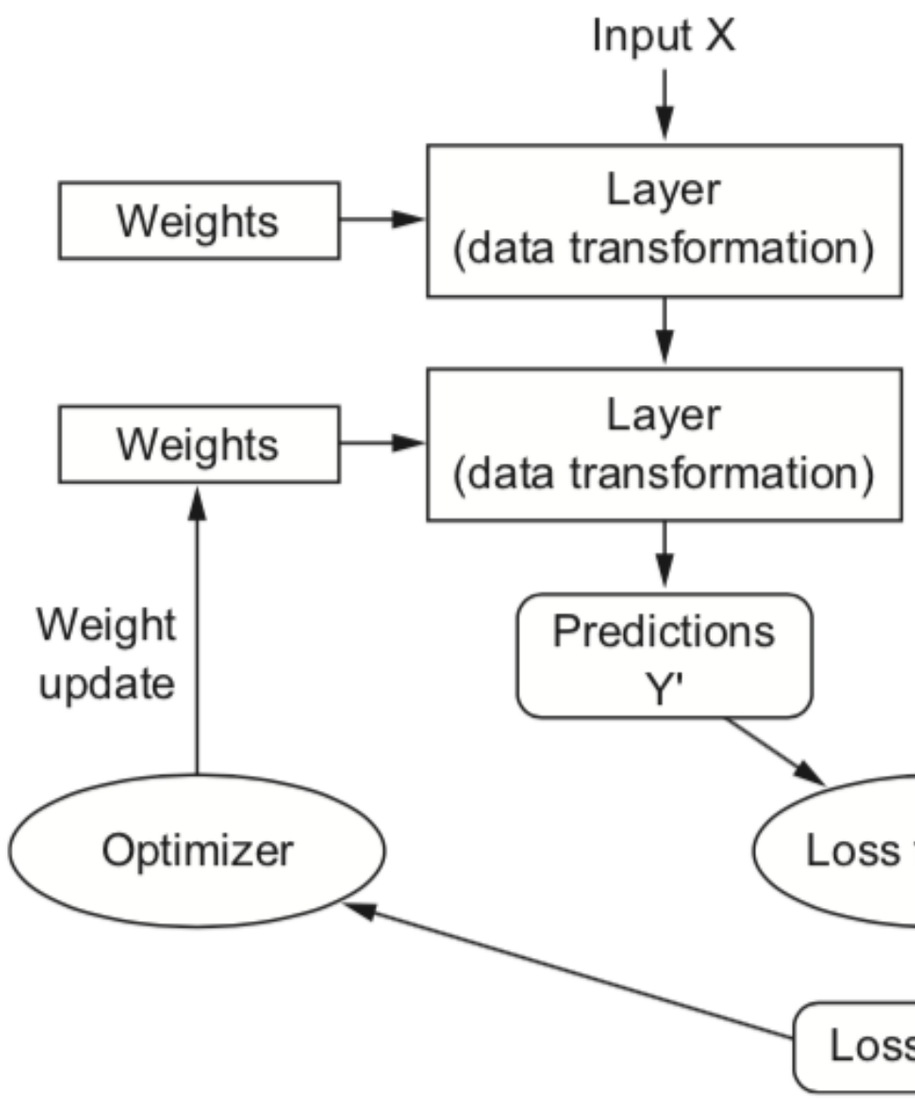


New Careers ☞ Data Scientist, Data Analyst & Data Engineer

類神經網路演算法

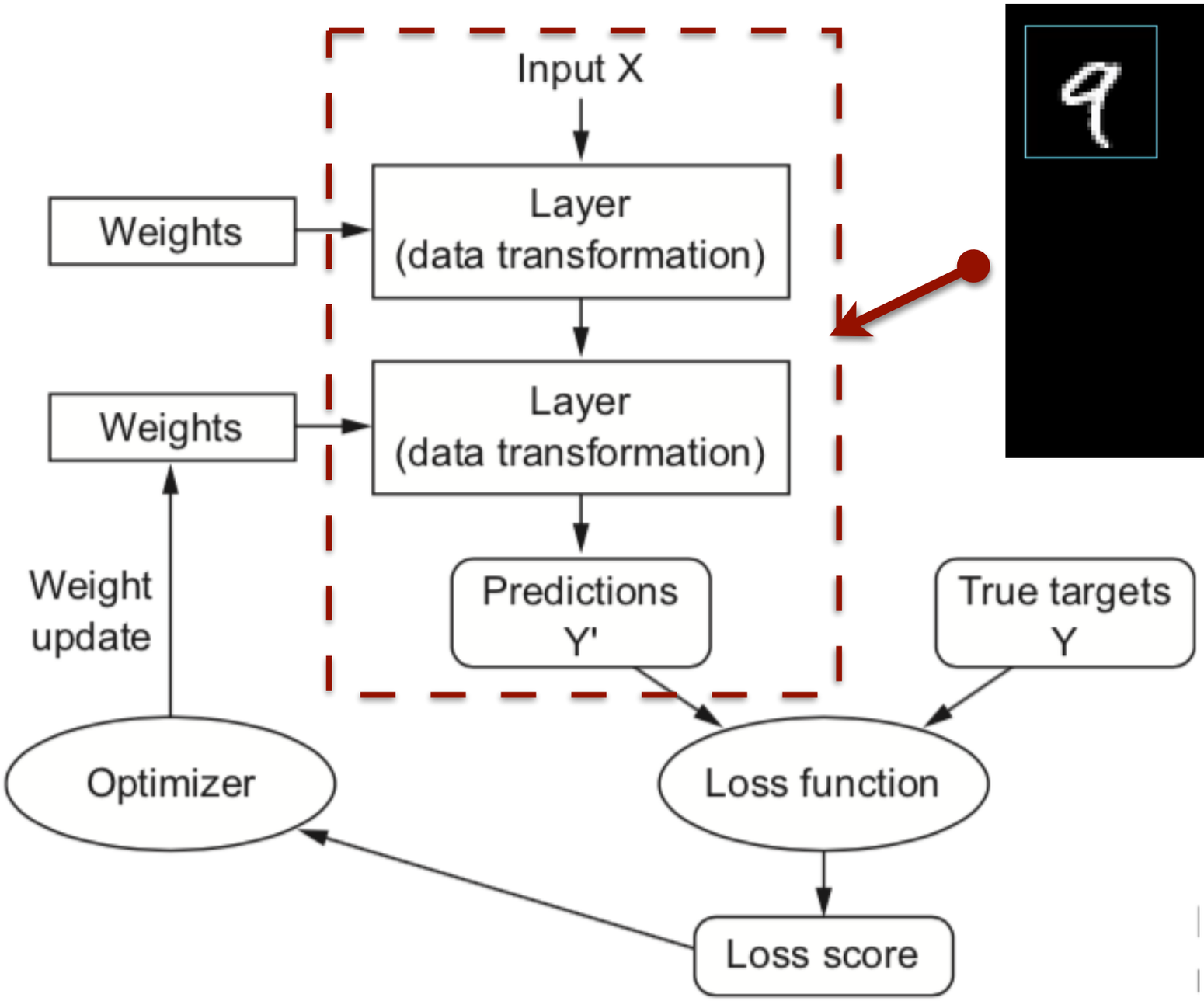
Core components of neural networks

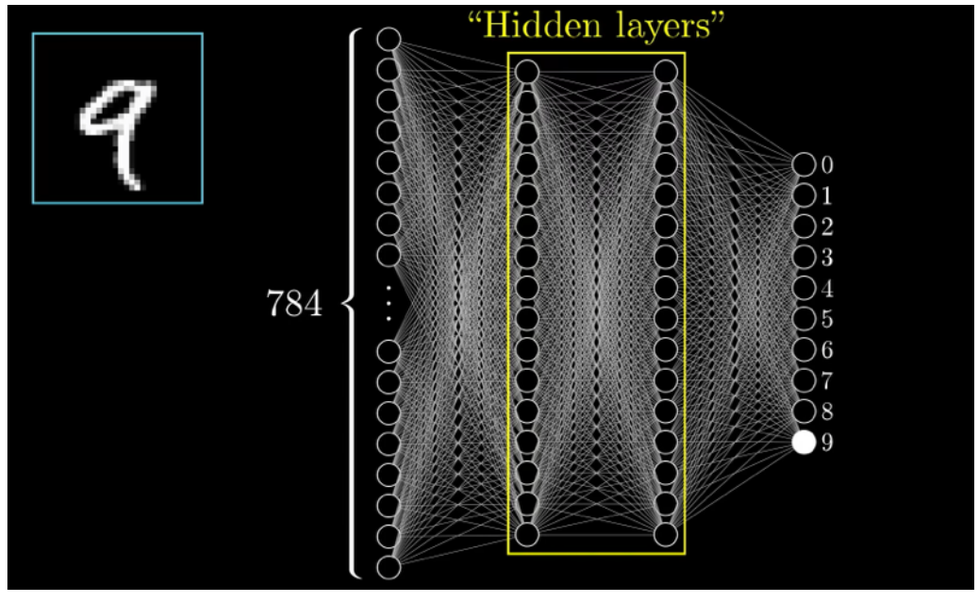
- ***Layers***, which are combined into a *network* (or *model*)
- The ***input data*** and corresponding ***targets***
- The ***loss function***, which defines the feedback signal used for learning
- The ***optimizer***, which determines how learning proceeds



Relationship between the network, layers, loss function, and optimizer

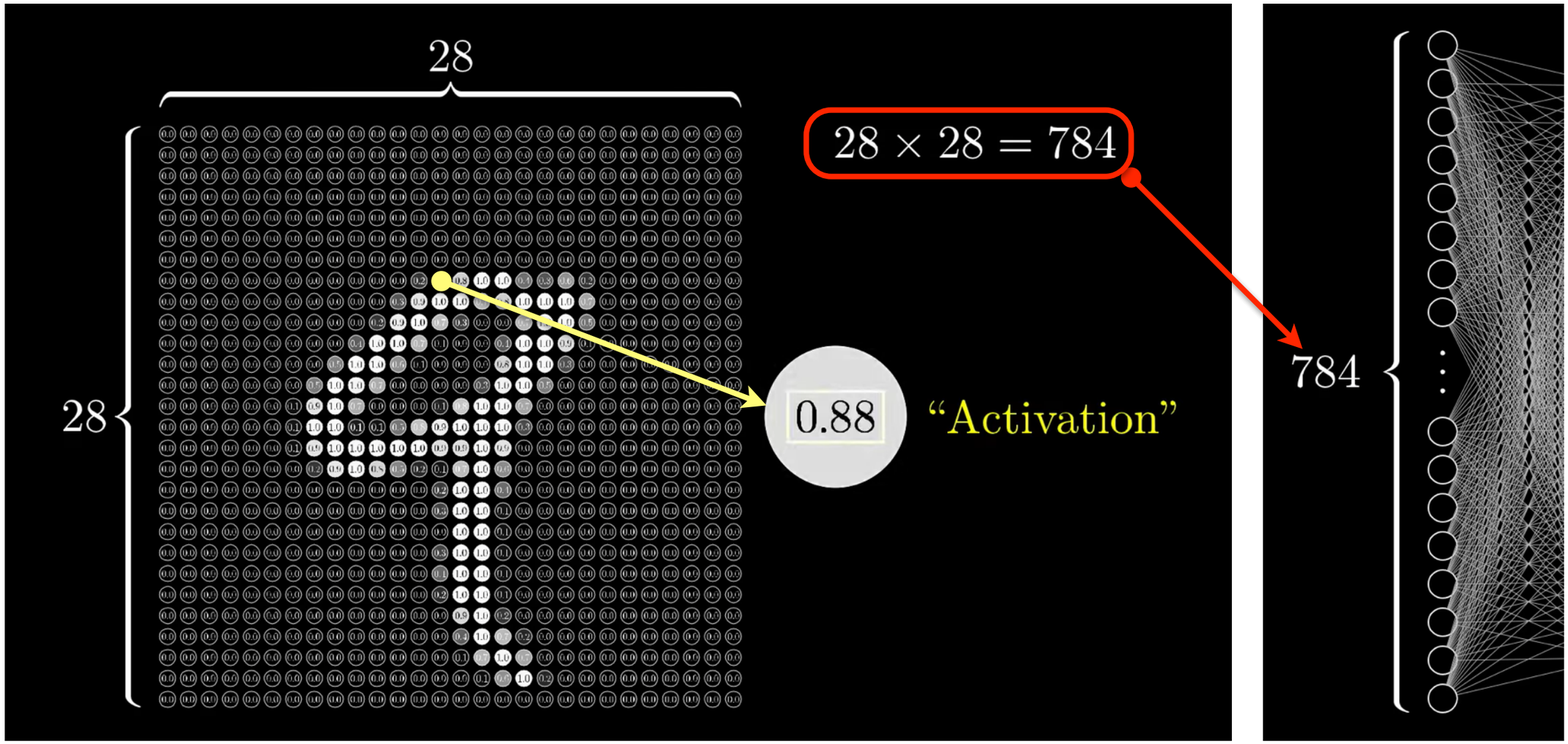
STEP 1 Forward Propagation — Prediction





2.1 *Layers* : the building blocks of deep learning

- An **Input Layer** — for example, MNIST dataset to classify grayscale images of handwritten digits (28 × 28 pixels) into their 10 categories (0 through 9).

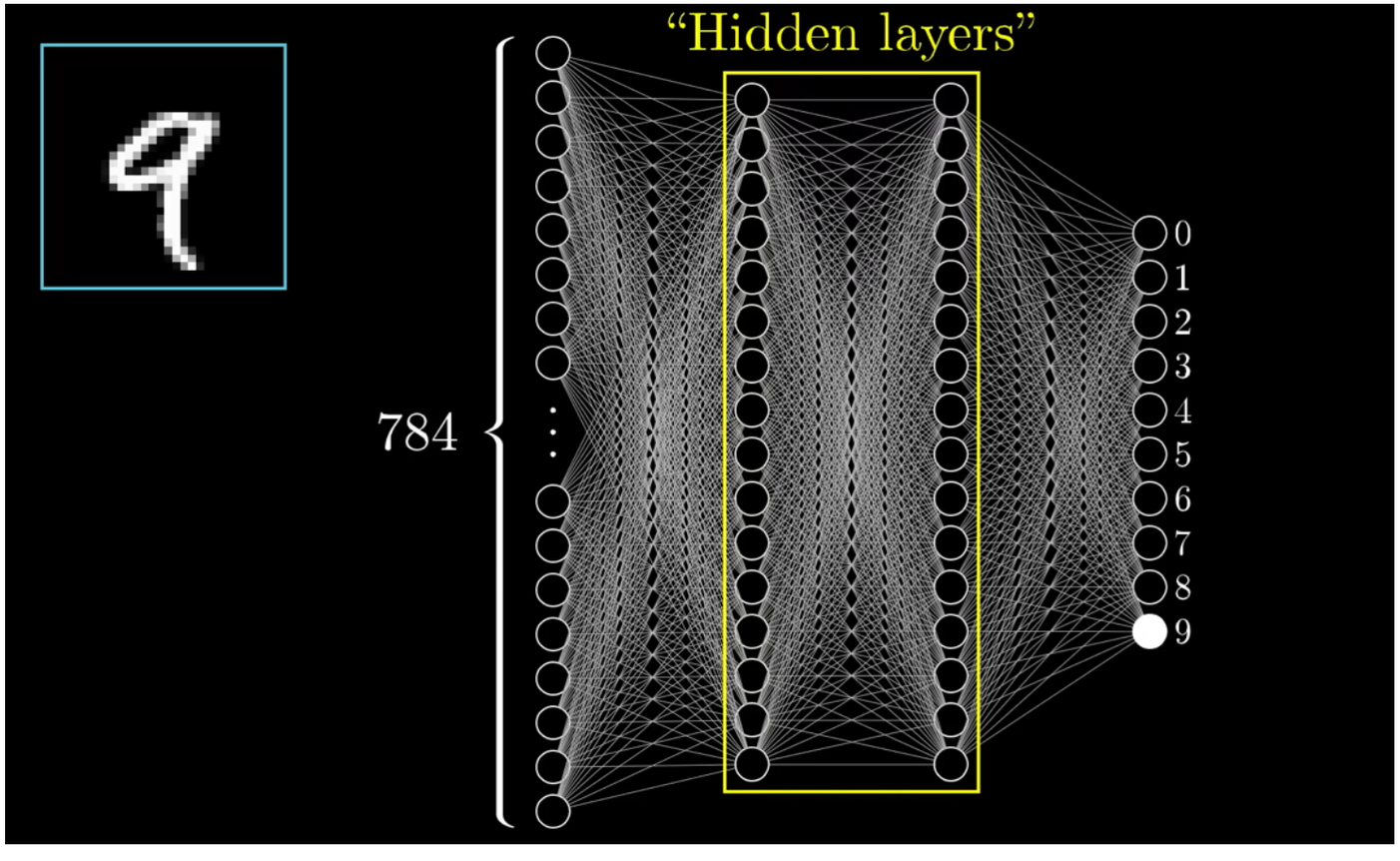


https://youtu.be/aircAruvnKk

Input Layer

2.1 *Layers* : the building blocks of deep learning (cont'd)

- A **layer** is a data-processing module that takes as input one or more tensors and that outputs one or more tensors.
- Some layers are stateless, but more frequently layers have a state: the layer's ***weights***, one or several tensors learned with **stochastic gradient descent**, which together contain the network's *knowledge*.



https://youtu.be/aircAruvnKk

2.1 *Layers* : the building blocks of deep learning (cont'd)

Different layers are appropriate for different tensor formats and different types of data processing. For instance,

- **simple vector data, stored in 2D tensors of shape** `(samples, features)`, is often processed by *densely connected* layers, also called *fully connected* or *dense* layers (the `Dense` class in Keras).
- **Sequence data, stored in 3D tensors of shape** `(samples, timesteps, features)`, is typically processed by *recurrent* layers such as an `LSTM` layer.
- **Image data, stored in 4D tensors**, is usually processed by 2D convolution layers (`Conv2D`).

2.1 *Layers* : the building blocks of deep learning (cont'd)

Layer Compatibility **(相容性) in Keras :**

- You can think of layers as the LEGO bricks of deep learning, a metaphor that is made explicit by frameworks like Keras.
- Building deep-learning models in Keras is done by clipping together compatible layers to form useful data-transformation pipe-lines.
- The notion of *layer compatibility* here refers specifically to the fact that every layer will only accept input tensors of a certain shape and will return output tensors of a certain shape.

2.1 *Layers* : the building blocks of deep learning (cont'd)

[Example] : A dense layer with 32 output units

```
from keras import layers
layer = layers.Dense(32, input_shape=(784,))
```

We're creating a layer that will only accept as input 2D tensors where the first dimension is 784 (axis 0, the batch dimension, is unspecified, and thus any value would be accepted).

- This layer will return a tensor where the first dimension has been transformed to be 32.
- *Thus this layer can only be connected to a downstream layer that expects 32-dimensional vectors as its input*. When using Keras, you don't have to worry about compatibility, because the layers you add to your models are dynamically built to match the shape of the incoming layer.

2.1 *Layers* : the building blocks of deep learning (cont'd)

[Example] : Two dense layers — each with 32 output units

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(32, input_shape=(784,)))
model.add(layers.Dense(32))
```

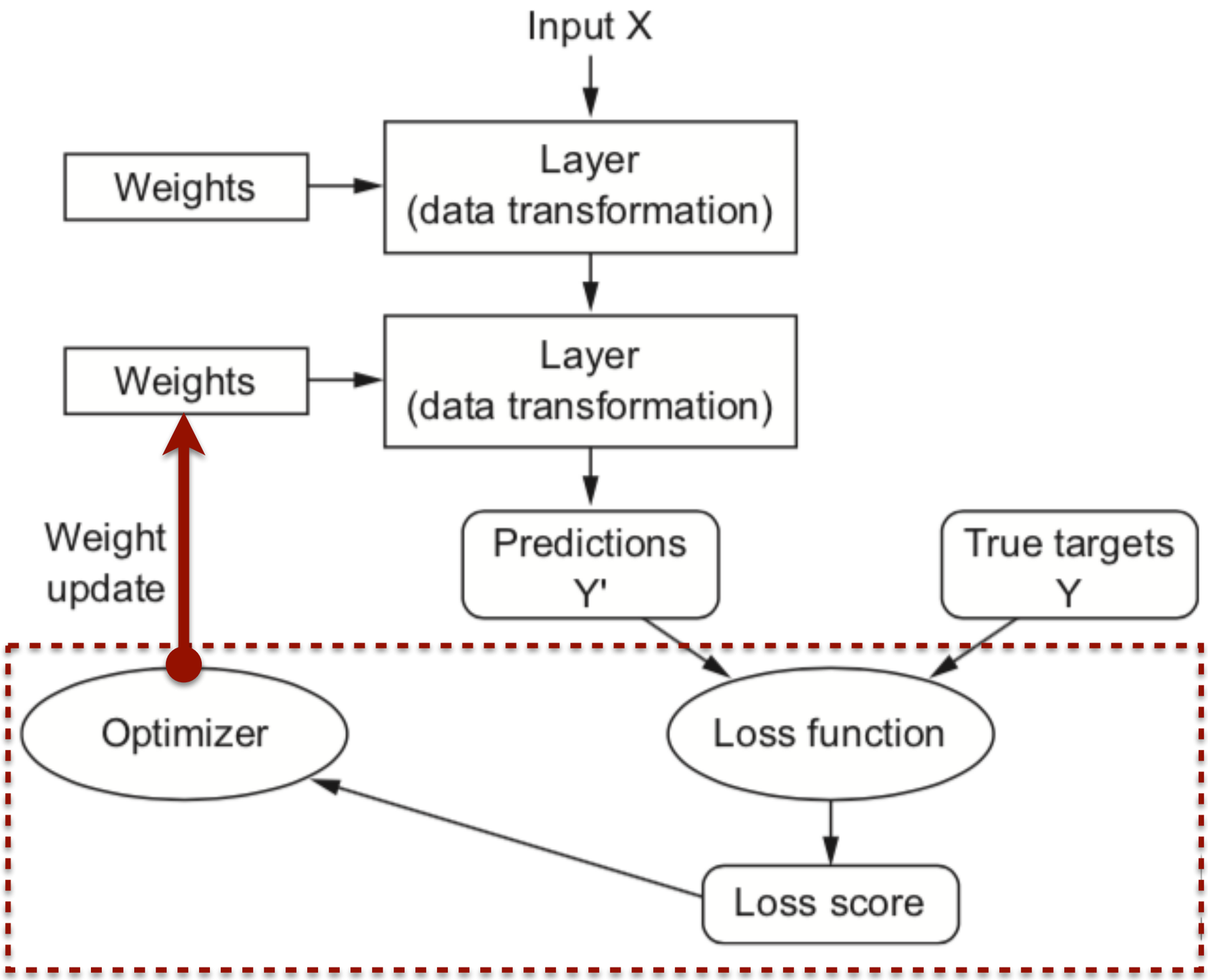
The second layer didn't receive an input shape argument—*instead, it automatically inferred its input shape as being the output shape of the layer that came before*.

2.1 *Layers* : the building blocks of deep learning (cont'd)

[NOTE] : Models — Networks of Layers

- A deep-learning model is a directed, acyclic graph of layers.
- The topology of a network defines a *hypothesis space*.
- By choosing a network topology, you constrain your *space of possibilities* (hypothesis space) to a specific series of tensor operations, mapping input data to output data.
- What you'll then be searching for is a good set of values for the weight tensors involved in these tensor operations.
- **Picking the right network architecture is more an art than a science**; and although there are some best practices and principles you can rely on, only practice can help you become a proper neural-network architect.

STEP 2 Back-propagation — Optimization

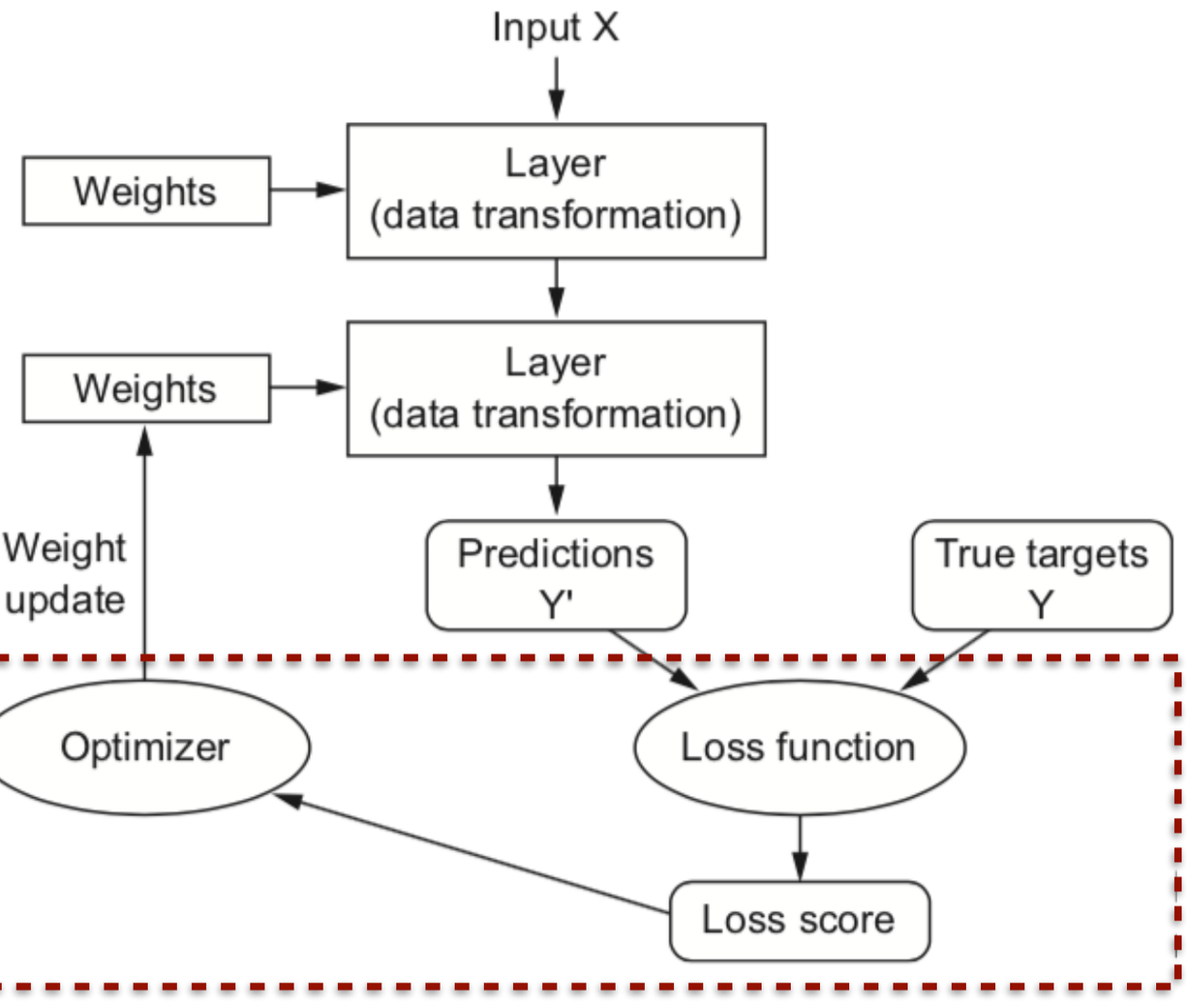


2.2 *Loss Functions & Optimizers*

Loss Functions & Optimizers : keys to configuring the learning process

Once the network architecture is defined, you still have to choose two more things:

- ***Loss function (objective function)*** — The quantity that will be minimized during training. It represents a measure of success for the task at hand.
- ***Optimizer*** — Determines how the network will be updated based on the loss function. It implements a specific variant of stochastic gradient descent (SGD).



2.2 *Loss Functions & Optimizers* (cont'd)

About *Loss Functions (Objective Functions)* …

- A neural network that has multiple outputs may have multiple loss functions (one per output).
- But the gradient-descent process must be based on a *single* scalar loss value; so, *for multi-loss networks, all losses are combined (via averaging) into a single scalar quantity*.
- Choosing the right objective function for the right problem is extremely important: *your network will take any shortcut it can, to minimize the loss*; so if the objective doesn't fully correlate with success for the task at hand, your network will end up doing things you may not have wanted.

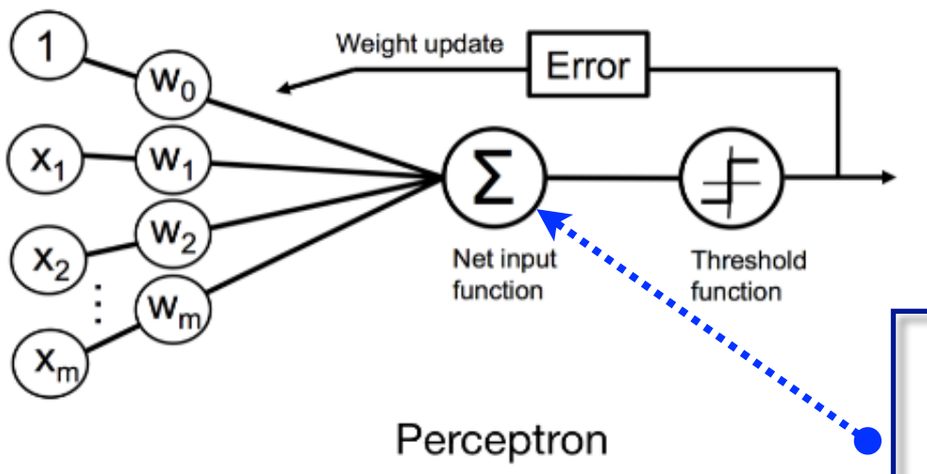
About *Loss Functions (Objective Functions)* …

- A neural network that has multiple outputs may have multiple loss functions (one per output).
- But the gradient-descent process must be based on a *single* scalar loss value; so, *for multi-loss networks, all losses are combined (via averaging) into a single scalar quantity*.
- Choosing the right objective function for the right problem is extremely important: *your network will take any shortcut it can, to minimize the loss*; so if the objective doesn't fully correlate with success for the task at hand, your network will end up doing things you may not have wanted.

2.2 *Loss Functions & Optimizers* (cont'd)

Simple Neural Networks — Perceptron *vs.* Logistic Regression (Binary Classifiers)

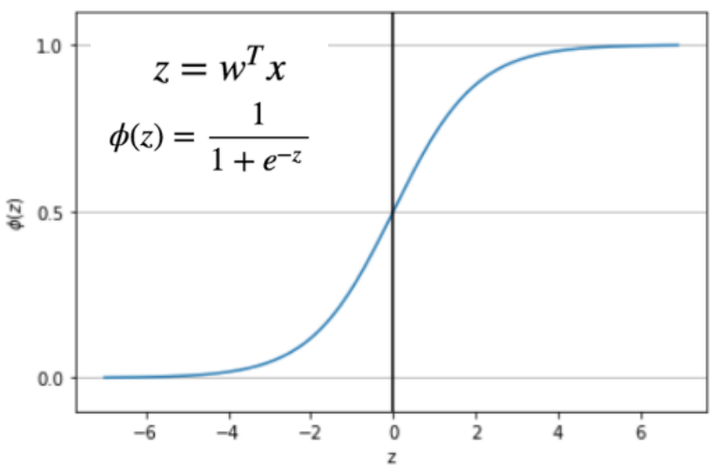
— Activation functions : Step function *vs. Sigmoid function* (Logistic function)

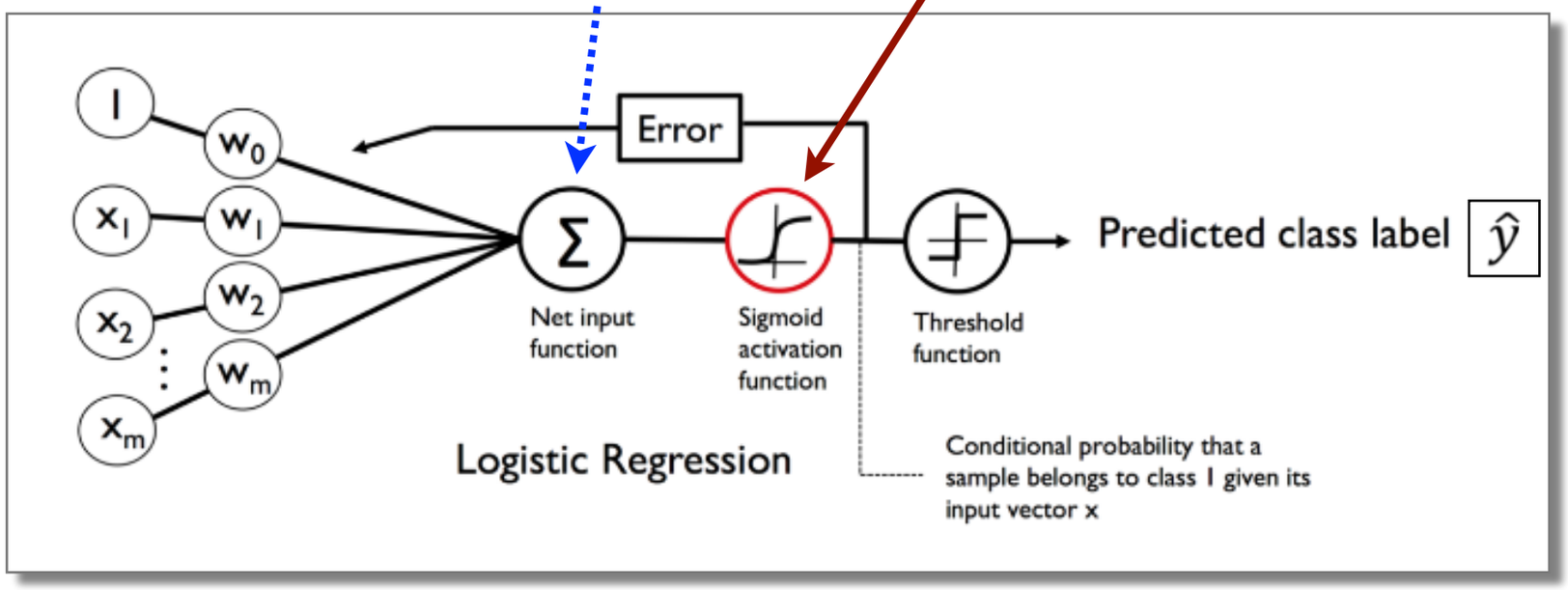


***hyperplane* (a linear divisor)**

$$\hat{y} = \phi(z)$$

$$z = w^T x = \sum_{j=0}^{M} w_j * x_{ij}$$



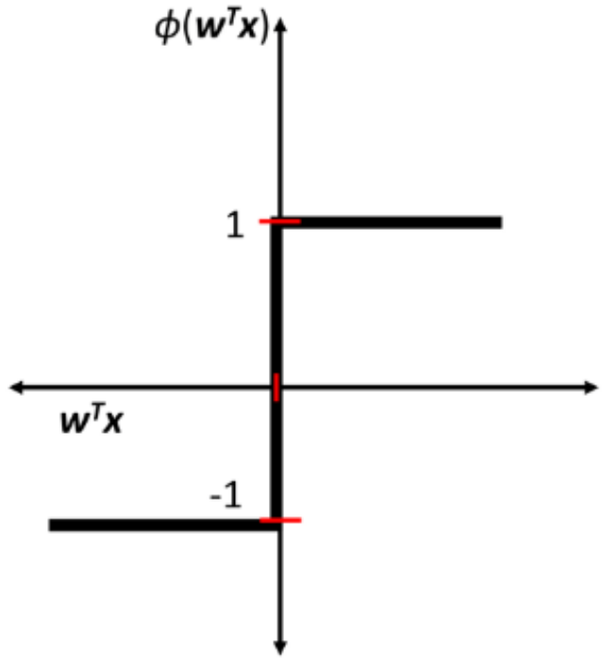


2.2 *Loss Functions & Optimizers* (cont'd)

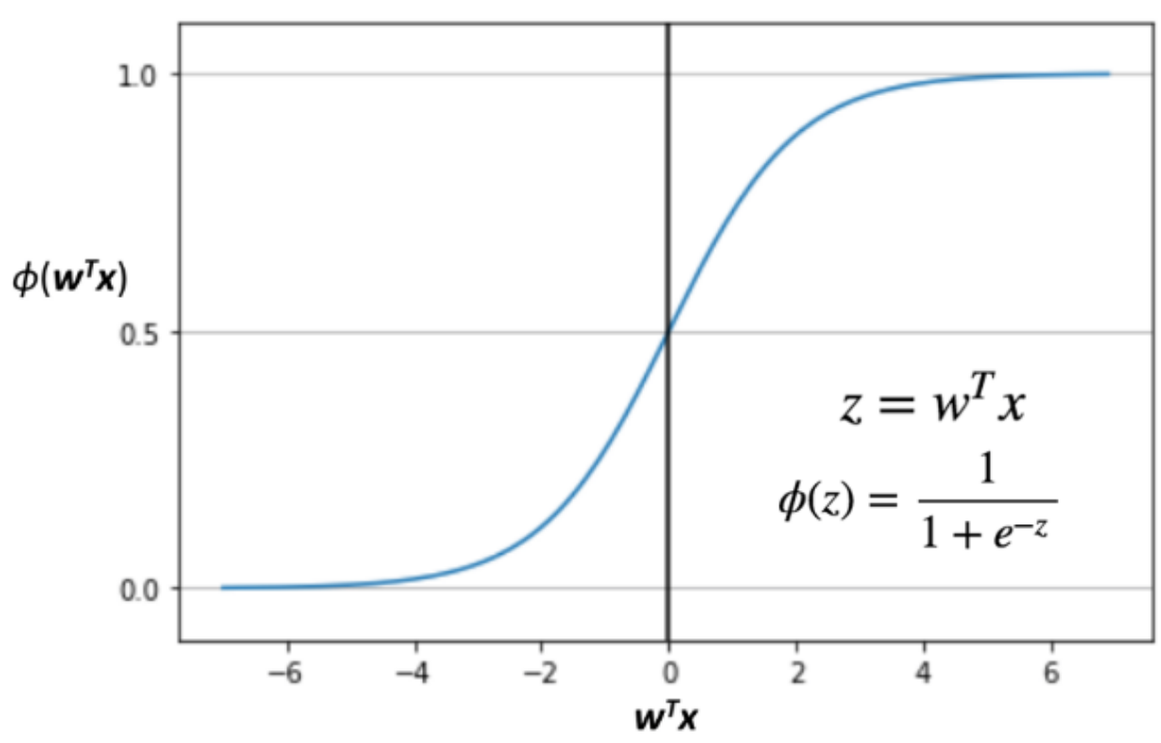
Simple Neural Networks — Perceptron *vs.* Logistic Regression (Binary Classifiers)

— **Activation functions** : Step function *vs. Sigmoid function* (Logistic function)

Perceptron



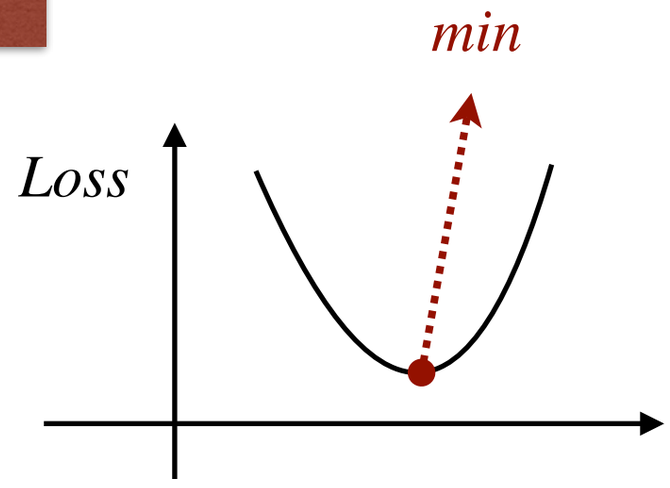
Logistic Regression



2.2 *Loss Functions & Optimizers* (cont'd)

Loss Function for Multi-output Neural Networks

$$Loss = \frac{1}{2}\{y_i - \hat{y}_i\}^2$$



— *loss function for a single training sample*

Cost Function for Multi-output Neural Networks

$$Cost\ (W) = RSS(W) = \sum_{i=1}^{N}\{y_i - \hat{y}_i\}^2 = \sum_{i=1}^{N}\left\{y_i - \sum_{j=0}^{M} w_j\, x_{ij}\right\}^2$$

— *cost function for the entire training dataset*

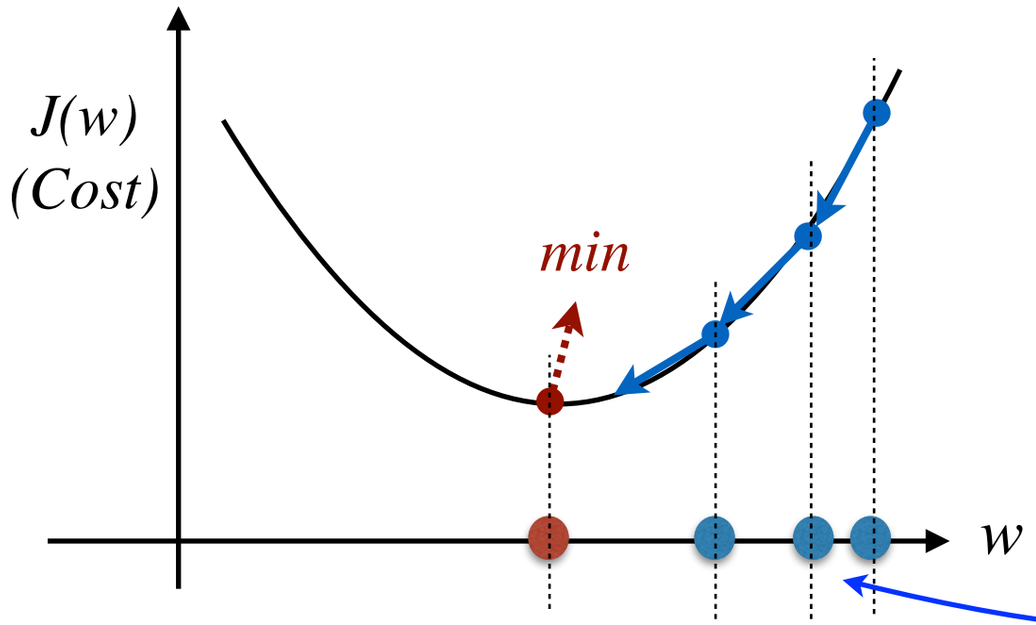
[NOTE] : RSS — Residual Sum of Squares

2.2 *Loss Functions & Optimizers* (cont'd)

Cost Function for Multi-output Neural Networks

$$Cost\ (W) = RSS(W) = \sum_{i=1}^{N} \{y_i - \hat{y}_i\}^2 = \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2$$

Gradient Descent for the Optimizer



Update ***w*** with the following iteration :

Iterate {

w := w — eta * (dJ / dw)

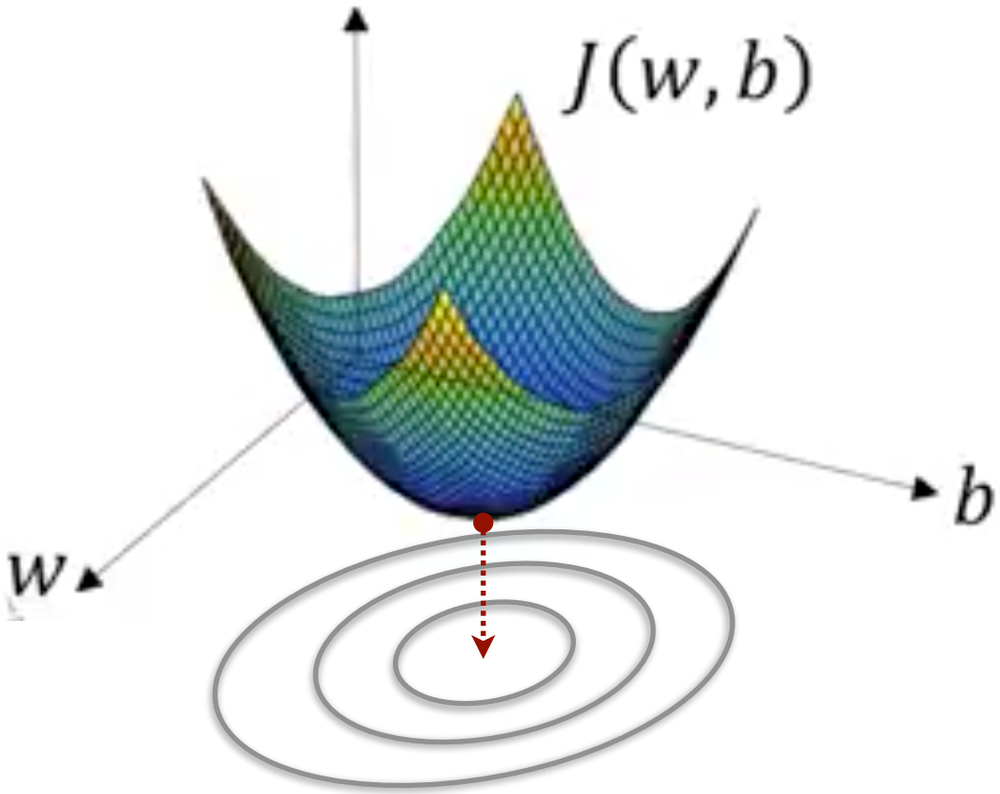
}

[NOTE] : *eta — learning rate*

$\triangle w = -\ eta * (dJ / dw)$

2.3 *Stochastic Gradient Descent*

Gradient Descent for the Optimizer



Update w with the following iteration :

```
Iterate {
        w := w — eta * (dJ / dw)
        b := b — eta * (dJ / db)
      }
```

[NOTE] :

- " *d* " — partial derivative
- *eta* — *learning rate*

1. Mini-Batch Gradient Descent

2. Stochastic Gradient Descent

Ref : Andrew Ng, "57. Understanding Mini-Batch Gradient Descent"

https://youtu.be/-_4Zi8fCZO4

2.4 *Regularization*

(Ref : "05.06-Linear-Regression" from the Python Data Science Handbook, by Jake VanderPlas)

— Traditional ML algorithms : ***Bias-Variance*** Tradeoff

=> *Selecting the best model !*

— More complicated algorithms (*high variance*) lead to ***over-fitting*****.**
Accordingly, deep neural networks tend to be over-fitting models.

— For example, if we choose too many hidden layers/neurons,
there typically exists the over-fitting behavior in the fully connected networks.

— We could reduce explicitly the overfitting in the model with **regularizations**

1. by ***dropout*** the connections/nodes
2. by ***penalizing*** *large values of the model parameters*.

- **Ridge regression (L_2 regularization)**
- **Lasso regression (L_1 regularization)**

2.4 *Regularization* (cont'd)

[Ref. "**A Complete Tutorial on Ridge and Lasso Regression in Python**," AARSHAY JAIN, JANUARY 28, 2016
https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/]

- **Linear Regression**

$$\hat{y}_i = \sum_{j=0}^{M} w_j * x_{ij}$$

$$Cost\ (W) = RSS(W) = \sum_{i=1}^{N} \{y_i - \hat{y}_i\}^2 = \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2$$

Q : Why Penalize the Magnitude of Coefficients?
=> *Over-fitting* !!

- **Ridge regression**
  (L_2 regularization)

$$Cost\ (W) = RSS(W) + \lambda * (sum\ of\ squares\ of\ weights)$$

$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} w_j^2$$

- **Lasso regression**
  (L_1 regularization)

$$Cost\ (W) = RSS(W) + \lambda * (sum\ of\ absolute\ value\ of\ weights)$$

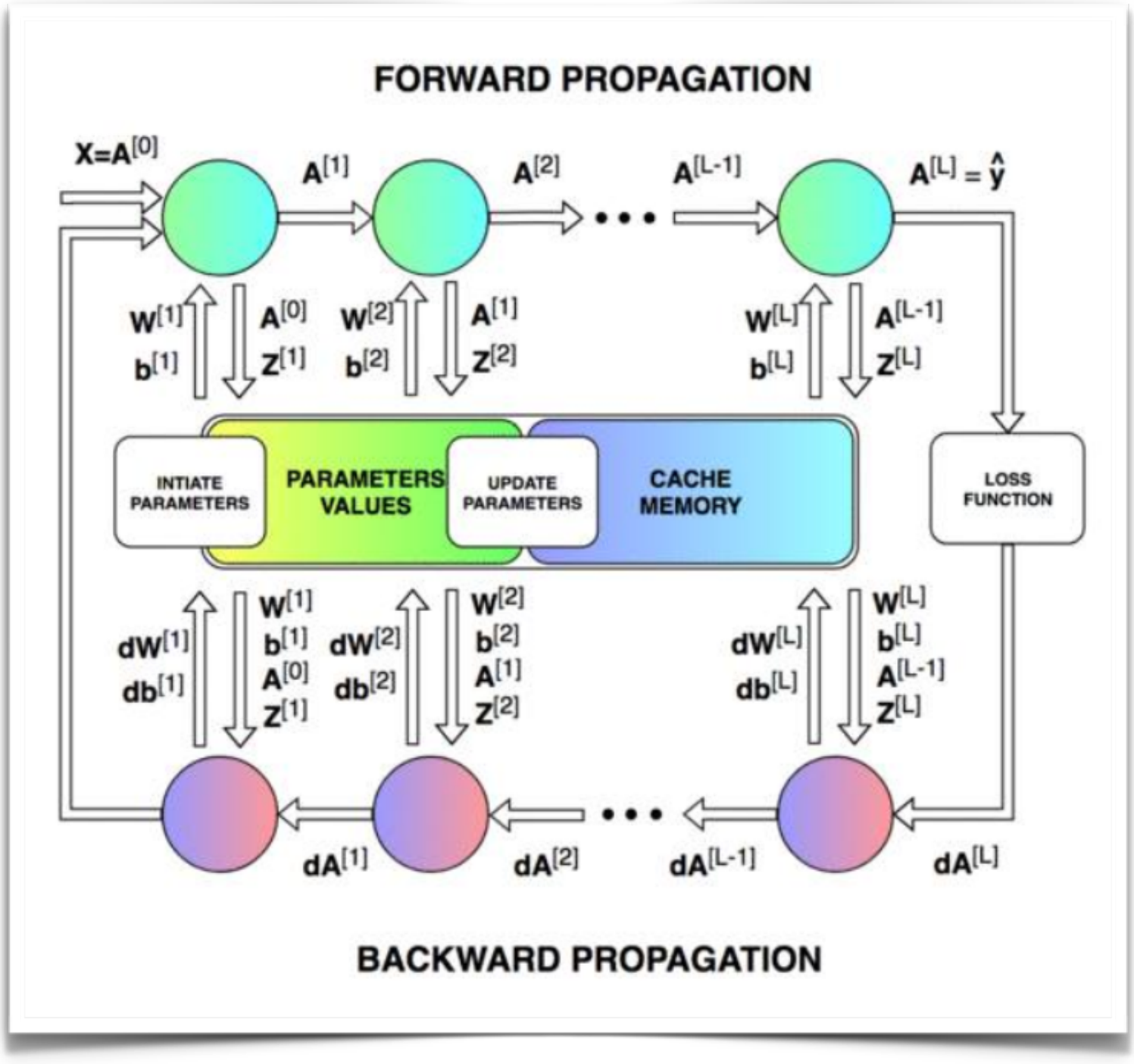
$$= \sum_{i=1}^{N} \left\{ y_i - \sum_{j=0}^{M} w_j\, x_{ij} \right\}^2 + \lambda \sum_{j=0}^{M} |w_j|$$

usually for sparse coefficients

Deep Learning

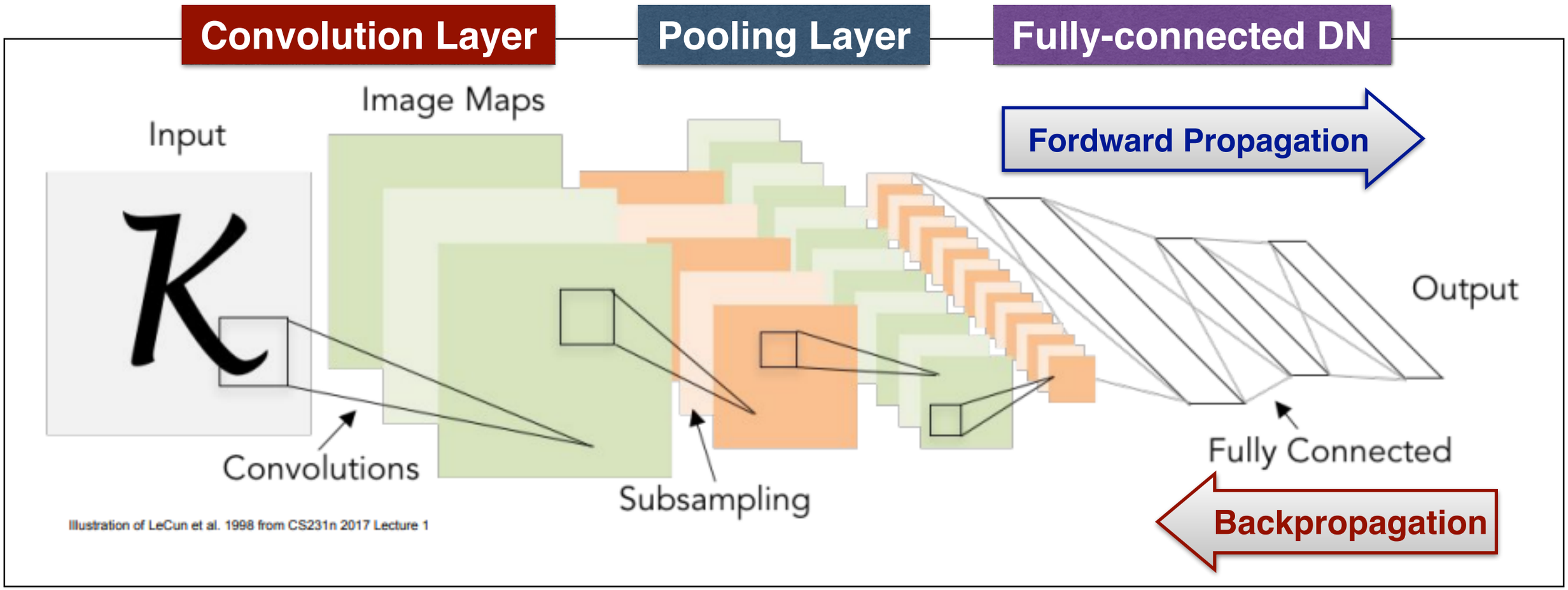
with **Big Data**

Fully-connected Deep Networks (FCDN)



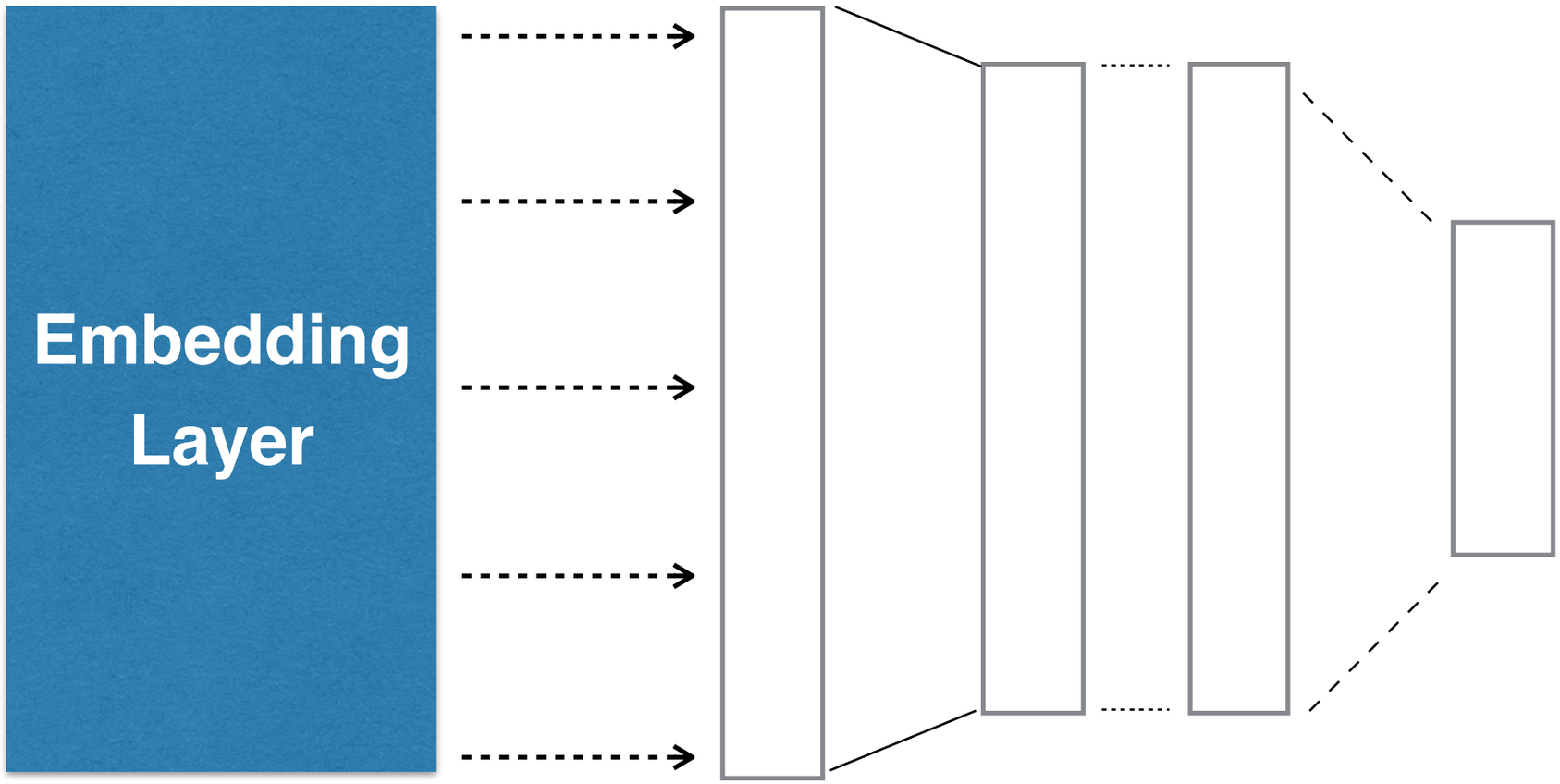
Ref. : Towards Data Science — **Piotr Skalski**, "***Let's code a Neural Network in plain NumPy***" (https://buff.ly/2EiP2mz)

ConvNets (CNN)



李飛飛教授：Convolutional Neural Networks (教學投影片)
(http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf)

Recurrent Neural Networks (RNN)

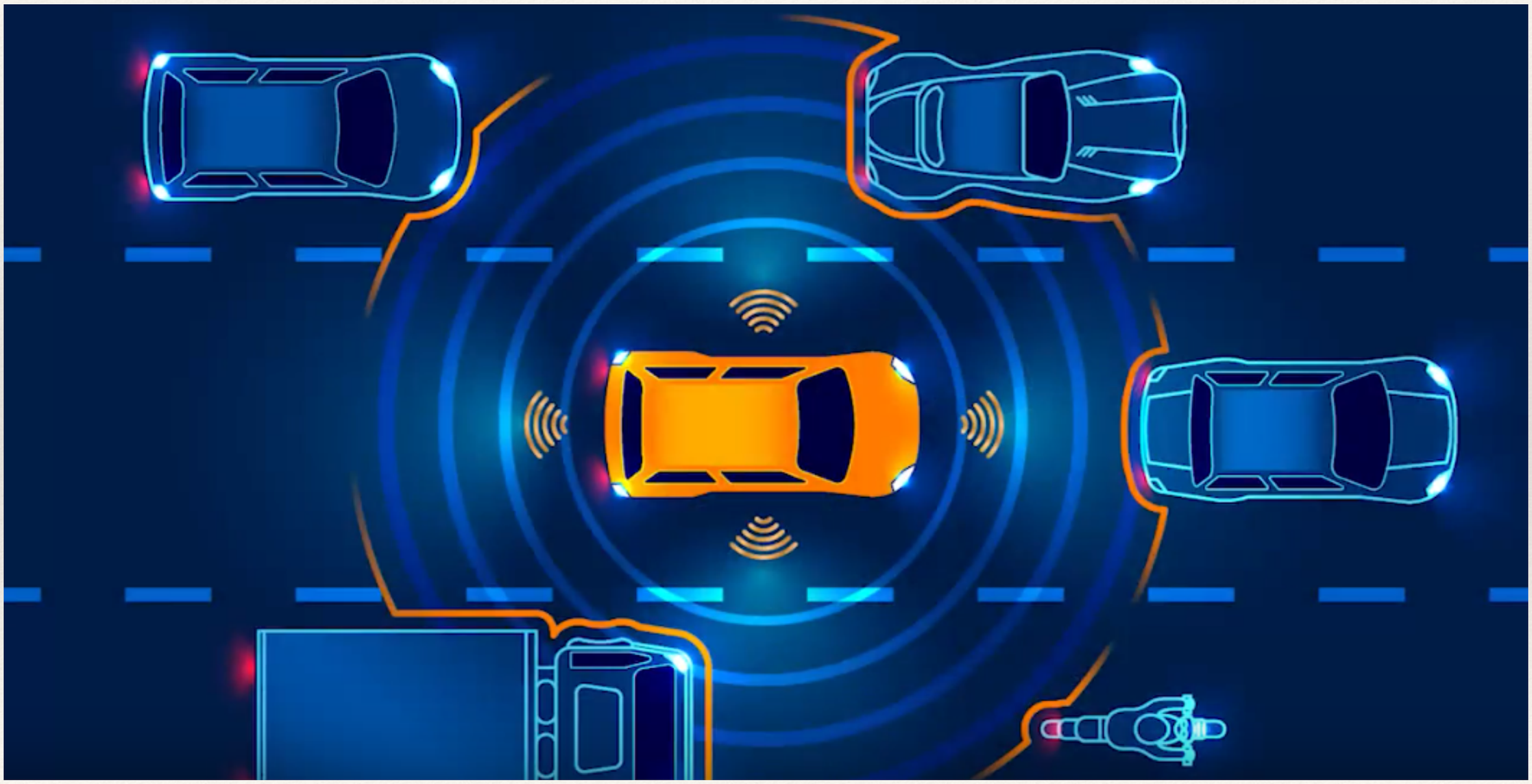


Recurrent Algorithm for 1D Dataset + **Fully-connected NN**

Machine Learning Applications

A Multi-Agent System (多代理人系統)

- 機聯網、車聯網、… ☞ **邊緣運算 => AIoT (智慧物聯網)**



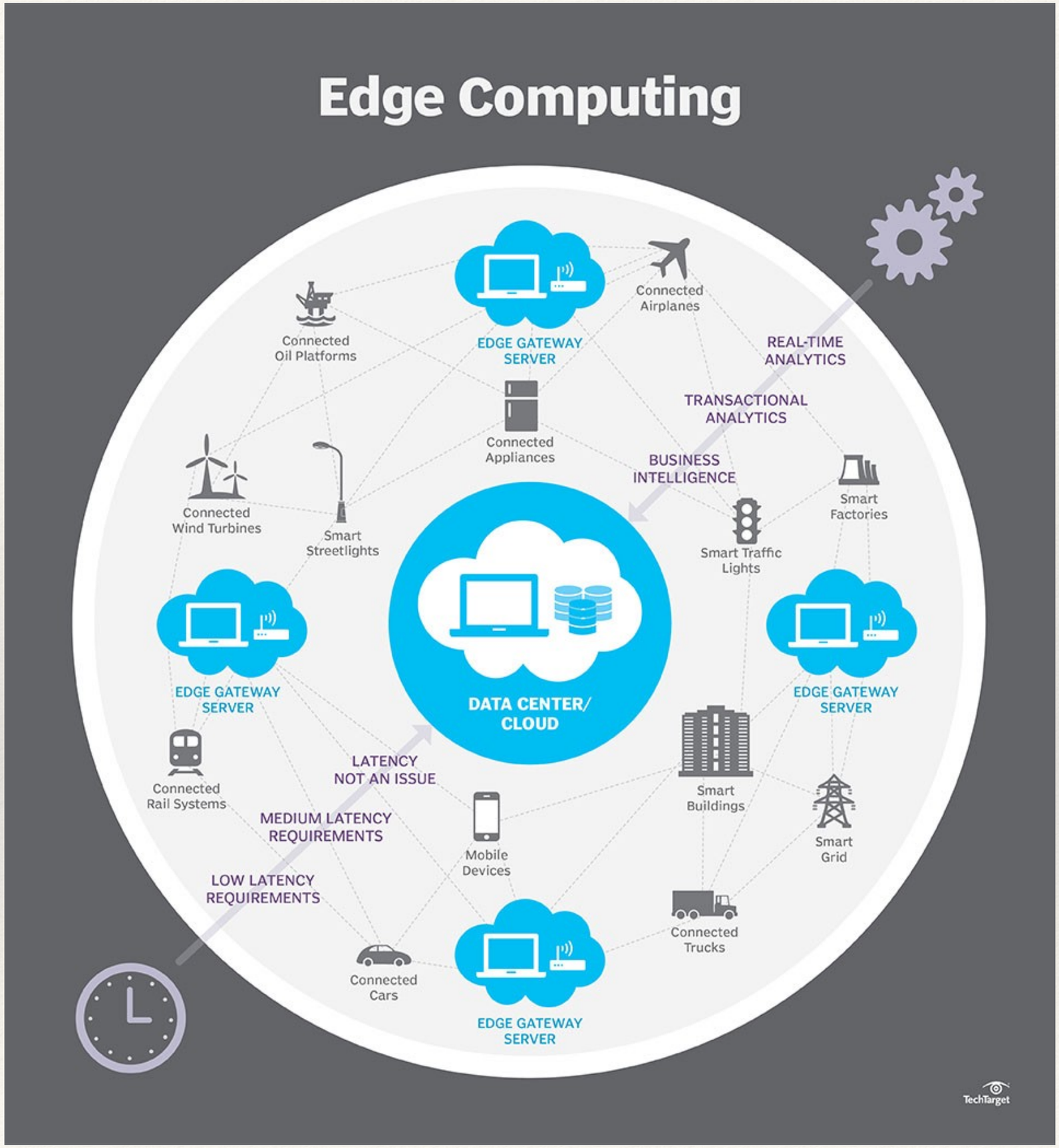
Edge Computing (邊緣運算) + AIoT (AI + 物聯網)

https://www.ithome.com.tw/news/114625

邊緣運算的定義與應用

- **邊緣運算 (Edge computing)**，又譯為邊緣計算，是一種分散式運算的架構，將應用程式、數據資料與服務的運算，由網路中心節點，移往網路邏輯上的邊緣節點來處理。
- 邊緣節點更接近於用戶終端裝置可以加快資料的處理與傳送速度減少延遲。

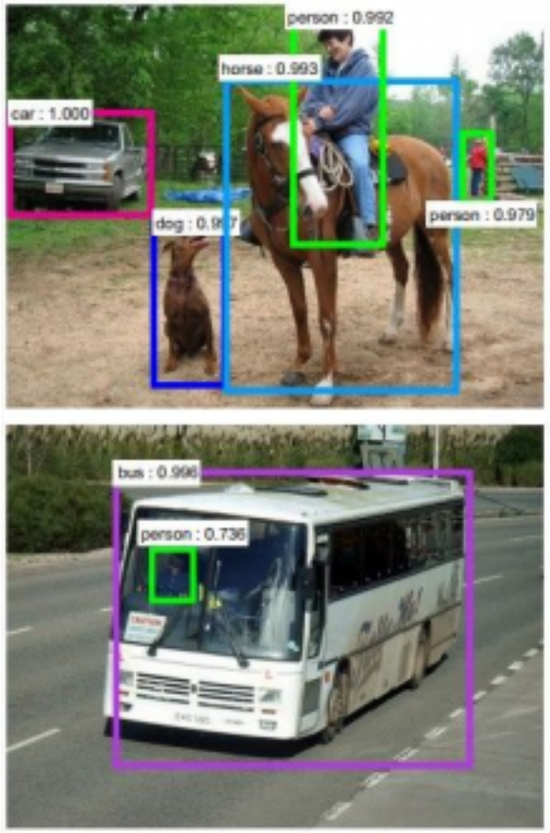
— *from Wikipedia*

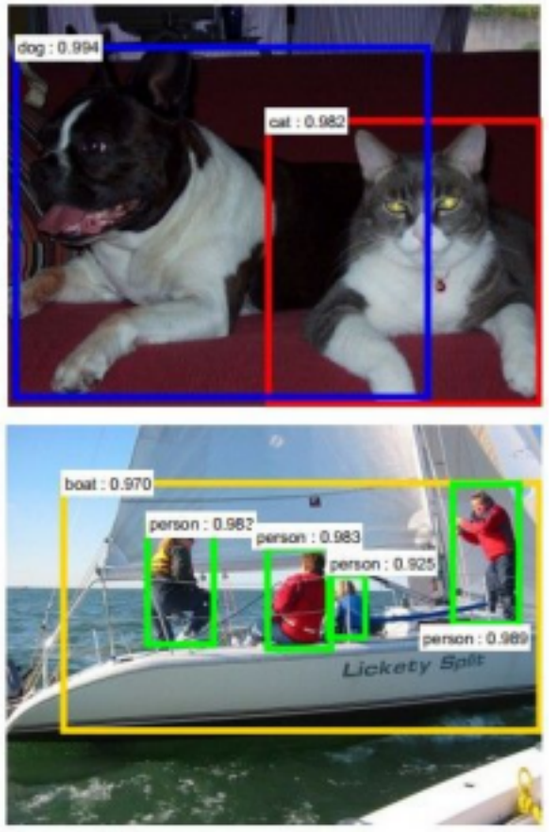


Deep Learning Applications

ConvNets Applications

Detection

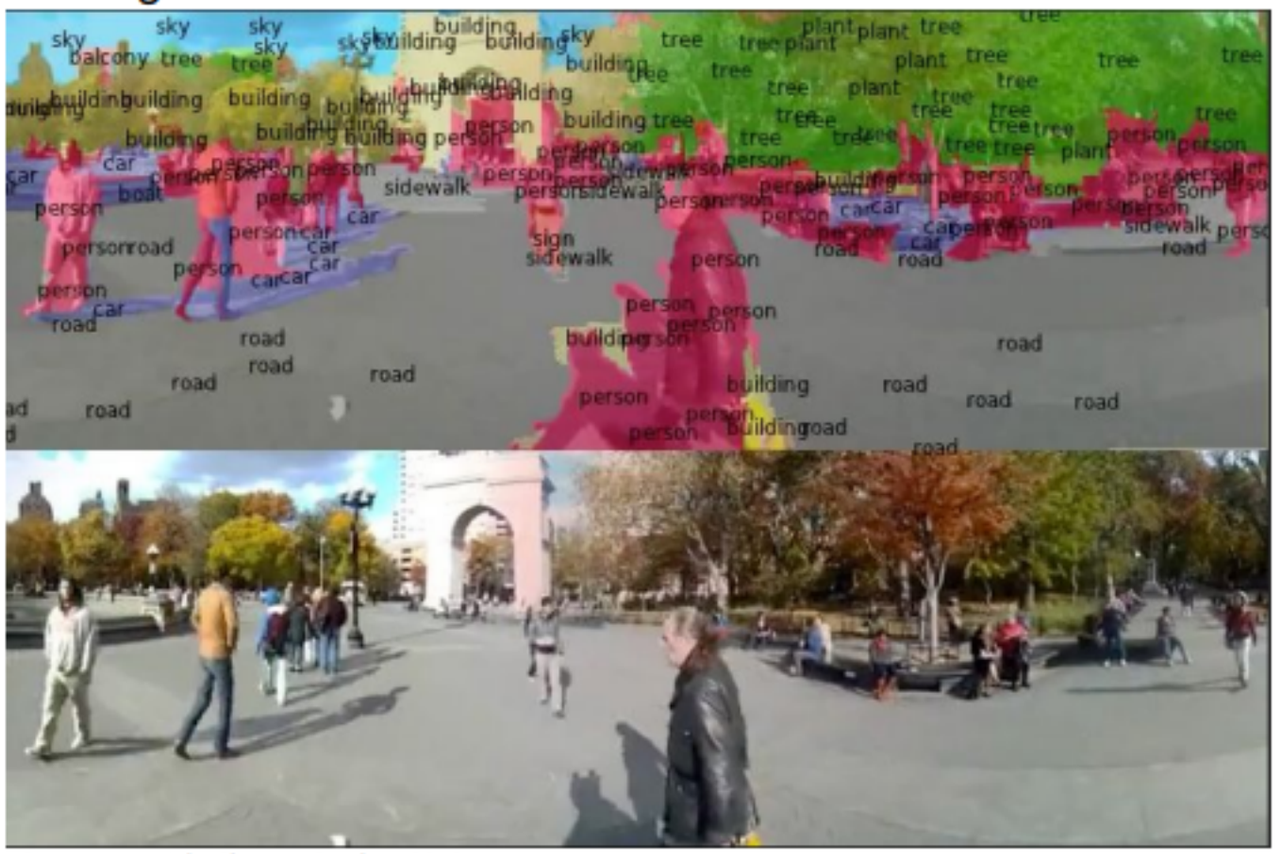




Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012.
Reproduced with permission.

[Farabet et al., 2012]

李飛飛教授：Convolutional Neural Networks (教學投影片)
(http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf)

ConvNets Applications

Photo by Lane McIntosh. Copyright CS231n 2017.

self-driving cars

NVIDIA Tesla line

(these are the GPUs on rye01.stanford.edu)

Note that for embedded systems a typical setup would involve NVIDIA Tegras, with integrated GPU and ARM-based CPU cores.

李飛飛教授：Convolutional Neural Networks (教學投影片)

(http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf)