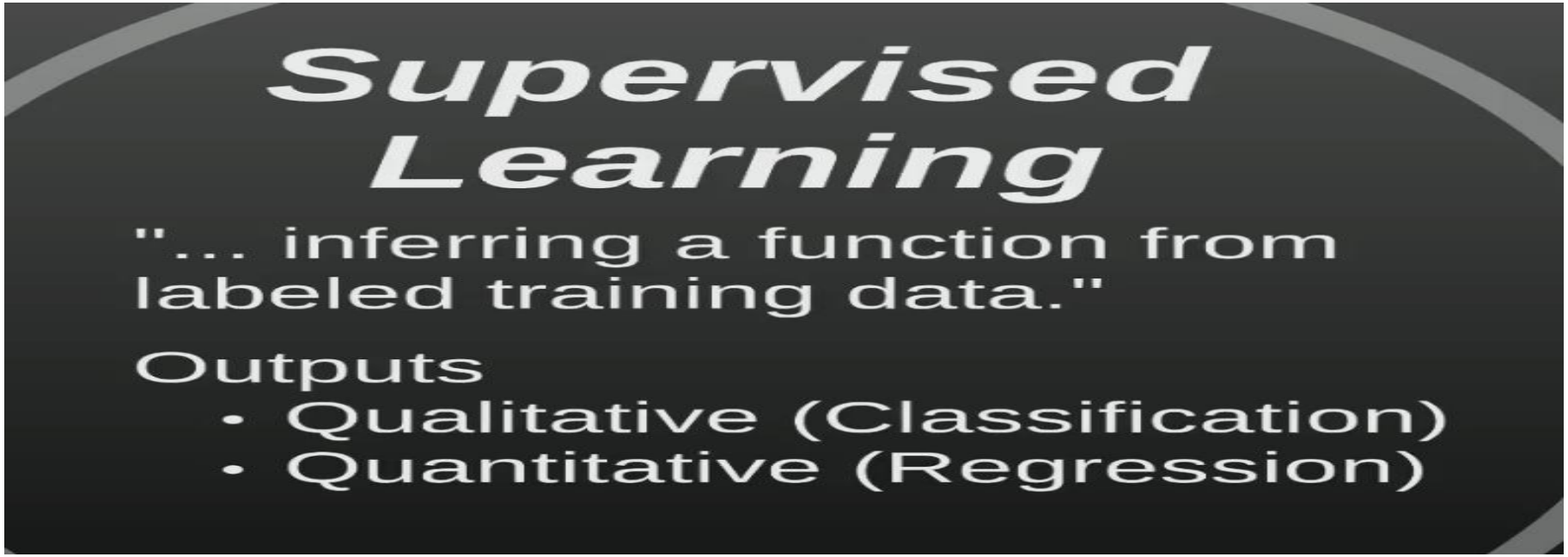
ALGORITMI DI MACHINE LEARNING

APPRENDIMENTO (TRAINING)

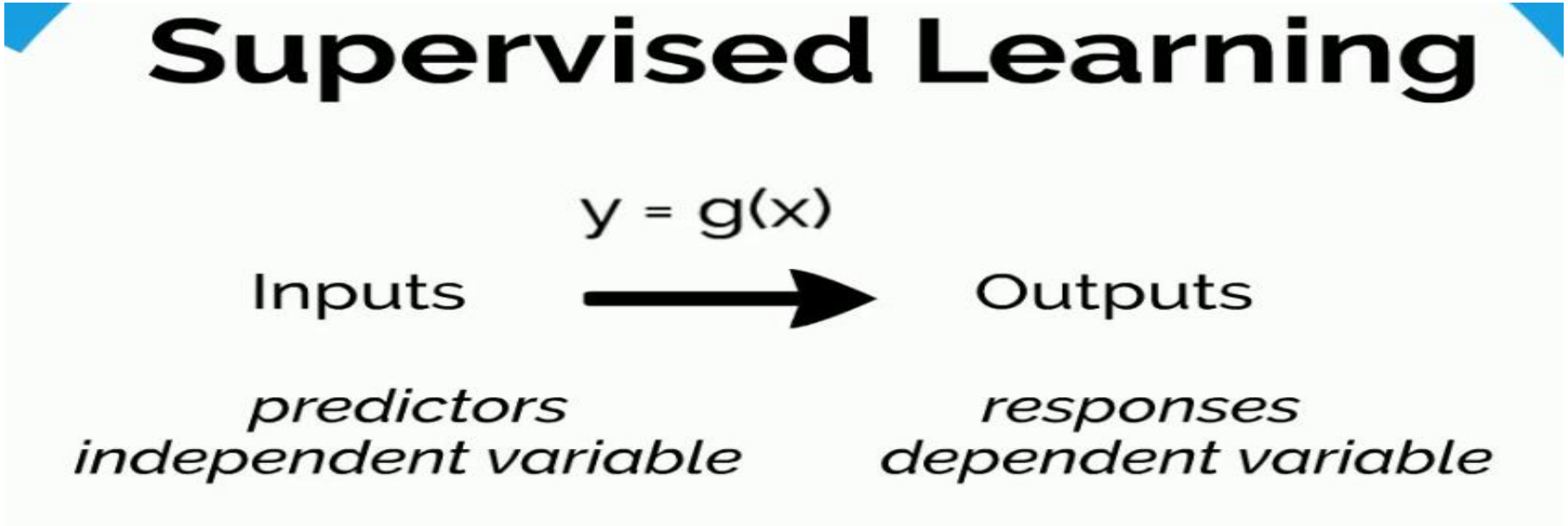
- Supervisionato (Supervised): sono note le classi dei patterr utilizzati per l'addestramento.
  - *il training set è etichettato*.



ALGORITMI DI MACHINE LEARNING

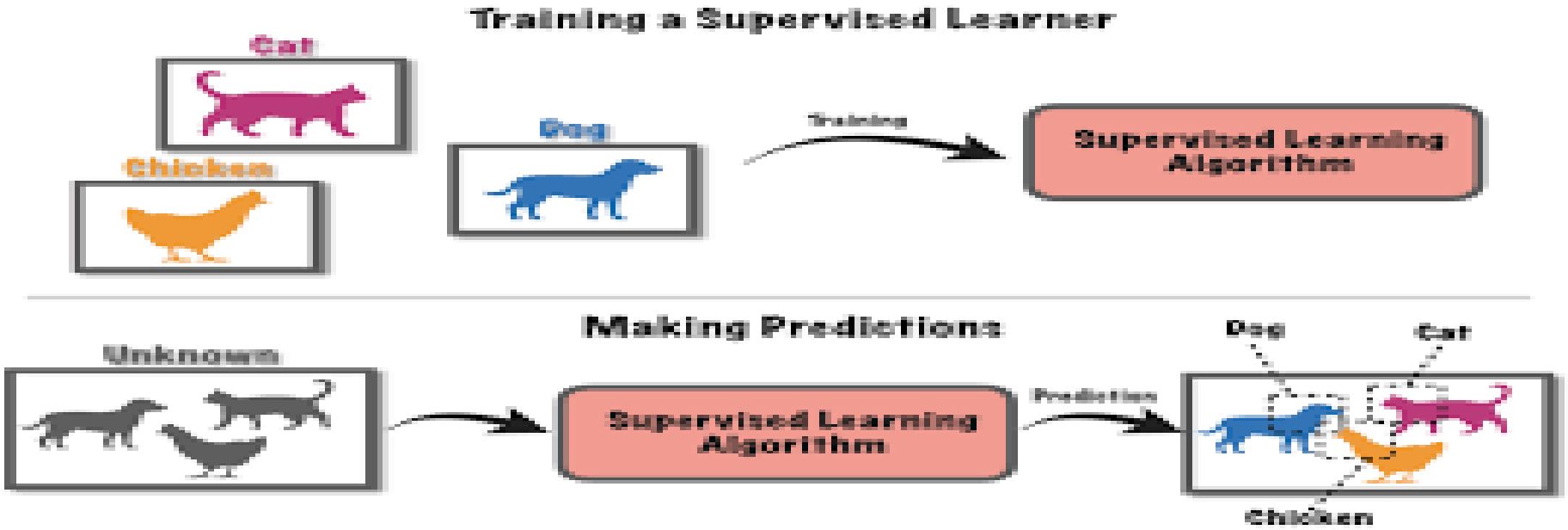
APPRENDIMENTO (TRAINING)

- Supervisionato (Supervised): sono note le classi dei patterr utilizzati per l'addestramento.
  - *il training set è etichettato*.



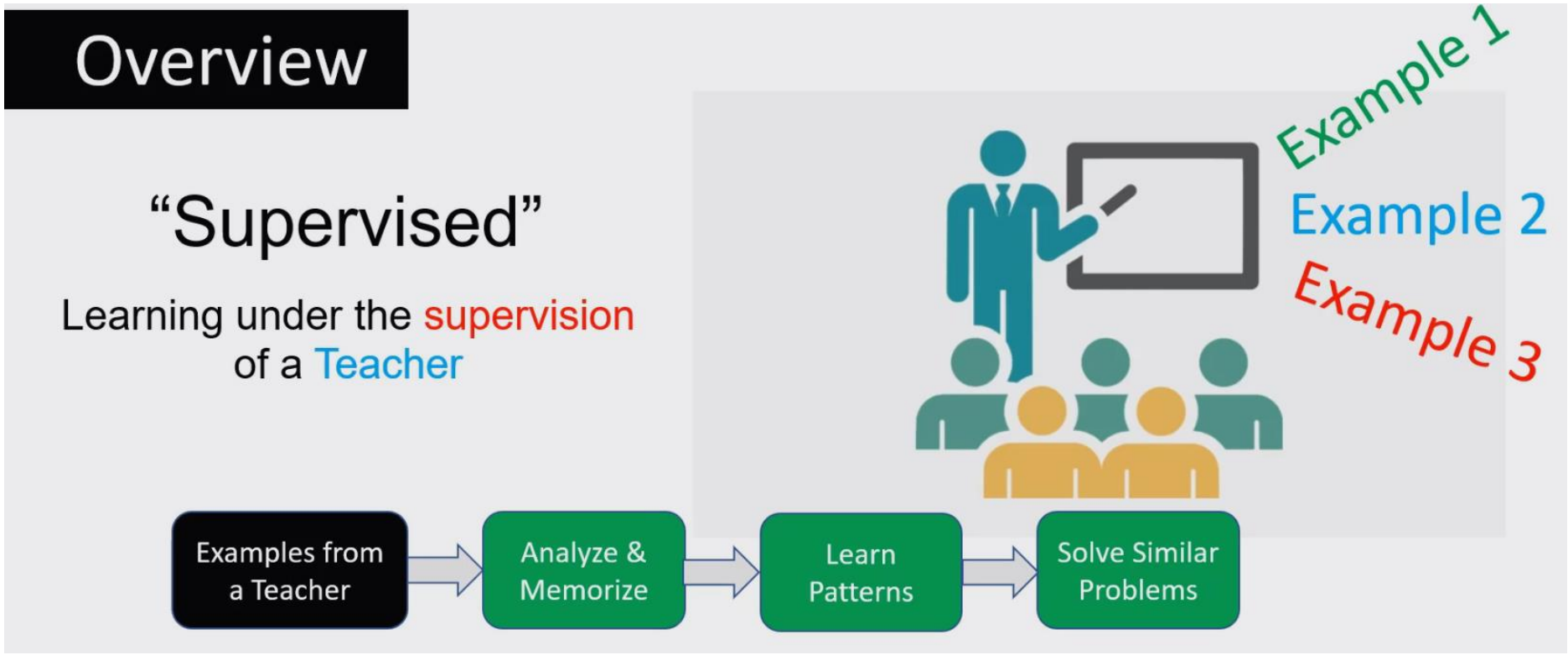
ALGORITMI DI MACHINE LEARNING

TEST SUPERVISIONATO

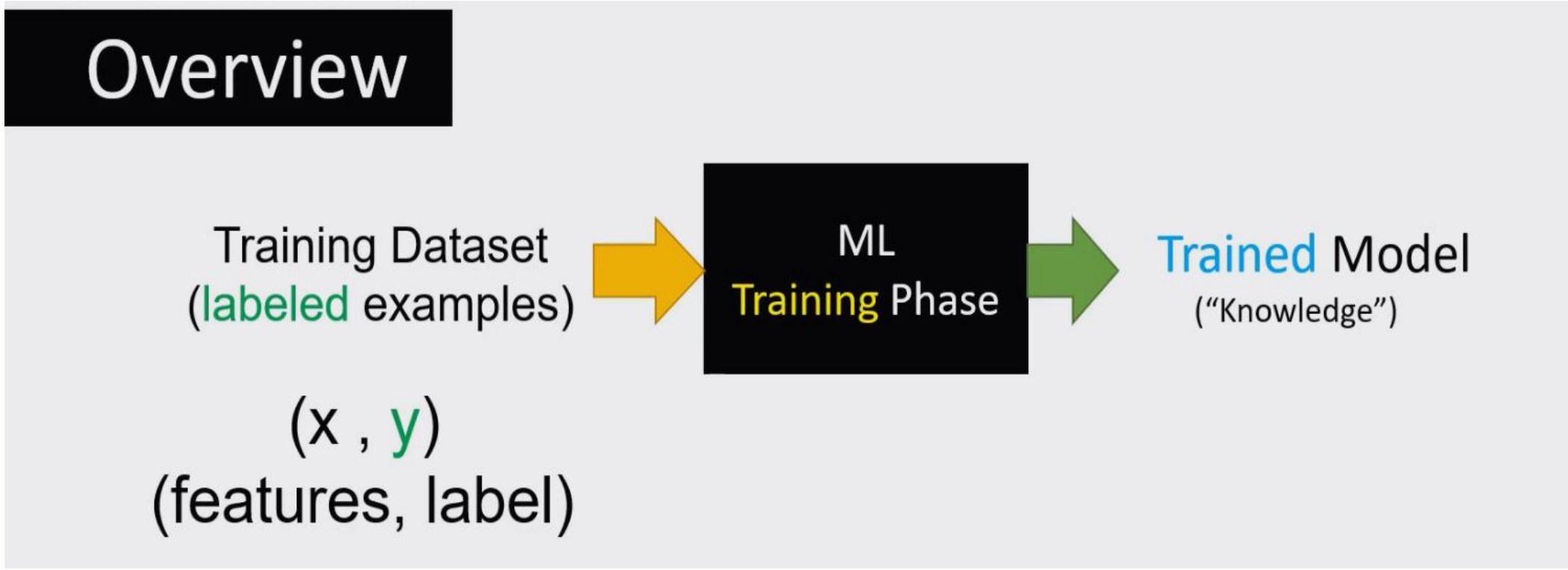


TEST SET: è l'insieme di pattern su cui valutare le prestazioni finali del sistema neurale.

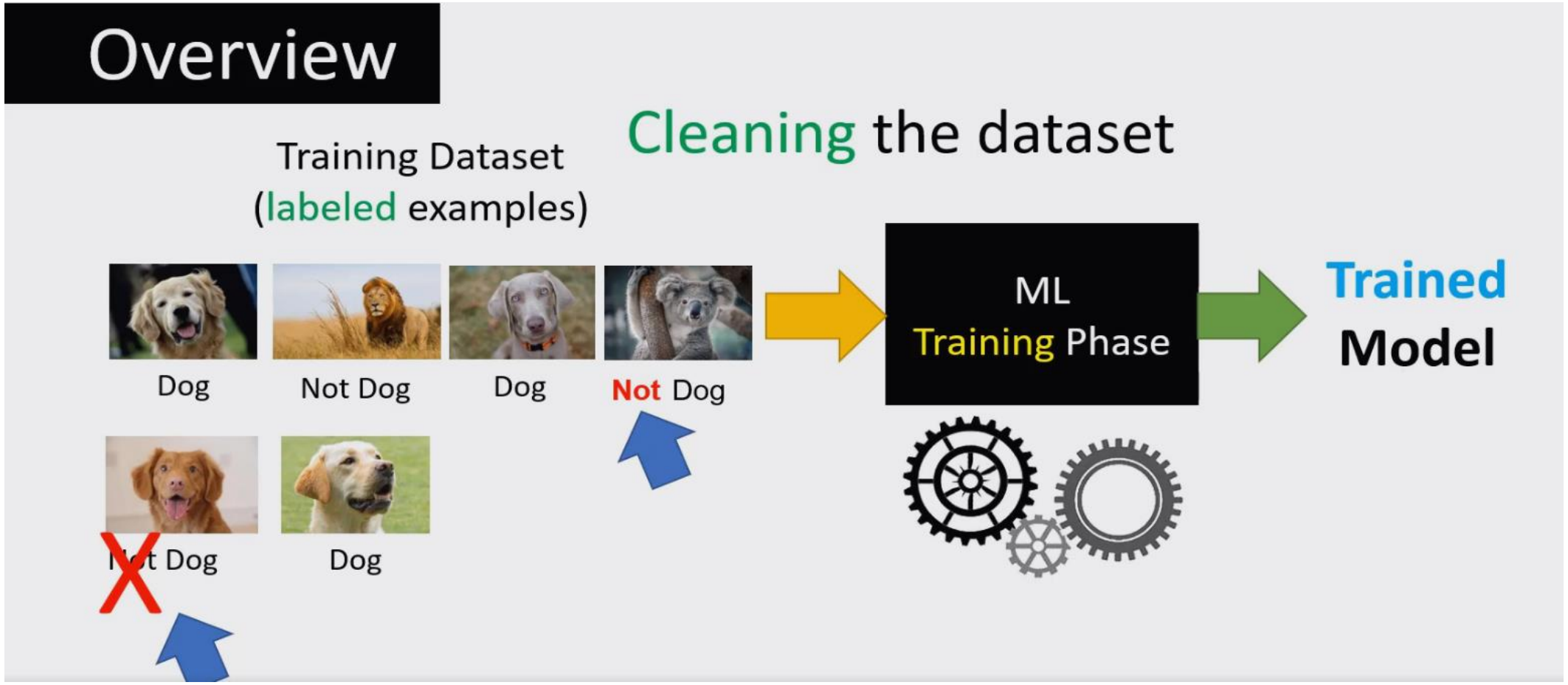
APPRENDIMENTO SUPERVISIONATO



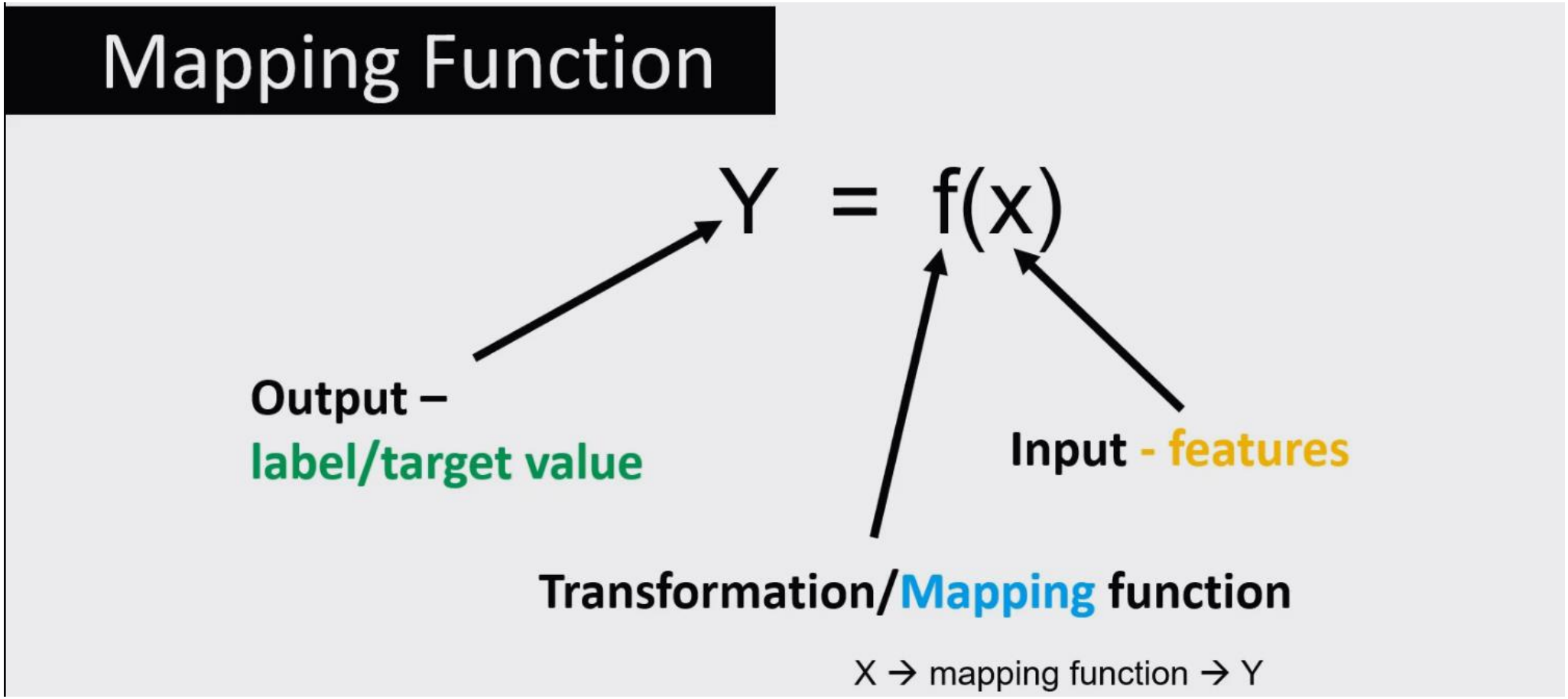
APPRENDIMENTO SUPERVISIONATO



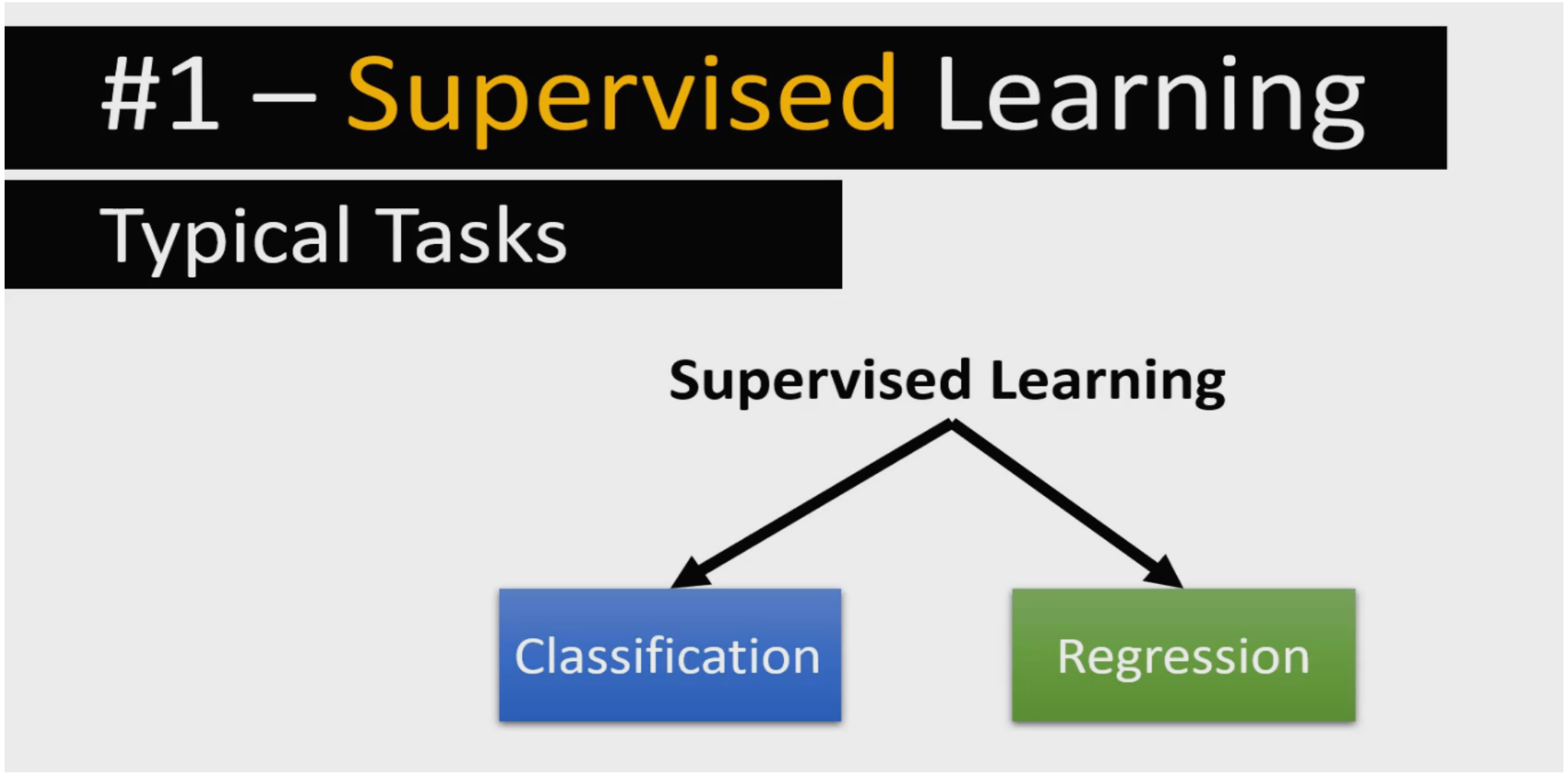
APPRENDIMENTO SUPERVISIONATO



APPRENDIMENTO SUPERVISIONATO



APPRENDIMENTO SUPERVISIONATO



COME FUNZIONA IL MACHINE LEARNING?

PROBLEMA DELLA CLASSIFICAZIONE

Classificazione

- Classificazione: assegna una classe a un pattern.
  - Necessario apprendere una funzione capace di eseguire il mapping dallo spazio dei pattern allo spazio delle classi
  - Si usa spesso anche il termine riconoscimento.
  - Nel caso di 2 sole classi si usa il termine binary classification, con più di due classi multi-class classification.

- Classe: insieme di pattern aventi proprietà comuni.
  - Es. i diversi modi in cui può essere scritto a mano libera il carattere A.
  - Il concetto di classe è semantico e dipende strettamente dall'applicazione:

    - 21 classi per il riconoscimento di lettere dell'alfabeto

    - 2 classi per distinguere le lettere dell'alfabeto italiano da quello cirillico

COME FUNZIONA IL MACHINE LEARNING?

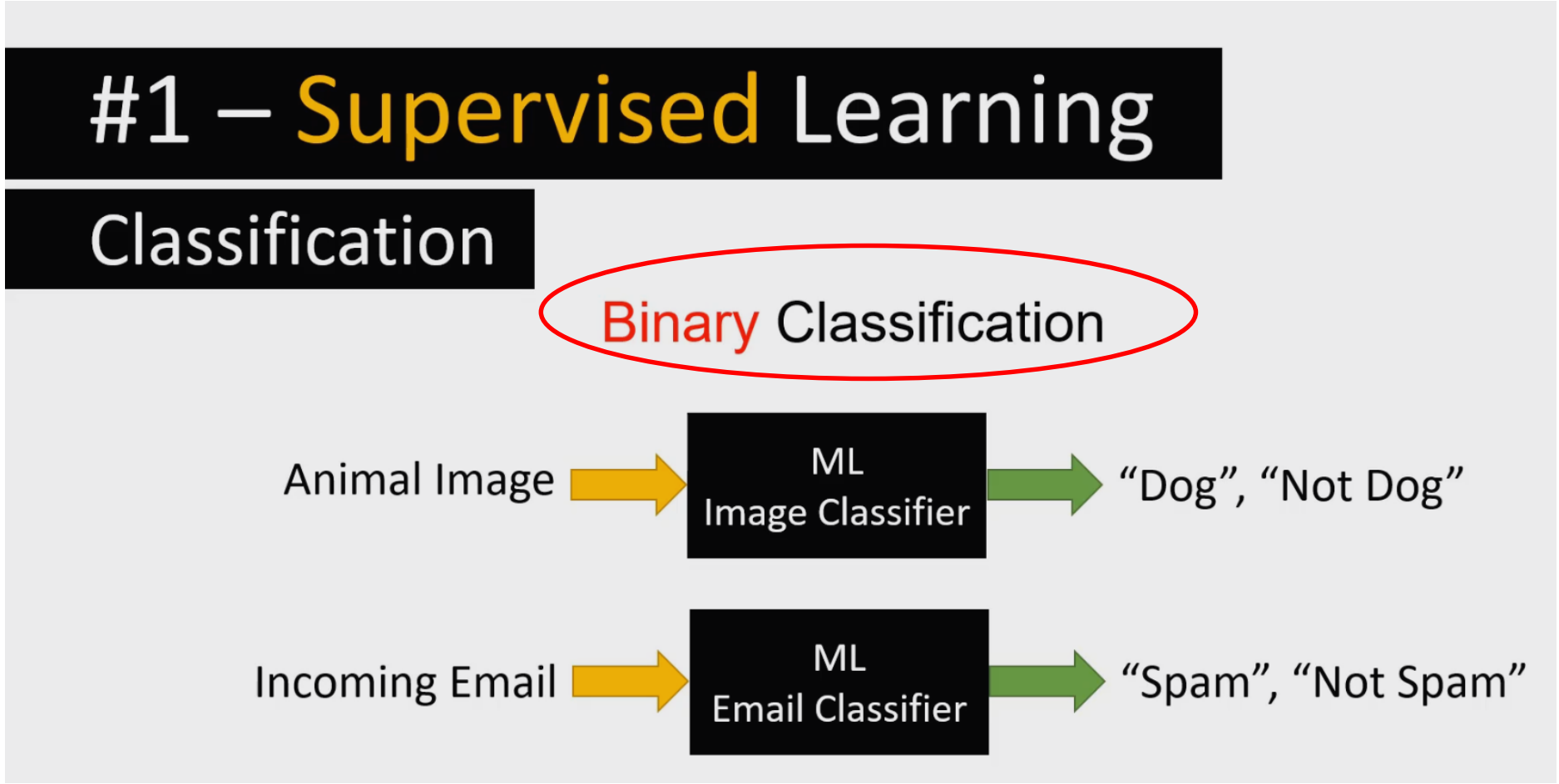
PROBLEMA DELLA CLASSIFICAZIONE

Handwritten ZIP codes on envelopes from US postal mail

- Output classes (0,1, ... , 9)
- Classification Problem

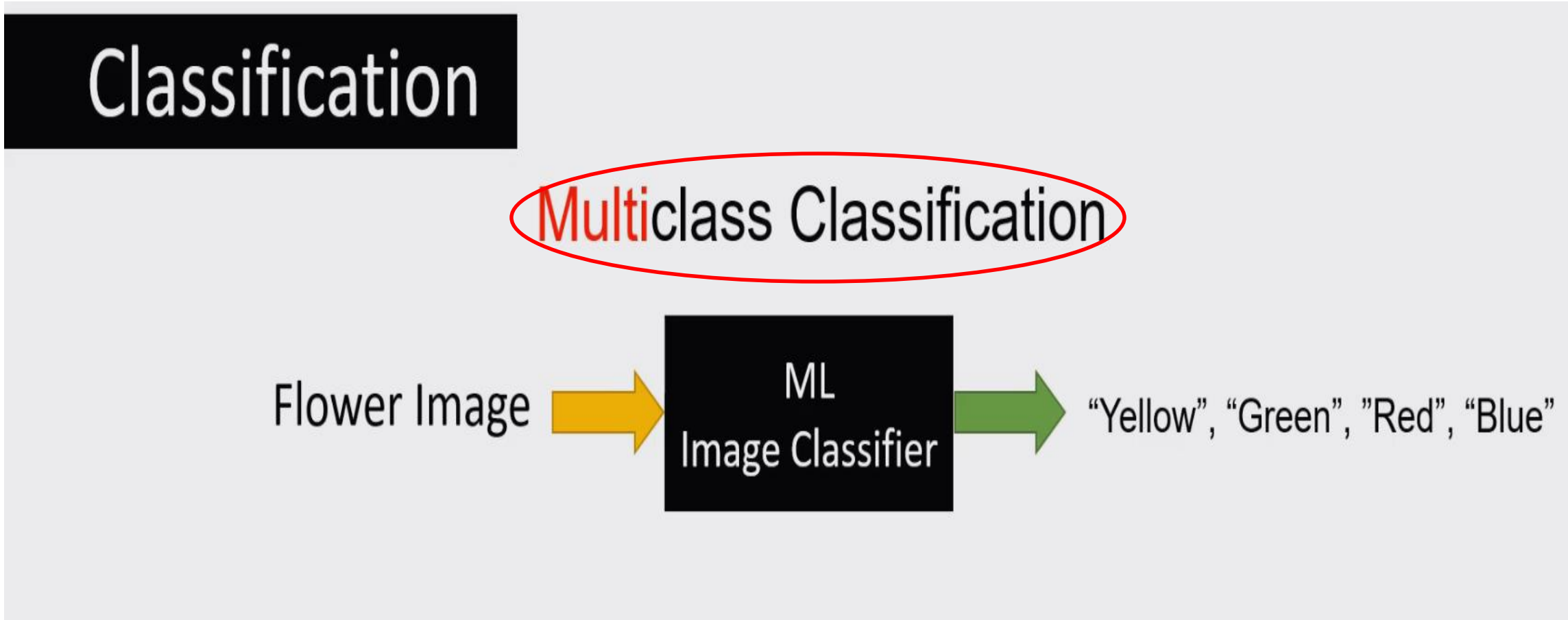
APPRENDIMENTO SUPERVISIONATO

CLASSIFICAZIONE



APPRENDIMENTO SUPERVISIONATO

CLASSIFICAZIONE



APPRENDIMENTO SUPERVISIONATO

CLASSIFICAZIONE

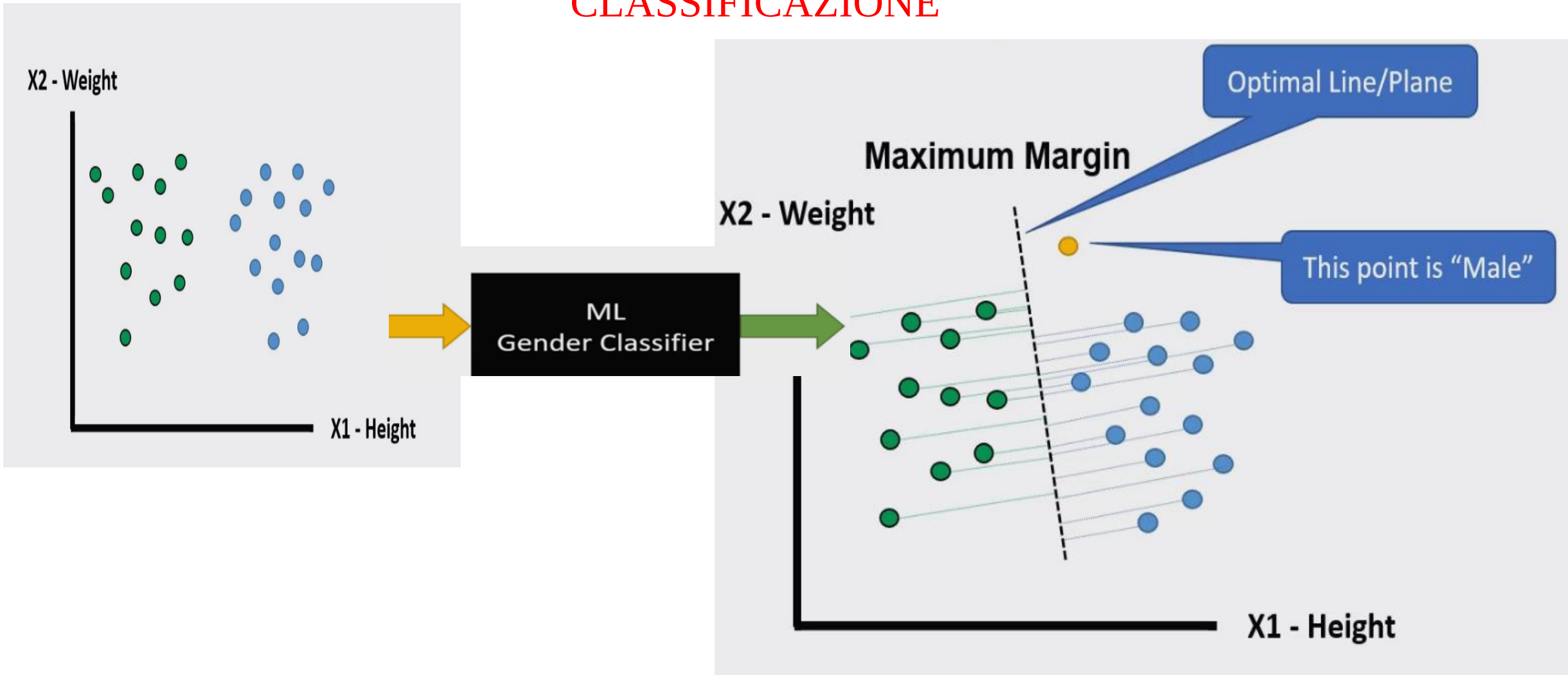
Classification

Support Vector Machines (SVMs)

X1 – H	X2 - W	Gender
160	56	F
167	74	M
183	85	M
.....		

APPRENDIMENTO SUPERVISIONATO

CLASSIFICAZIONE



APPRENDIMENTO SUPERVISIONATO

REGRESSIONE

Apprendimento per regressione

Nel machine learning la regressione lineare è una tecnica di classificazione degli esempi di un dataset (insieme di training) per consentire alla macchina di apprendere automaticamente un modello decisionale.

L'algoritmo assegna delle etichette (categorie) alle istanze utilizzando una funzione continua.

Ogni riga del dataset è un esempio composto da:

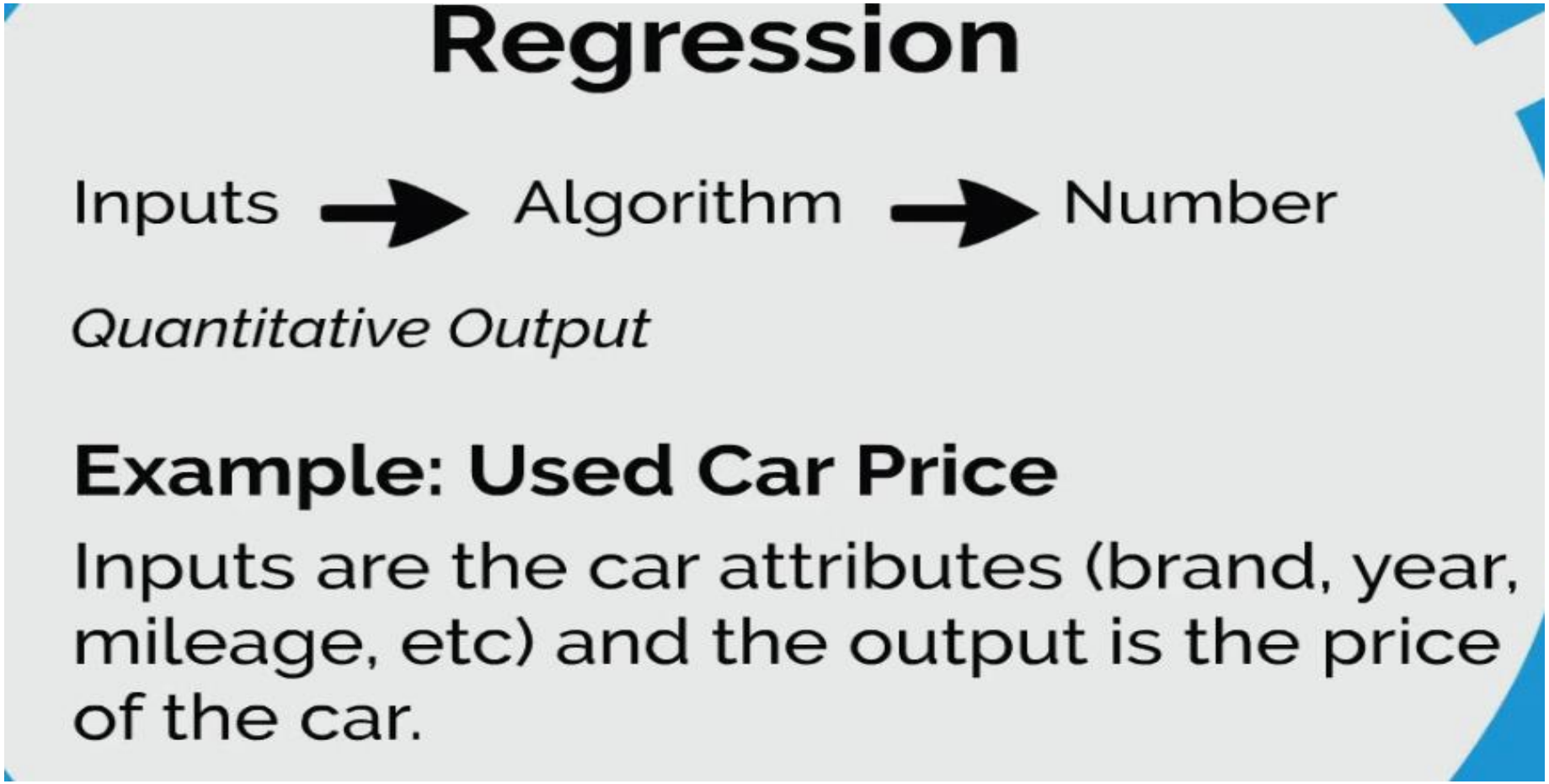
- **Gli attributi (X)**. Sono le variabili predittive che descrivono una categoria.
- **Il risultato (Y)**. E' la variabile target che indica alla macchina il risultato corretto se l'istanza appartenesse all'etichetta Z. E' il dato che istruisce la macchina a decidere in modo giusto.

L'algoritmo di machine learning deve trovare una relazione tra le variabili X e Y tramite la regressione.

$$Y=F(X)$$

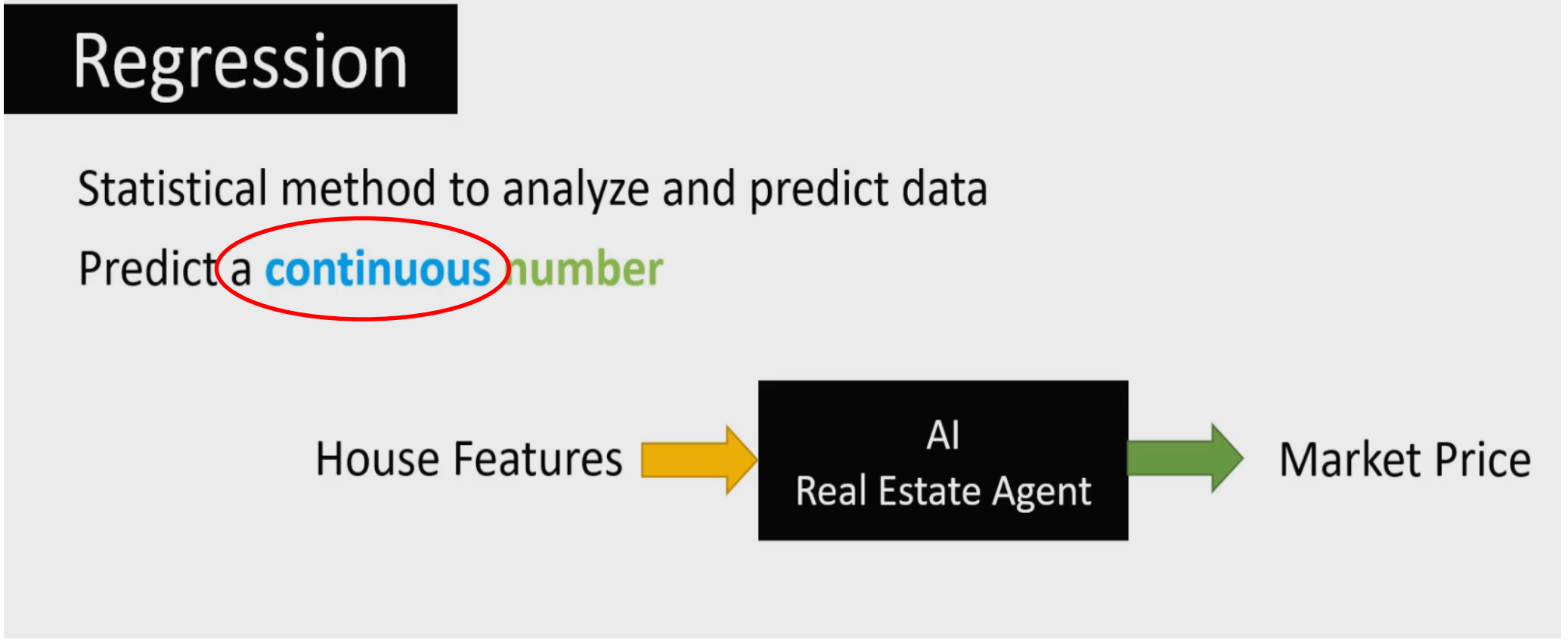
APPRENDIMENTO SUPERVISIONATO

REGRESSIONE

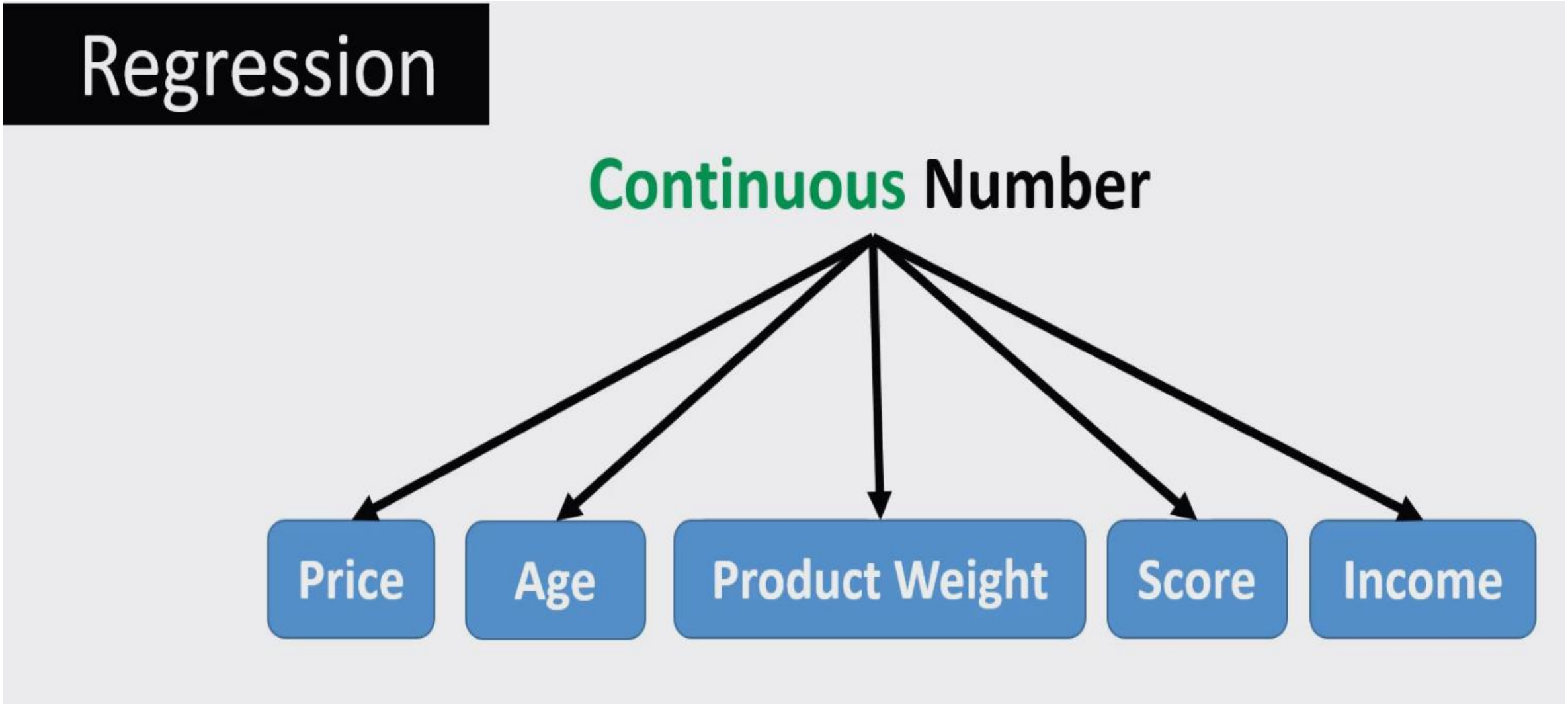


APPRENDIMENTO SUPERVISIONATO

REGRESSIONE

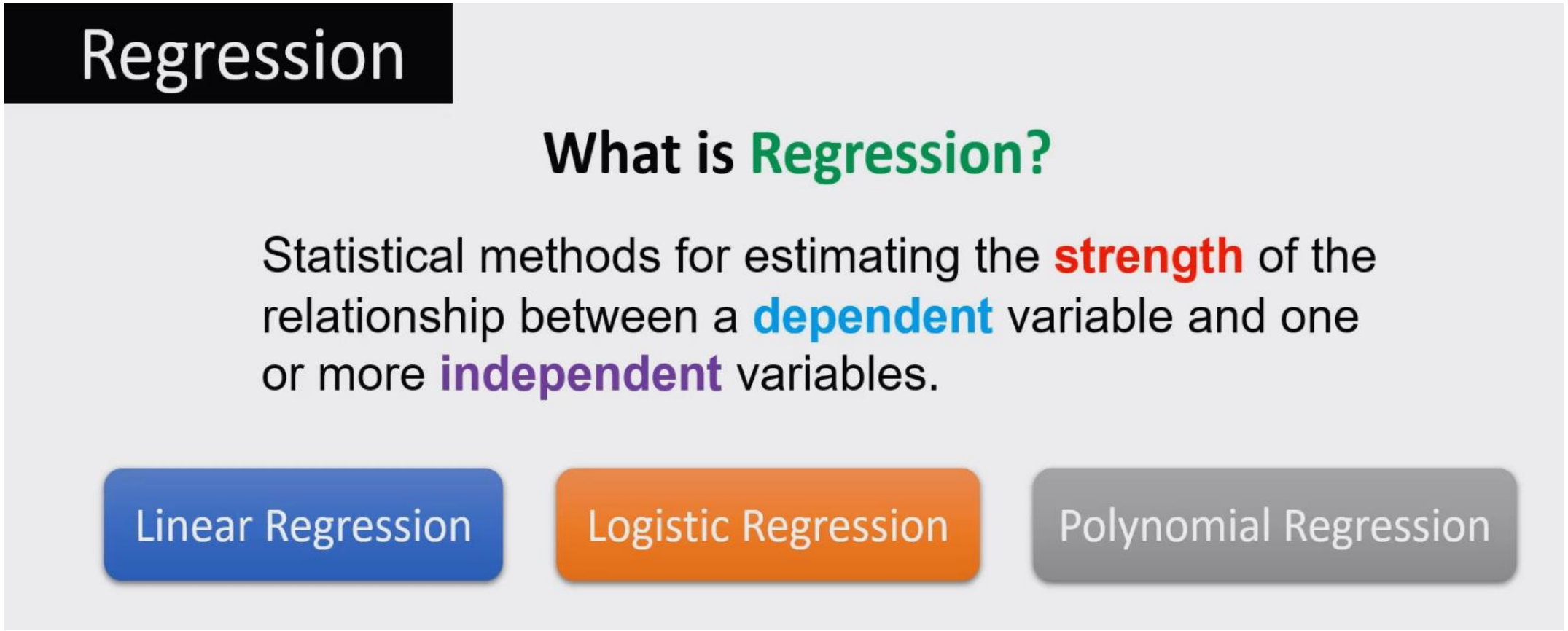


APPRENDIMENTO SUPERVISIONATO



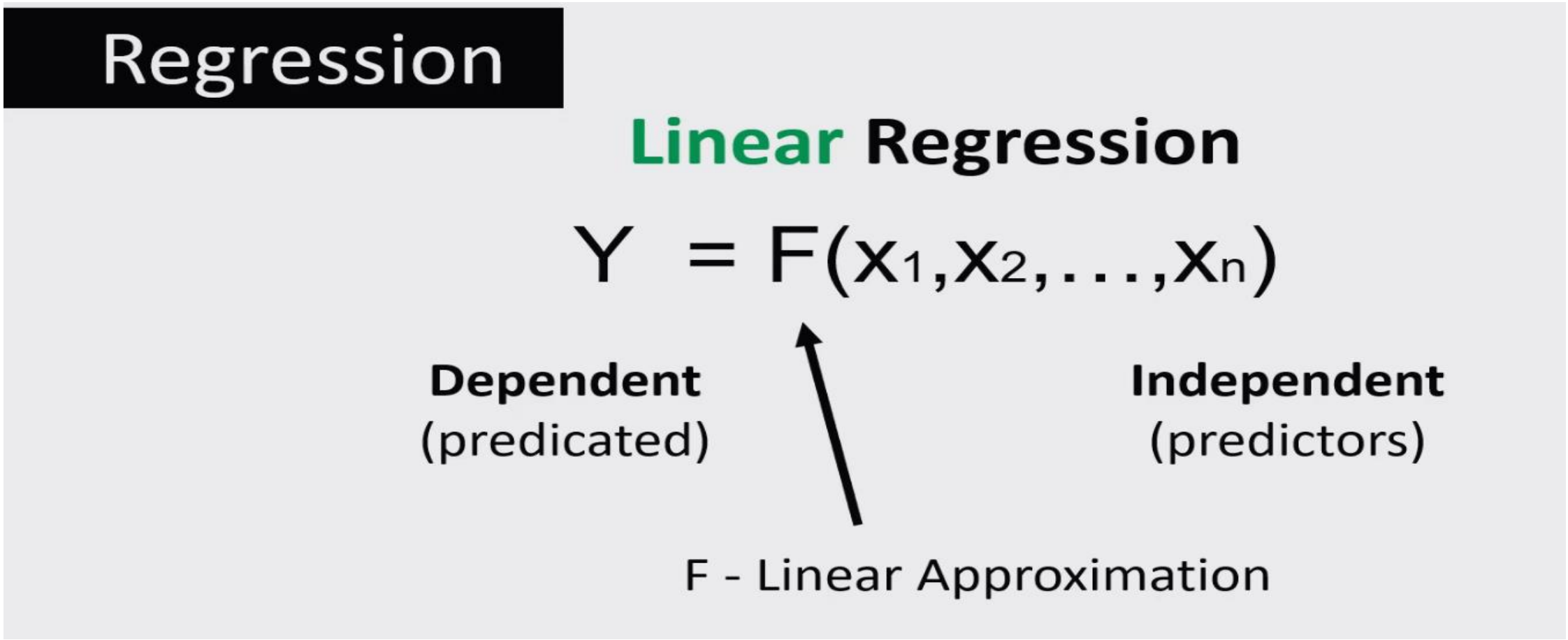
APPRENDIMENTO SUPERVISIONATO

REGRESSIONE



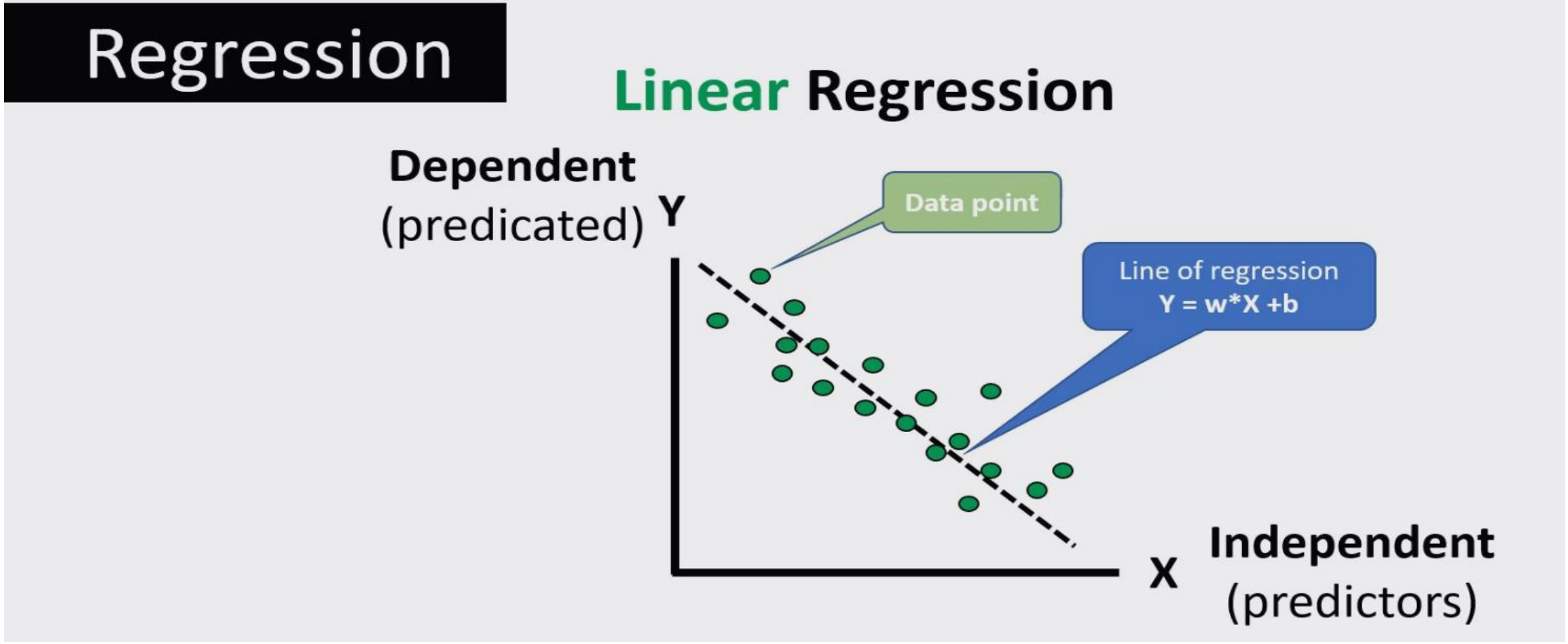
APPRENDIMENTO SUPERVISIONATO

REGRESSIONE



APPRENDIMENTO SUPERVISIONATO

REGRESSIONE



APPRENDIMENTO SUPERVISIONATO

REGRESSIONE

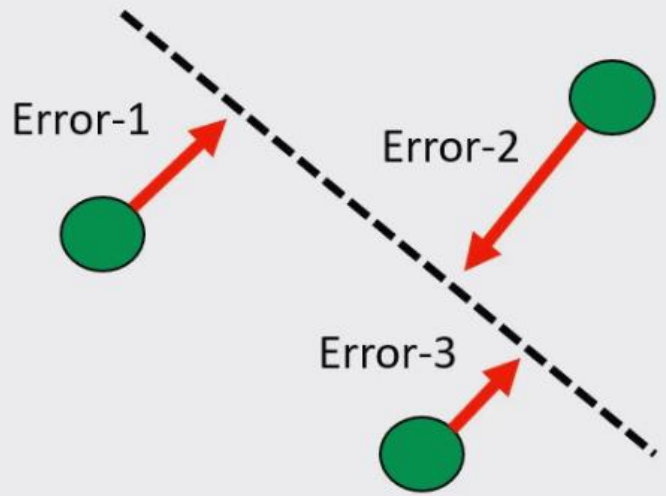
Regression

Linear Regression

Optimization using a Cost Function

Minimum(MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

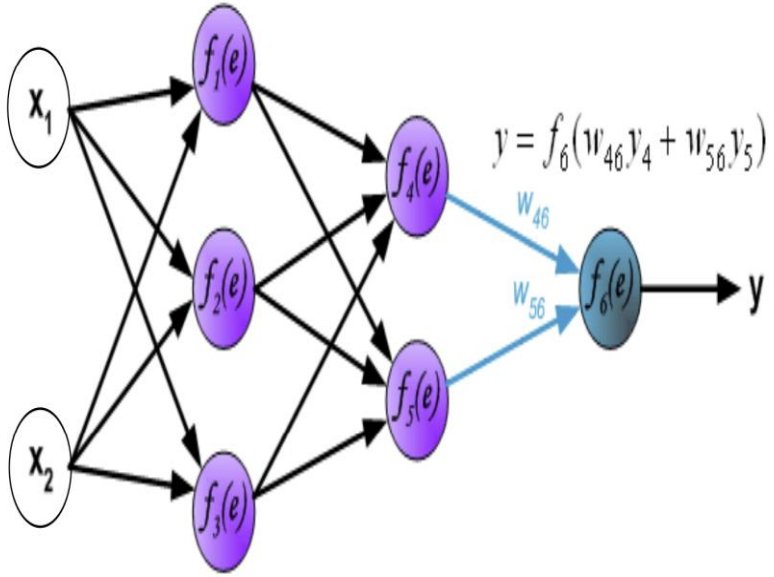


ALGORITMI DI MACHINE LEARNING

APPRENDIMENTO (TRAINING)

In genere, il comportamento di un algoritmo di Machine Learning è regolato da un set di parametri Θ (es. i pesi delle connessioni in una rete neurale). L'apprendimento consiste nel determinare il valore ottimo Θ^* di questi parametri.

Dato un training set $Train$ e un insieme di parametri, la funzione obiettivo $f(Train, \Theta)$ può indicare:



- l'ottimalità della soluzione (da massimizzare).

$$\Theta^* = argmax_\Theta\, f(Train, \Theta)$$

- oppure l'errore o perdita (loss-function) da minimizzare.

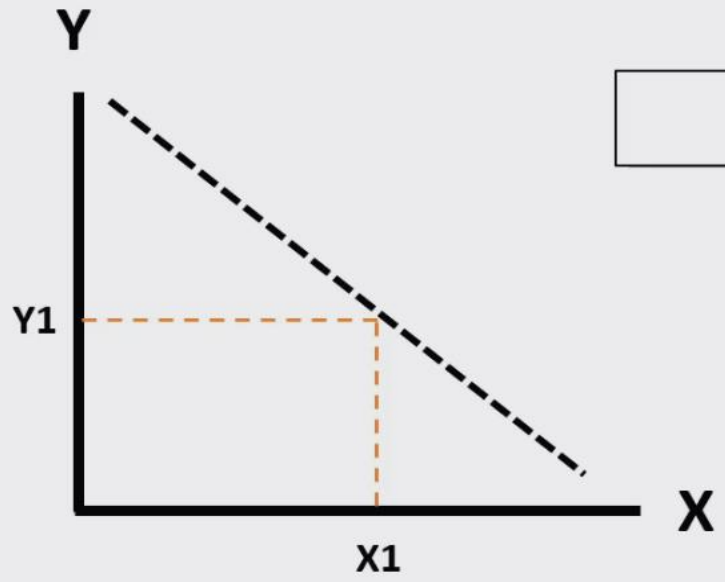
$$\Theta^* = argmin_\Theta\, f(Train, \Theta)$$

APPRENDIMENTO SUPERVISIONATO

REGRESSIONE

Regression

Linear Regression



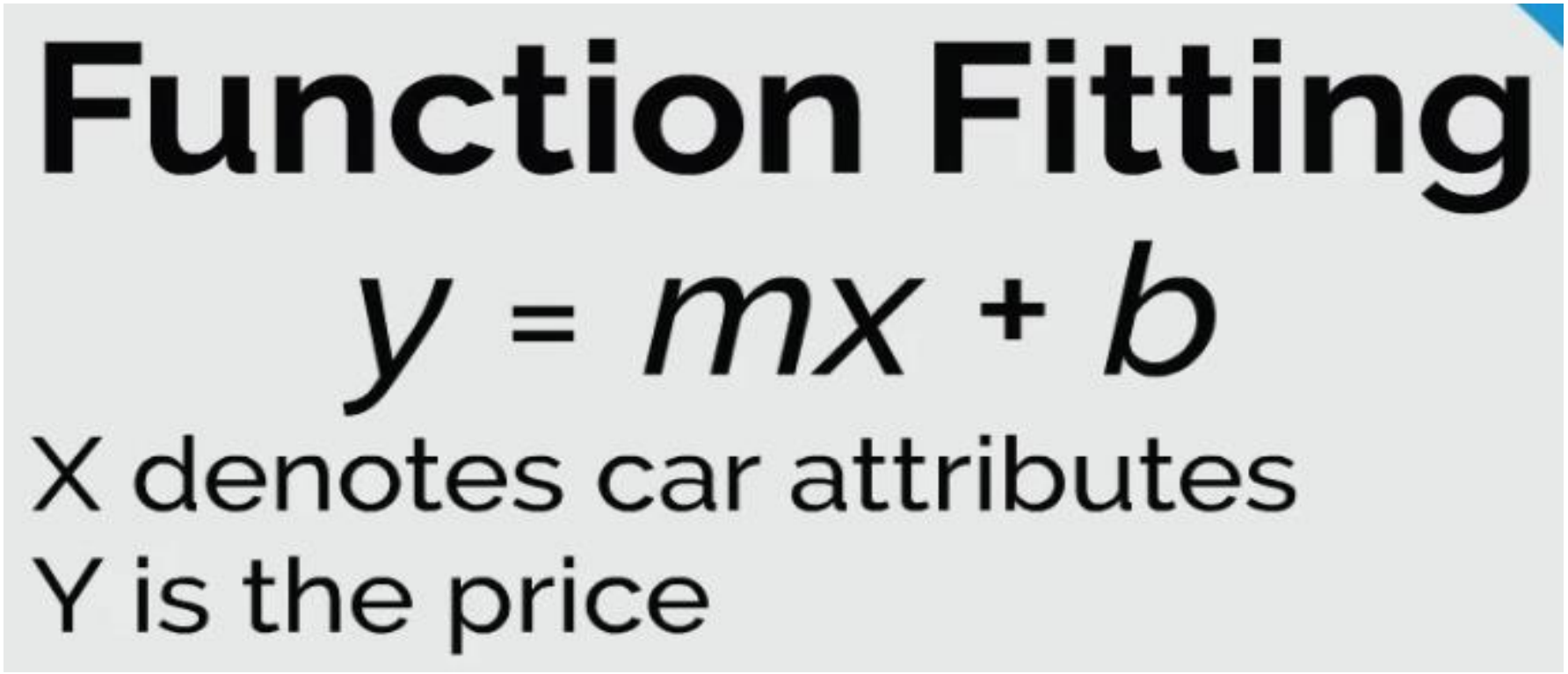
$$y = w[0]*x[0]+w[1]*x[1] +\ldots+ w[i]*x[i]+b$$

x[i] – input features

w[i], b – model parameters

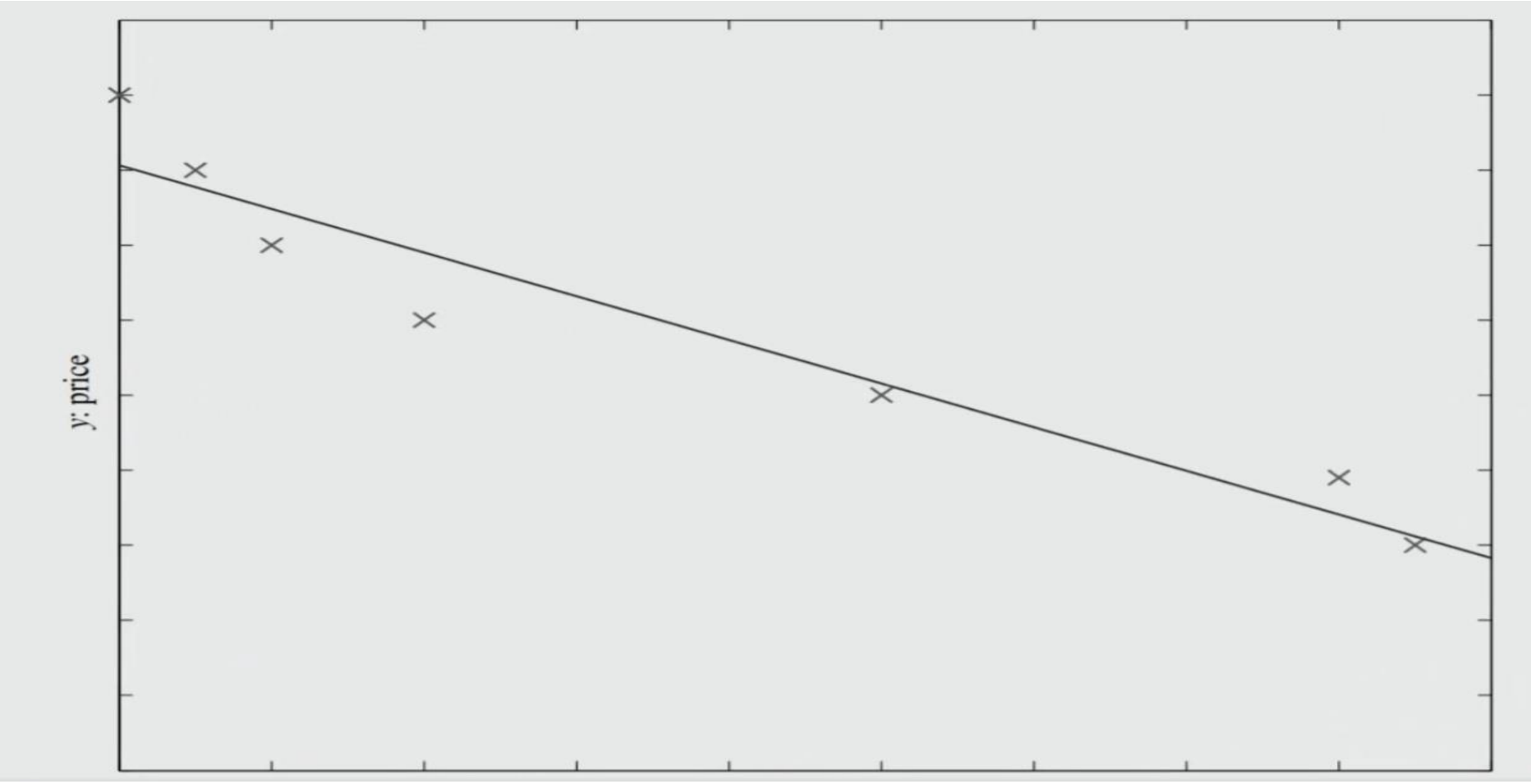
APPRENDIMENTO SUPERVISIONATO

REGRESSIONE



APPRENDIMENTO SUPERVISIONATO

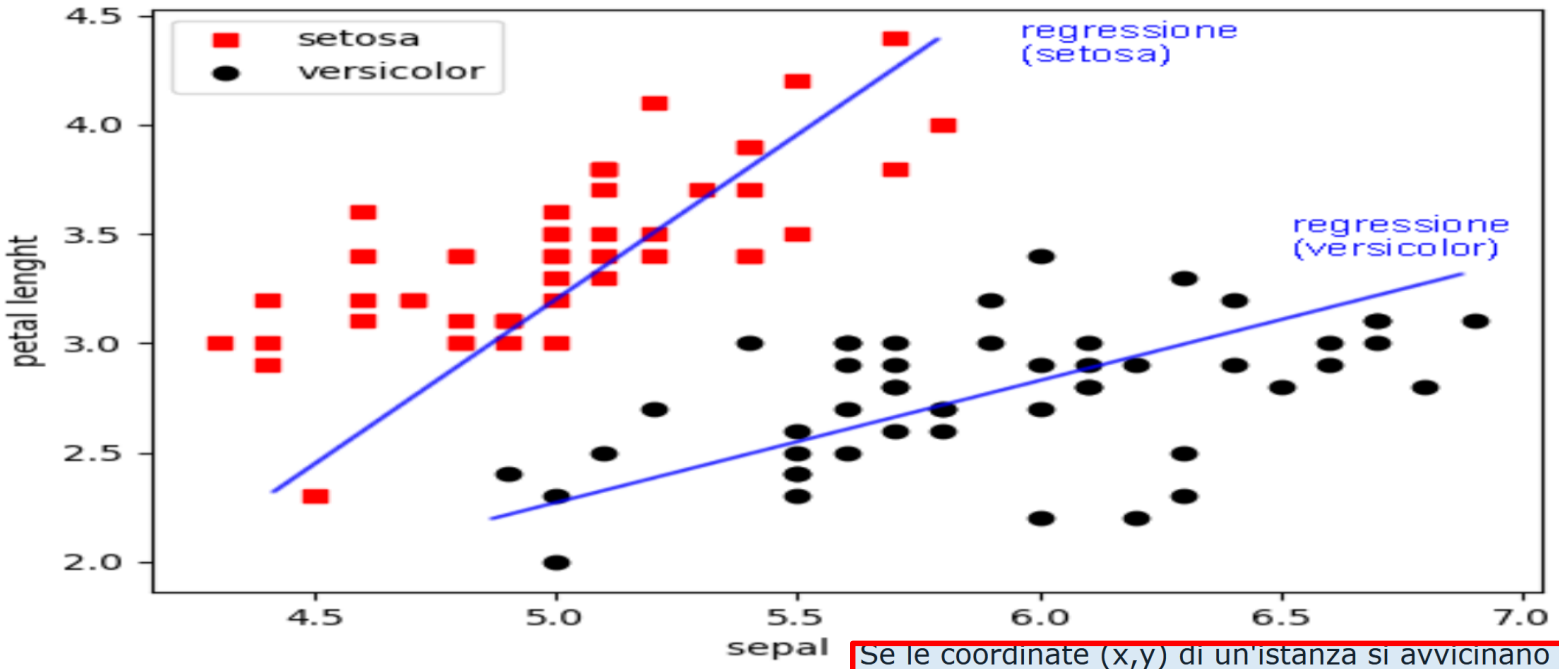
REGRESSIONE



APPRENDIMENTO SUPERVISIONATO

REGRESSIONE

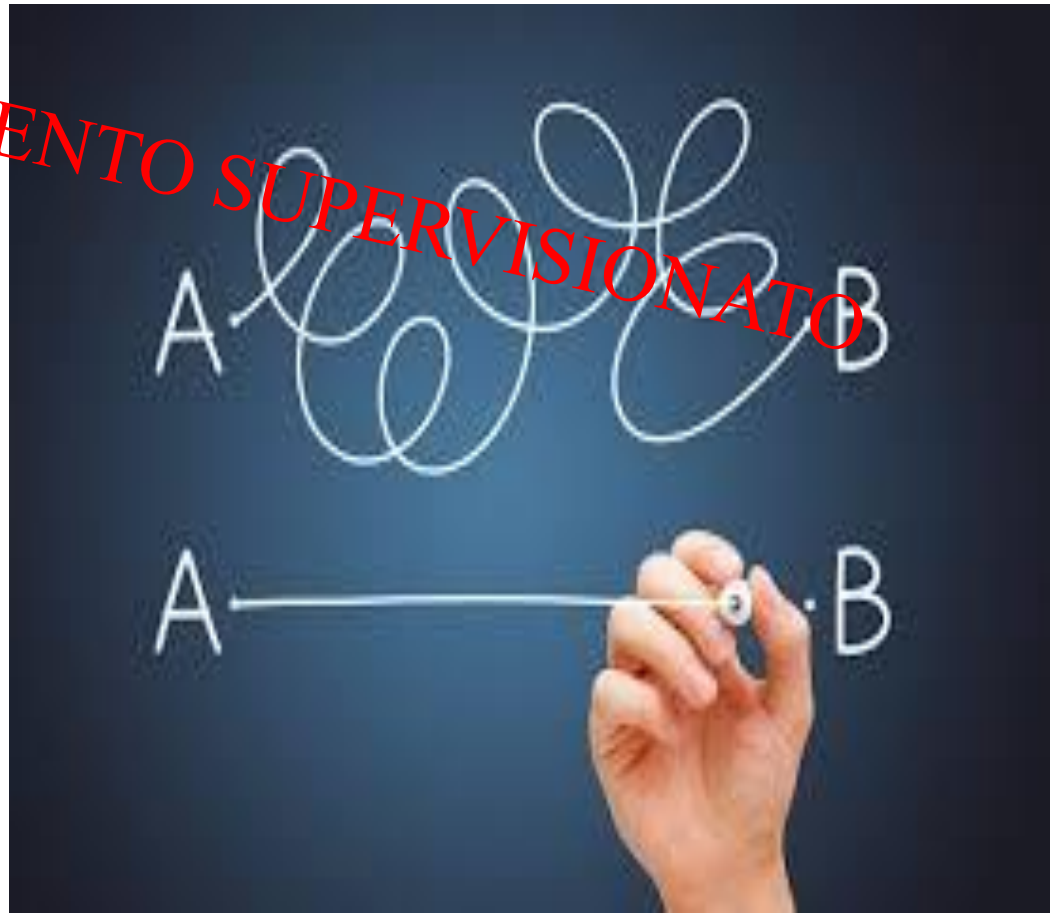
Il risultato finale è una linea retta che minimizza la distanza tra gli N esempi dell'insieme di training che appartengono alla stessa categoria.



Se le coordinate (x,y) di un'istanza si avvicinano alla funzione di regressione f(x), l'istanza viene classificata con l'etichetta Z.

Una volta trovata, la **funzione di classificazione** può essere utilizzata per valutare istanze diverse dall'insieme di training.

FINE APPRENDIMENTO SUPERVISIONATO



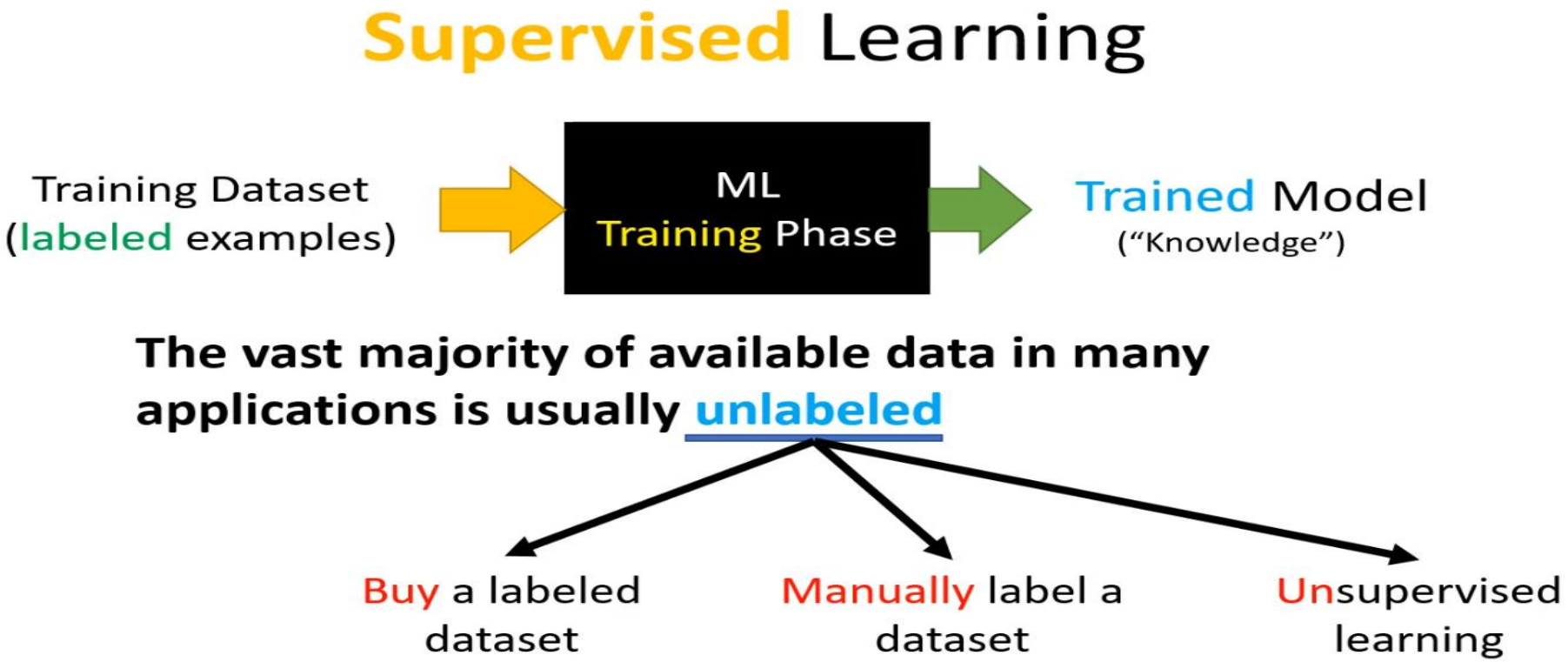
ALGORITMI DI MACHINE LEARNING

APPRENDIMENTO (TRAINING)

- Supervisionato (Supervised): sono note le classi dei patterr utilizzati per l'addestramento.
  - *il training set è etichettato*.
    - Non Supervisionato (Unsupervised): non sono note le classi dei pattern utilizzati per l'addestramento.
      - *il training set non è etichettato*.

- It is often easier to obtain unlabeled data – from a lab instrument or a computer – than labeled data, which can require human intervention

APPRENDIMENTO NON SUPERVISIONATO

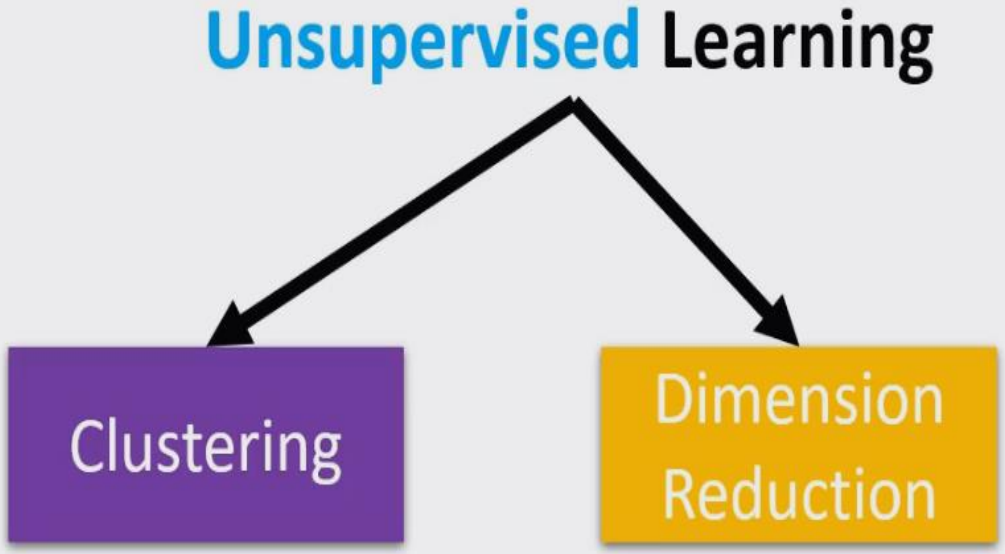


APPRENDIMENTO NON SUPERVISIONATO

Overview

Learning without a “teacher” supervising the learning process

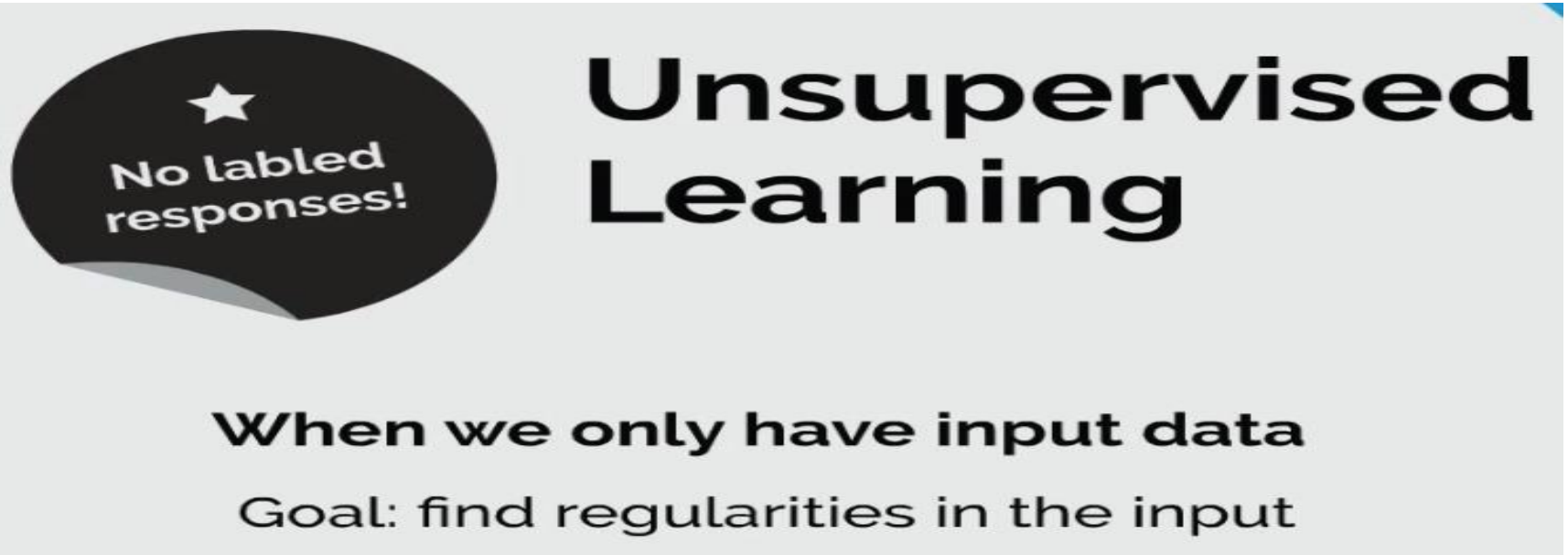
Identify automatically meaningful patterns in unlabeled data



ALGORITMI DI MACHINE LEARNING

APPRENDIMENTO (TRAINING)

- Non Supervisionato (Unsupervised): non sono note le classi dei pattern utilizzati per l'addestramento.
  - *il training set non è etichettato*.



APPRENDIMENTO NON SUPERVISIONATO

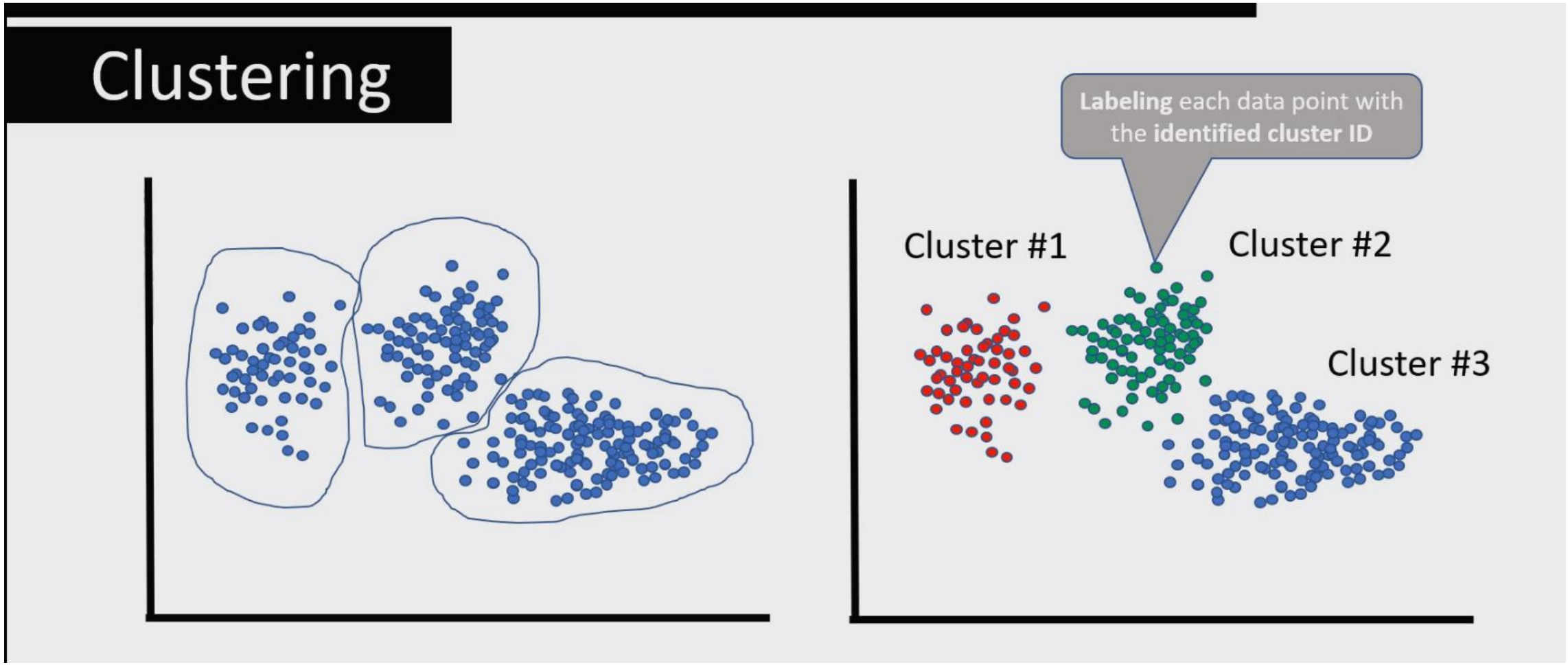
Clustering

What is Clustering?

Clustering is the task of identifying similar instances with **shared attributes** in a dataset and **group them together into clusters**

The **output** of the algorithm would be a set of **"labels"** assigning each data point to one of the identified clusters

APPRENDIMENTO NON SUPERVISIONATO



APPRENDIMENTO NON SUPERVISIONATO

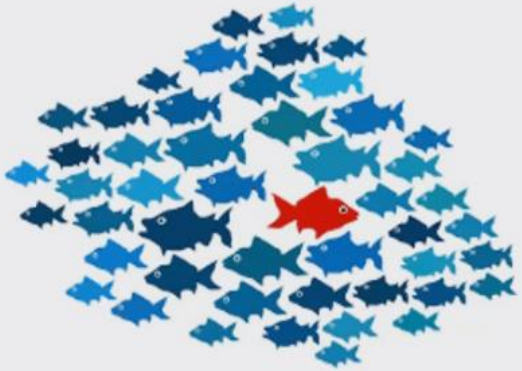
Clustering

Use Cases for Clustering

Customer Segmentation

Anomaly/Outlier Detection

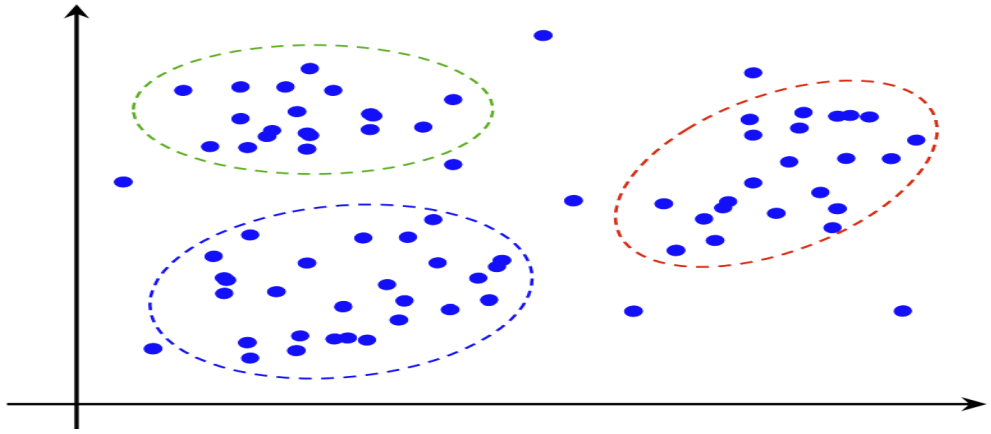
Semi-supervised Learning



ALGORITMI DI MACHINE LEARNING

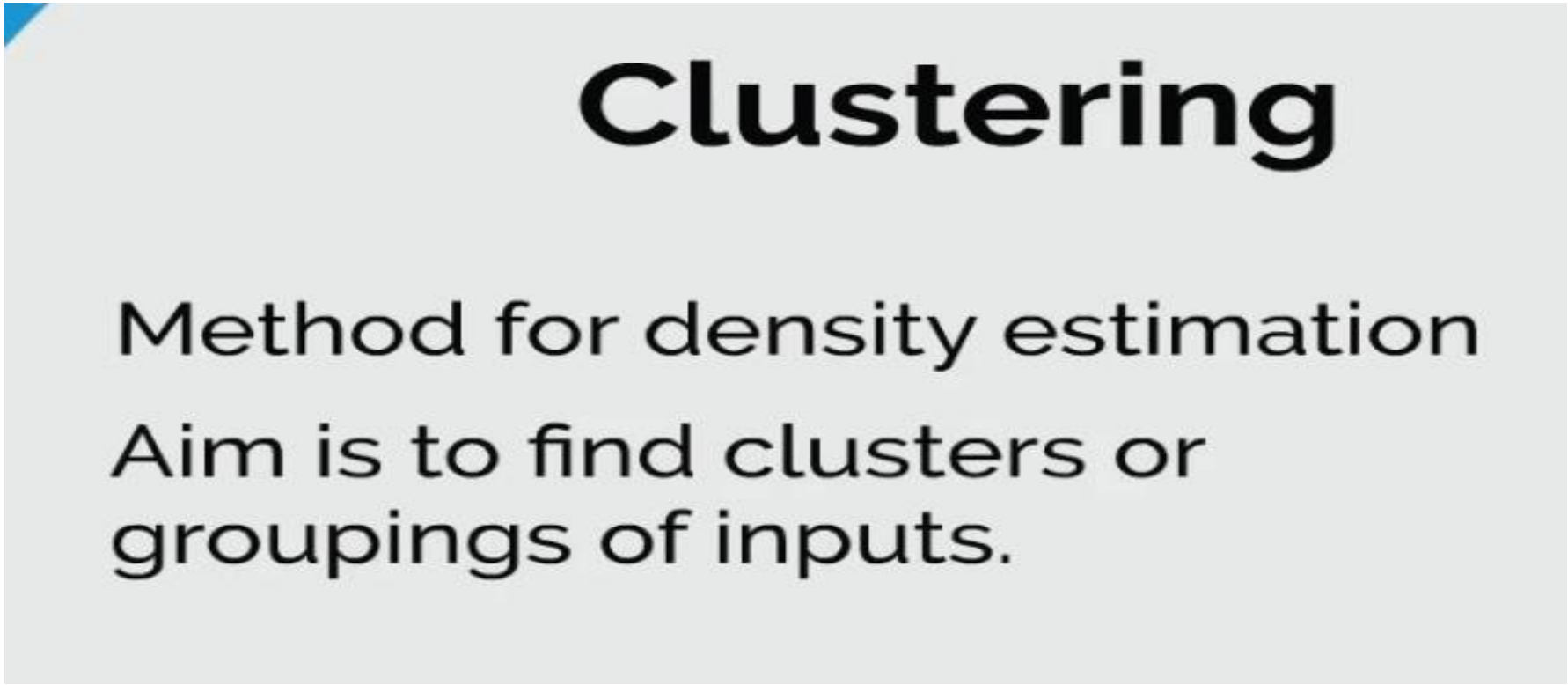
TRAINING NON SUPERVISIONATO - CLUSTERING

- Clustering: individua gruppi (cluster) di pattern con caratteristiche simili.
  - Le classi del problema non sono note e i pattern non etichettati → la natura non supervisionata del problema lo rende più complesso della classificazione.
  - Spesso nemmeno il numero di cluster è noto a priori
  - I cluster individuati nell'apprendimento possono essere poi utilizzati come classi.



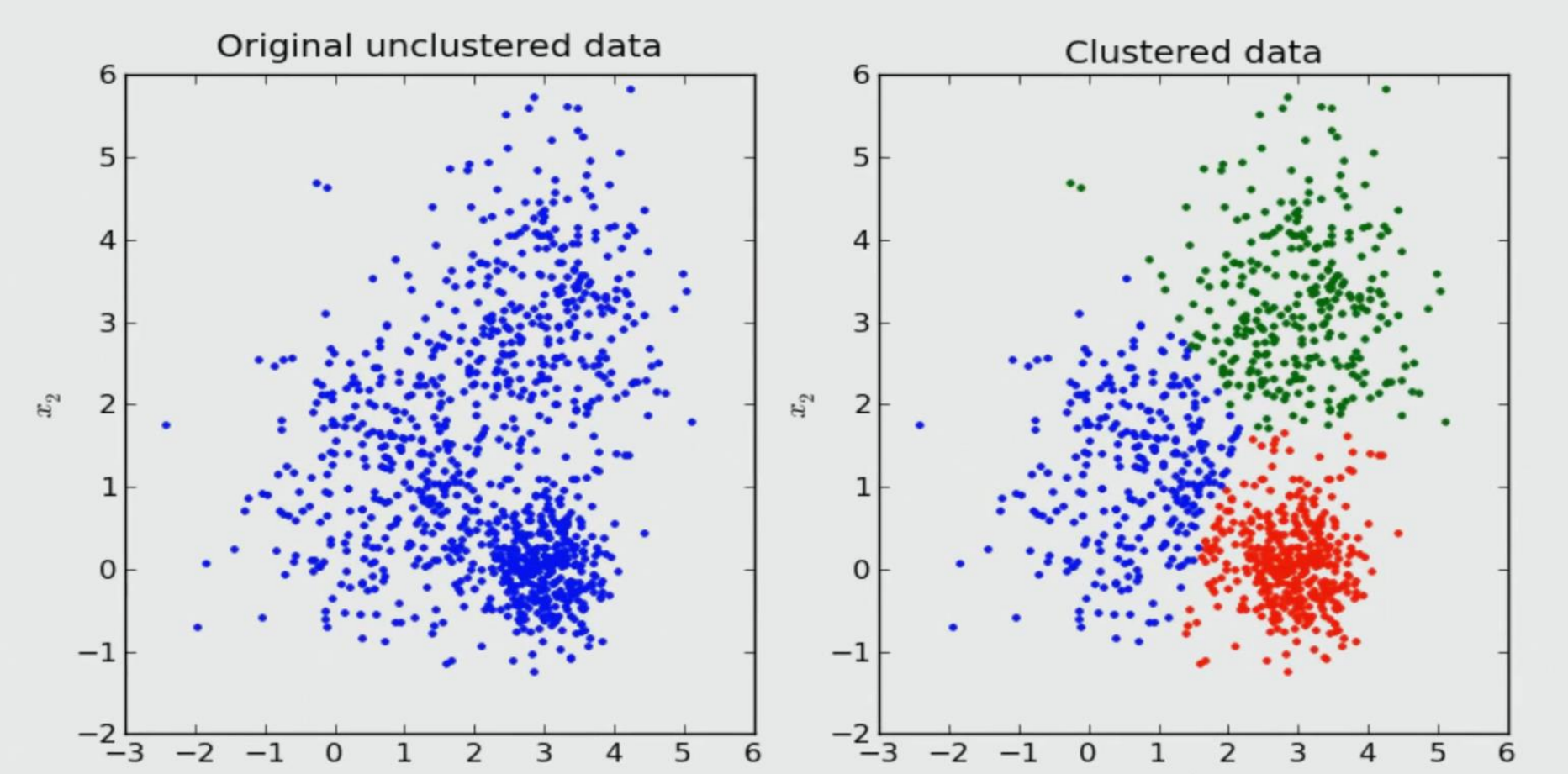
ALGORITMI DI MACHINE LEARNING

TRAINING NON SUPERVISIONATO - CLUSTERING



ALGORITMI DI MACHINE LEARNING

TRAINING NON SUPERVISIONATO - CLUSTERING



ABBIAMO SOLO DUE VARIABILI

ALGORITMI DI MACHINE LEARNING

TIPOLOGIE DI TRAINING

- Batch: l'addestramento è effettuato una sola volta su un training set dato.
  - una volta terminato il training, il sistema passa in «working mode» e non è in grado di apprendere ulteriormente.
  - Attualmente, la maggior parte dei sistemi di machine learning opera in questo modo.

- Incrementale: a seguito dell'addestramento iniziale, sono possibili ulteriori sessioni di addestramento.
  - Scenari: Sequenze di Batch, Unsupervised Tuning.
  - Rischio: Catastrofic Forgetting (il sistema dimentica quello che ha appreso in precedenza).

- Naturale: addestramento continuo (per tutta la vita)
  - Addestramento attivo in working mode.

ALGORITMI DI MACHINE LEARNING

ACCURATEZZA RISULTATI

In un problema di Classificazione, l'accuratezza di classificazione [0…100%] è la percentuale di pattern correttamente classificati. L'errore di classificazione è il complemento.

$$Accuratezza = \frac{pattern\ correttamente\ classificati}{pattern\ classificati}$$

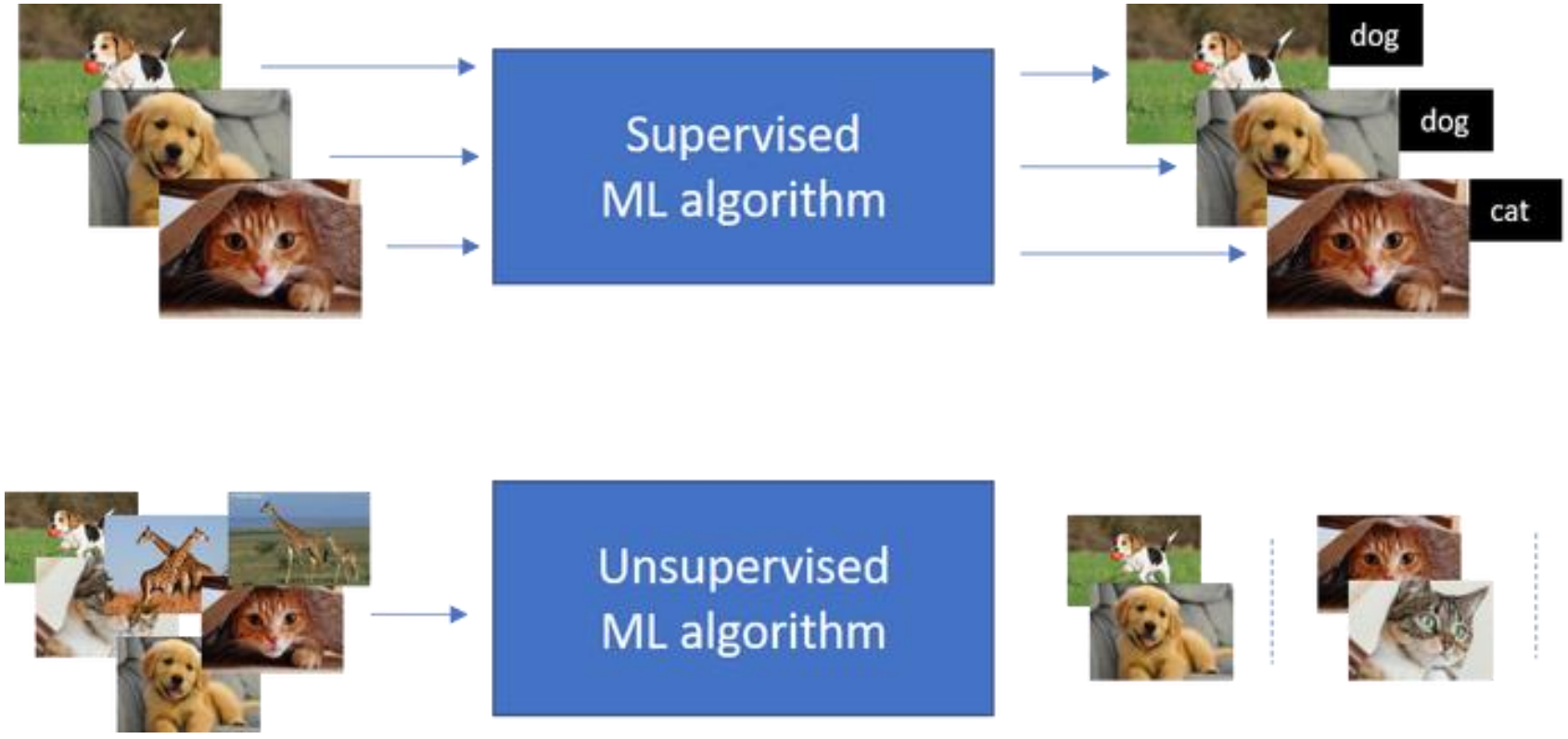
$$Errore = 100\% - Accuratezza$$

Nei problemi di Regressione, si valuta in genere l'RMSE (Root Mean Squared Error) ovvero la radice della media dei quadrati degli scostamenti tra valore vero e valore predetto.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1..N}(pred_i - true_i)^2}$$

ALGORITMI DI MACHINE LEARNING

ESTREMA SINTESI



ALGORITMI DI MACHINE LEARNING

ESTREMA SINTESI

ALGORITMI DI MACHINE LEARNING

MEMENTO

- Non utilizzate approcci di Machine Learning per problemi sui quali non avete a disposizione sufficienti esempi per il Training e il Test.
- Collezionare esempi (ed etichettarli) può richiedere ingenti sforzi, a meno che non siate in grado di reperire i pattern in rete, e/o non possiate pagare qualcuno per collezionarli/etichettarli al posto vostro (*es. Crowdsourcing via Amazon Mechanical Turk per ImageNet*).
- Collezionate pattern rappresentativi del problema da risolvere e distribuiteli adeguatamente tra Train, Valid e Test.

Tool per il Machine Learning

Nel corso degli anni ricercatori, sviluppatori indipendenti e imprese hanno sviluppato numerosi tool software (librerie, framework, simulatori), gran parte dei quali open-source.

La scelta del tool (e relativo linguaggio di programmazione) dipende dagli obiettivi del progetto e dalla preferenze dello sviluppatore.

Tra i tool più noti, ricordiamo:

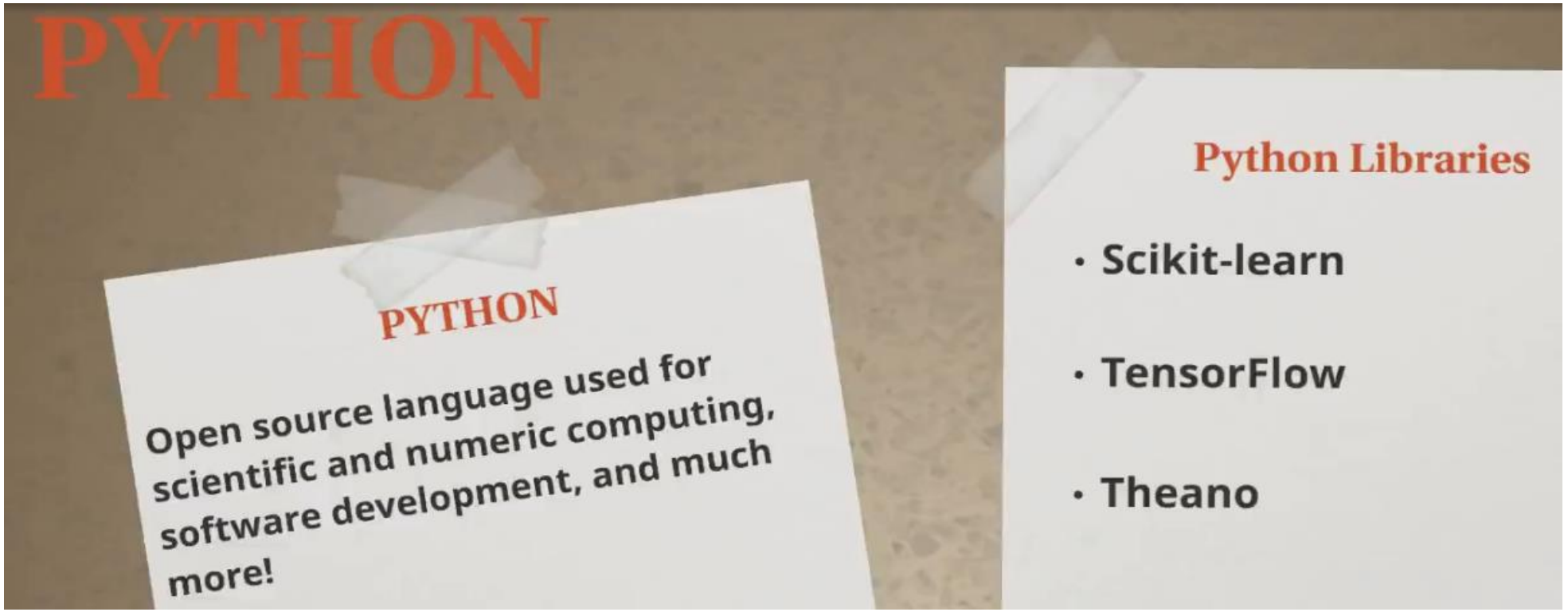
- Scikit-learn* (Python). *General Purpose*
- OpenCV (C++). *Molto utilizzato in ambito Visione Artificiale*
- Weka (Java). *Molto utilizzato in ambito Data Mining*
- R and Caret. *Dalla Statistica al Machine Learning*

Per un elenco più dettagliato:

https://github.com/josephmisiti/awesome-machine-learning.

SOFTWARE PER MACHINE LEARNING

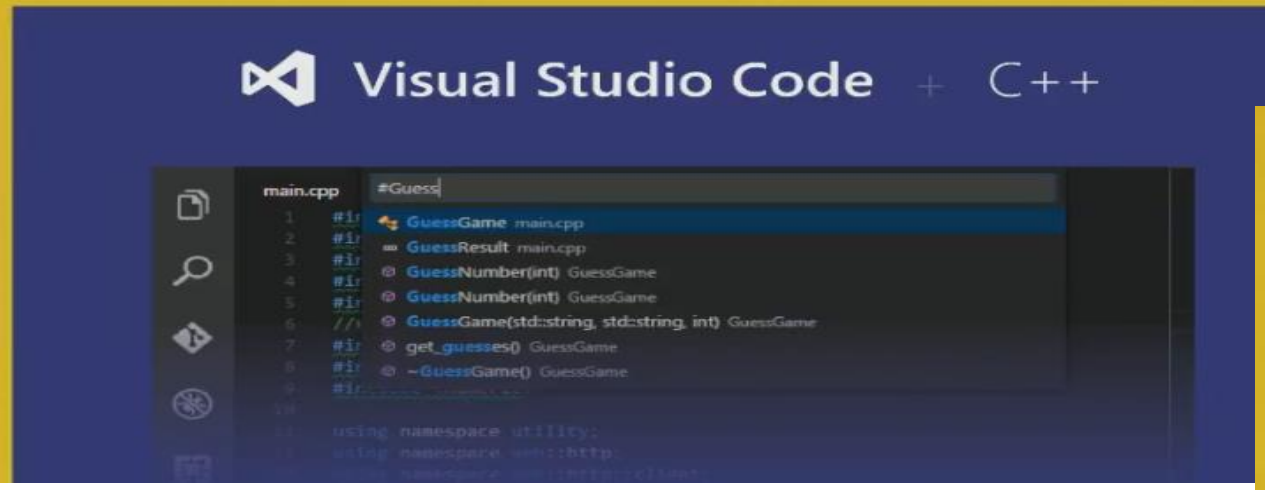
SOFTWARE PER MACHINE LEARNING



SOFTWARE PER MACHINE LEARNING

C++

General-purpose programming language with imperative, object-oriented, and generic programming features.



Common Libraries

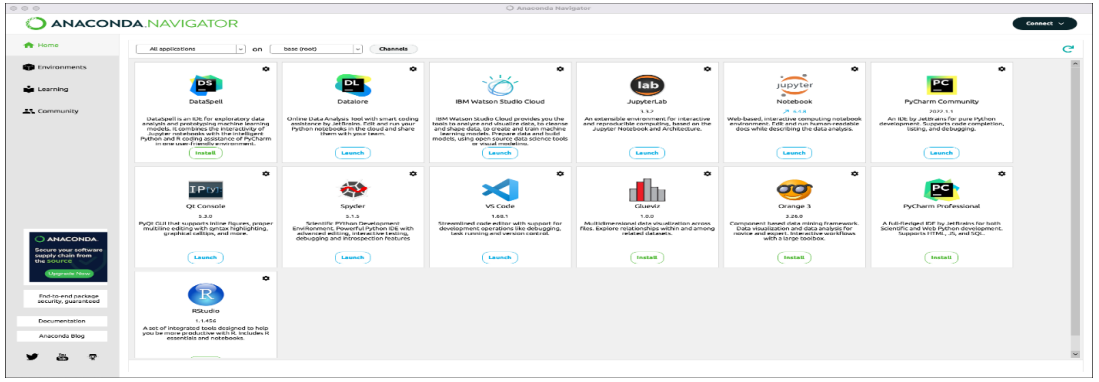
- Mlpack
- Shark
- Shogun

ALGORITMO DI MACHINE LEARNING

APPLICAZIONE PRATICA

Il sistema di ML utilizza quale *DATA set* quanto contenuto in un file, prodotto da sistemi di Cyber Defence,
posti a perimetro di una rete di un sistema aziendale e successivamente rielaborato da personale umano.
Il file di dati reali, denominato file A, è stato prodotto in formato .csv e contiene circa 14.000 righe, ciascuna descrivente un malware.
Utilizzeremo un CLASSIFICATORE SUPERVISIONATO
Ciascun *malware* è descritto da 79 features ripartite in (dettaglio nella grafica a colori):

- Indicazione delle 17 librerie che il malware utilizza per interagire con il Sistema Operativo;
- Indicazione di 37 API che vengono chiamate dal malware
- Estensione del file in cui è inserito il malware: sono considerate 11 diverse estensioni;
- Nome del malware: ne sono raccolti 14 diverse tipologie



python™

ALGORITMO DI MACHINE LEARNING

APPLICAZIONE PRATICA

- Indicazione delle 17 librerie che il malware utilizza per interagire con il Sistema Operativo;
- Indicazione di 37 API che vengono chiamate dal malware
- Estensione del file in cui è inserito il malware: sono considerate 11 diverse estensioni;
- Nome del malware: ne sono raccolti 14 diverse tipologie

```
Keys of malware:
Index([ appexception', 'apicall', 'codeinjection', 'dll-loaded', 'doc_summary',
       'exploitcode', 'file', 'folder', 'heapspraying', 'hiddenproc',
       'malicious-alert', 'mutex', 'network', 'process', 'regkey', 'thread',
       'wmiquery', 'CheckRemoteDebuggerPresent', 'ClipboardFormatListener',
       'ClipboardSequenceNumber', 'CLSIDFromString', 'CryptAcquireContext',
       'EncryptMessage', 'EnumWindows', 'ExitProcess', 'FindWindowEx',
       'FindWindow', 'GetClipboardData', 'GetComputerName',
       'GetComputerNameEx', 'GetDesktopWindow', 'GetForegroundWindow',
       'GetLocalTime', 'GetSystemDefaultLangID', 'GetSystemDirectory',
       'GetSystemTime', 'GetTokenInformation', 'GetVersionEx',
       'GetVolumeNameForVolumeMountPoint', 'IcmpSendEcho', 'IsDebuggerPresent',
       'MessageBox', 'NtAdjustPrivilegesToken', 'RegisterRawInputDevices',
       'SetClipboardData', 'SetClipboardViewer', 'SetProcessDEPPolicy',
       'SetTimer', 'SetWindowsHookEx', 'ShellExecute', 'Sleep', 'SleepEx',
       'StartService', 'SystemTimeToFileTime', 'xls', 'doc', 'exe', 'xlsx',
       'zip', 'xlsm', 'docx', 'xlsb', 'pif', 'scr', 'altra_estensione',
       'agent_tesla', 'carrotbat', 'icedid', 'asyncrat', 'ave_maria', 'emotet',
       'formbook', 'gozi', 'lokibot', 'nanocore', 'netsky', 'remcos', 'qakbot',
       'autoit'],
      dtype='object')
(14257, 79)
```

ALGORITMO DI MACHINE LEARNING

APPLICAZIONE PRATICA

```
In [2]: from sklearn import datasets
        from sklearn.svm import SVC
        from sklearn.multiclass import OneVsOneClassifier

In [3]: #IMPORTAZIONE FILE CSV ESTERNO, NON COMPRESO IN SKLEARN
        data_malware = pd.read_csv("malspam_def14k_new.csv")

In [4]: #COMANDO PER IMPORTARE DA SKLEARN LA FUNZIONE PER SPLITTARE IL CAMPIONE
        from sklearn.model_selection import train_test_split

In [5]: #CREO I SET DI TEST E TRAIN AL 80%
        train,test=train_test_split(data_malware,test_size=0.2)
```

In[2]: importo il modello del classificatore;

In[3]: importo il file di dati in formato .csv;

In[4]: importo la funzione per splittare i dati nelle due componenti: Training e Test.

In[5]: tramite la funzione importata al punto In[4] creo i due set di dati, Training e Test, secondo ripartizione che prevede 80% dei dati da utilizzare per il training e il restante 20% per i test

ALGORITMO DI MACHINE LEARNING

APPLICAZIONE PRATICA

In[12]: in questa parte viene effettuata la fase di addestramento del classificatore utilizzando i dati *xtrain* e *ytrain* ottenuti dalla funzione di *splitting* precedentemente descritta;

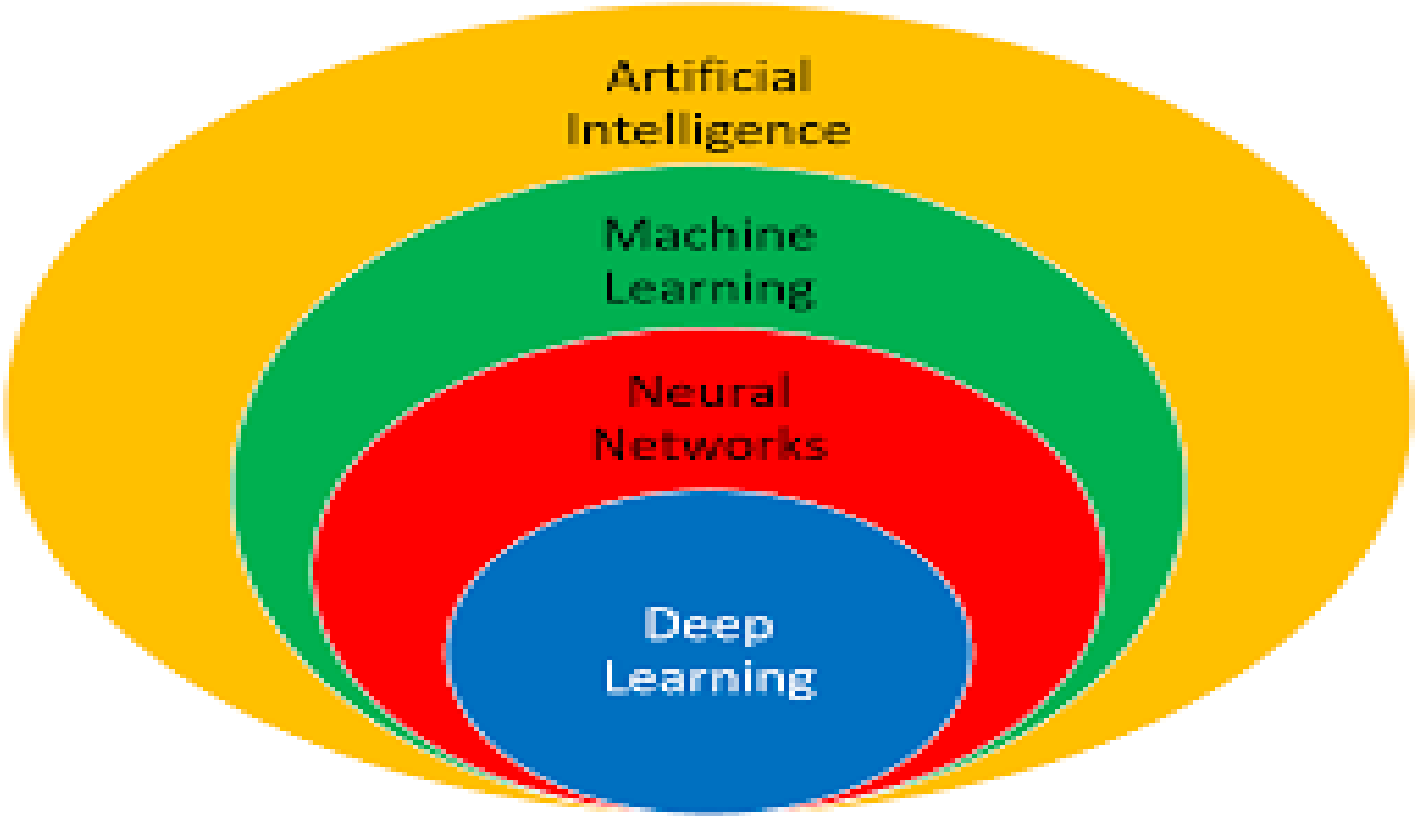
Out[12]: rappresenta *l'output* del comando precedente ossia il classificatore addestrato;

```
In [8]: xtrain, xtest, ytrain, ytest = train_test_split(x,y, test_size = 0.15)

In [9]: svc = SVC()

In [10]: o_vs_o = OneVsOneClassifier(svc)

In [12]: o_vs_o.fit(xtrain, ytrain)

Out[12]: OneVsOneClassifier(estimator=SVC())
```

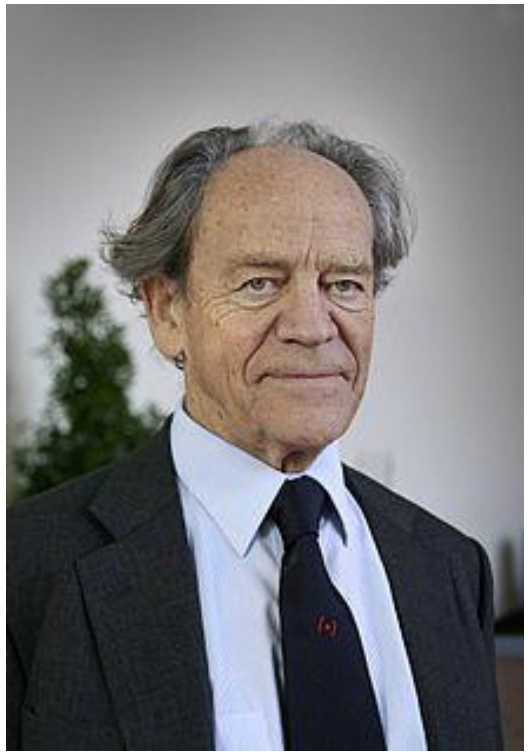
```
In [13]: ypred = o_vs_o.predict(xtest)
```

```
In [15]: # Utilizzo il modulo metrics per il calcolo, per poi stampare il risultato
         print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(ytest, ypred))

ACCURACY OF THE MODEL:  0.9859747545582047
```

DEEP LEARNING

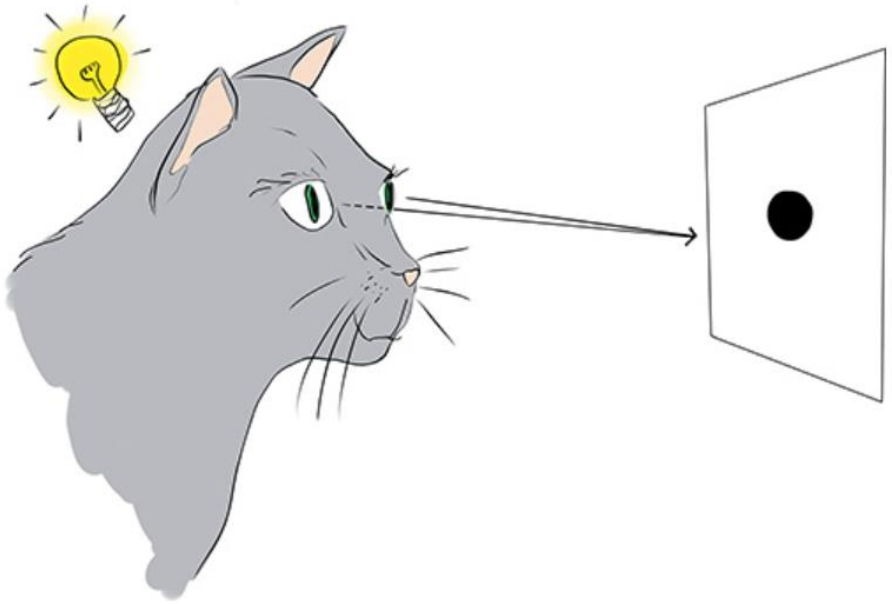


DEEP LEARNING

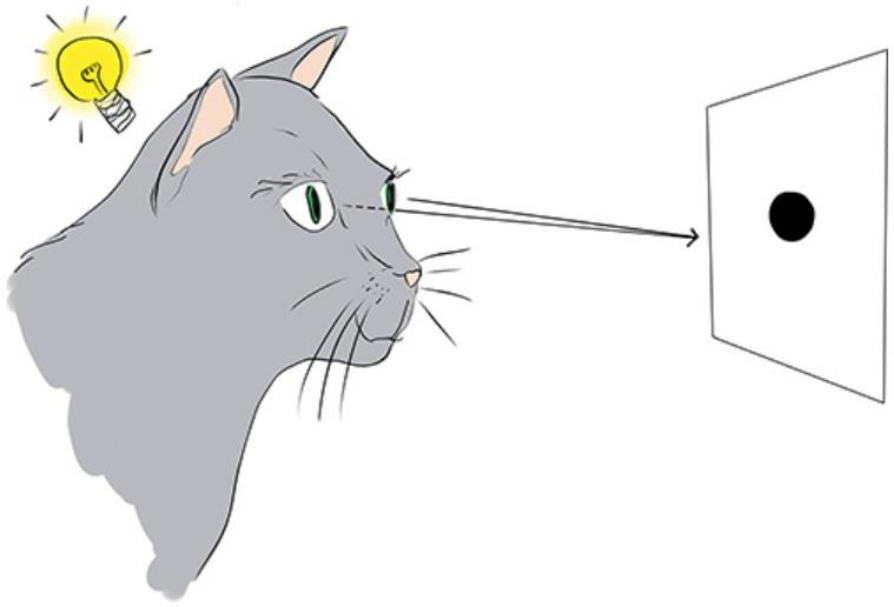


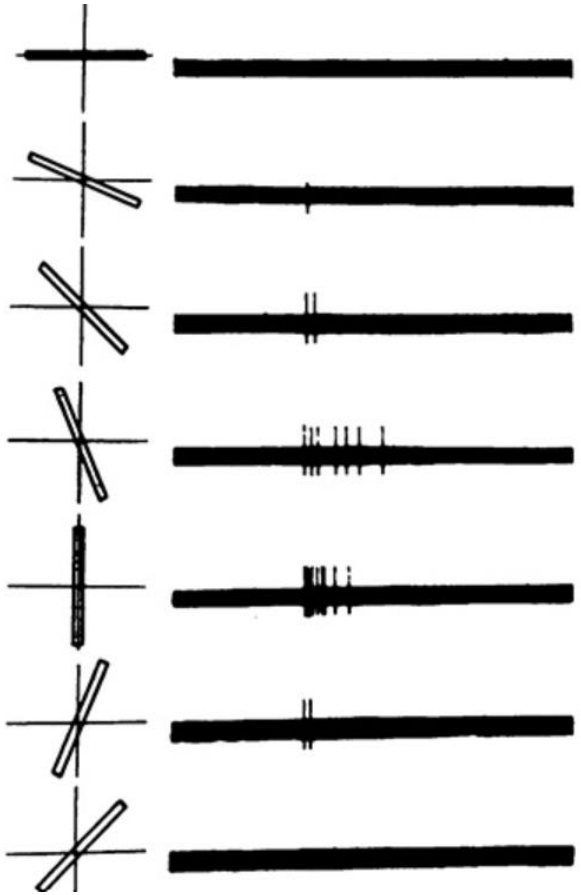
Torsten Wiesel

David Hubel

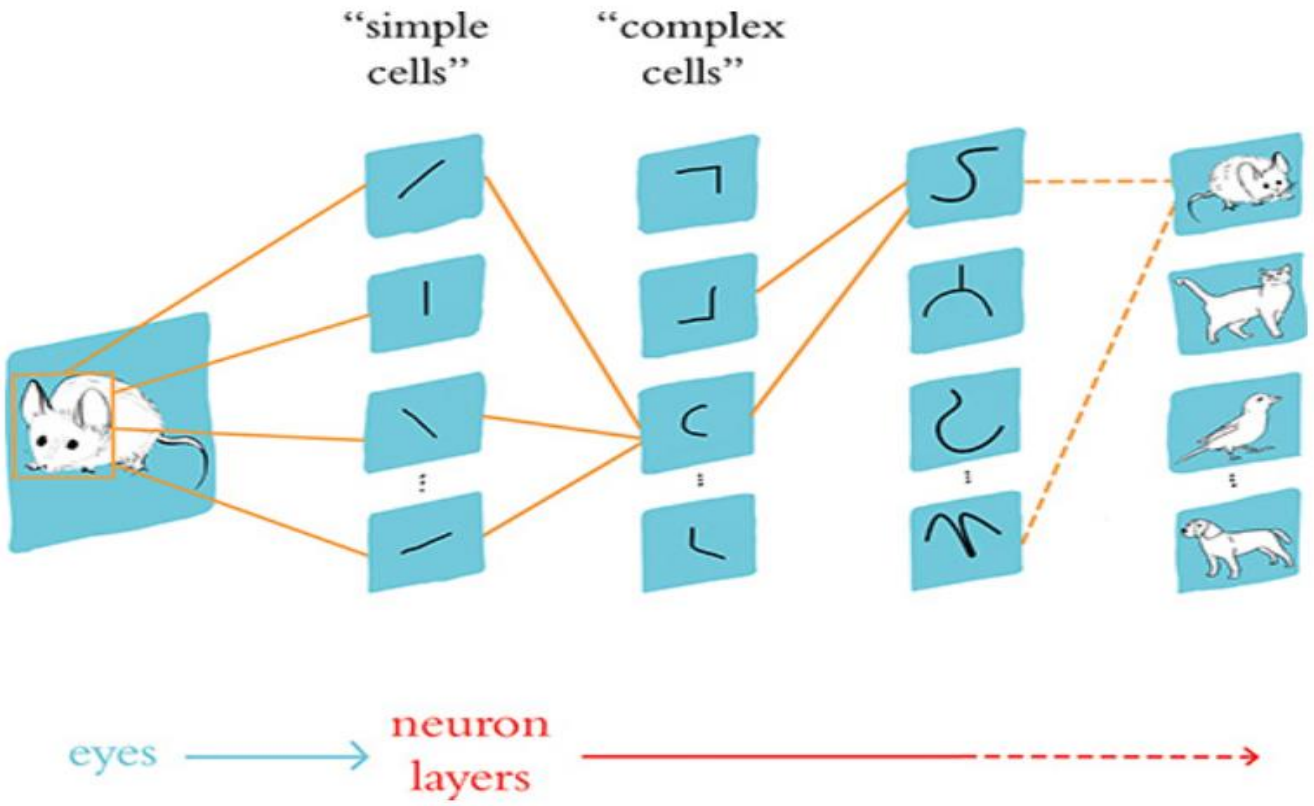


DEEP LEARNING





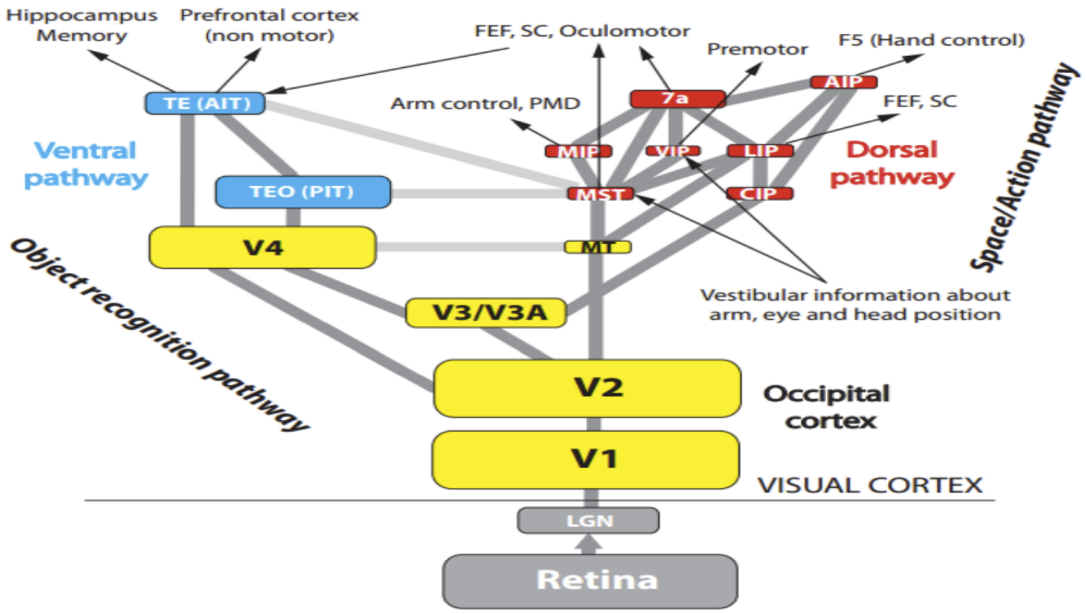
DEEP LEARNING



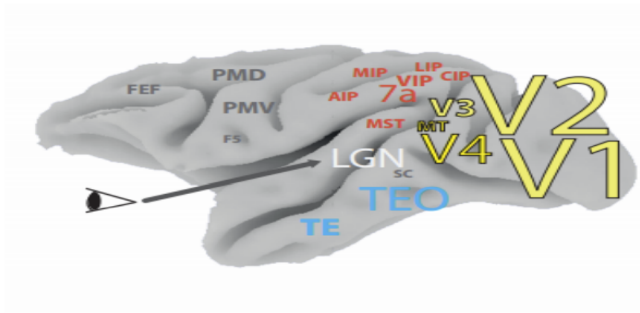
Perchè deep?

Con il termine DNN (Deep Neural Network) si denotano reti «profonde» composte da molti livelli (almeno 2 hidden) organizzati gerarchicamente.

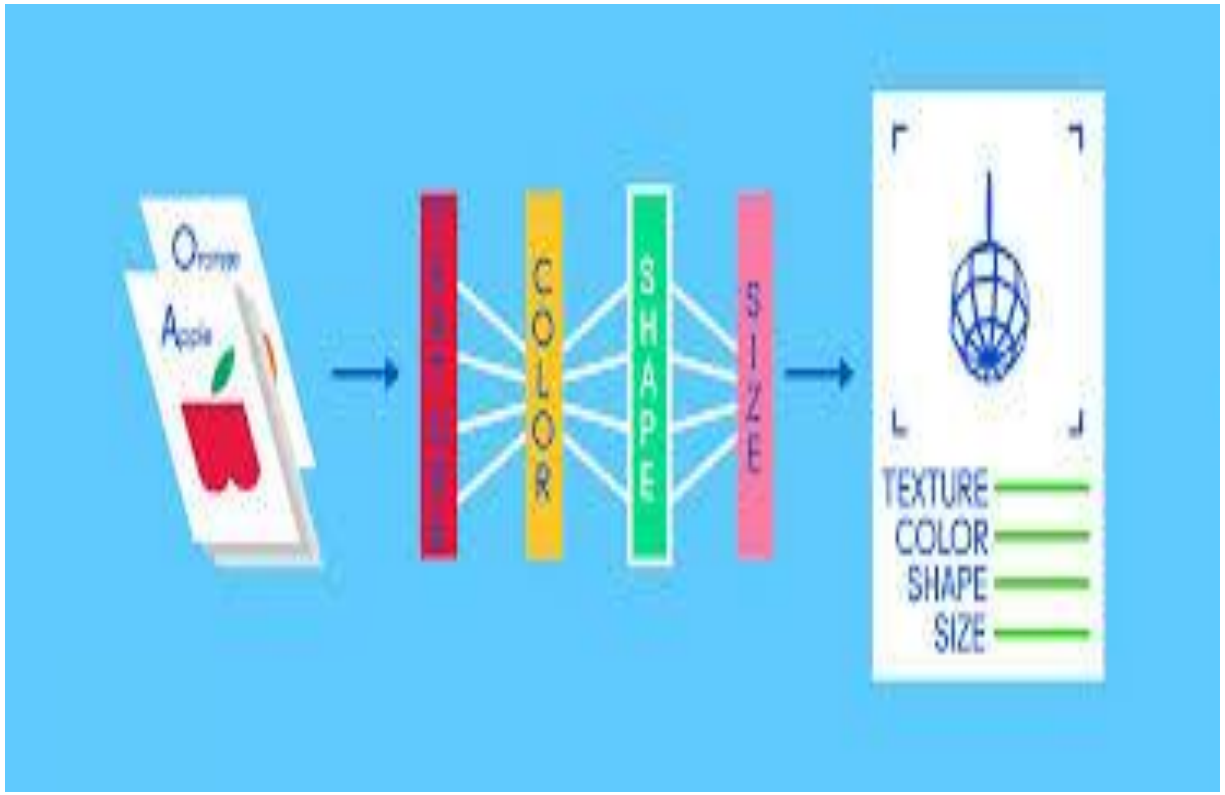
- Il nostro sistema visivo opera su una gerarchia di livelli (deep):



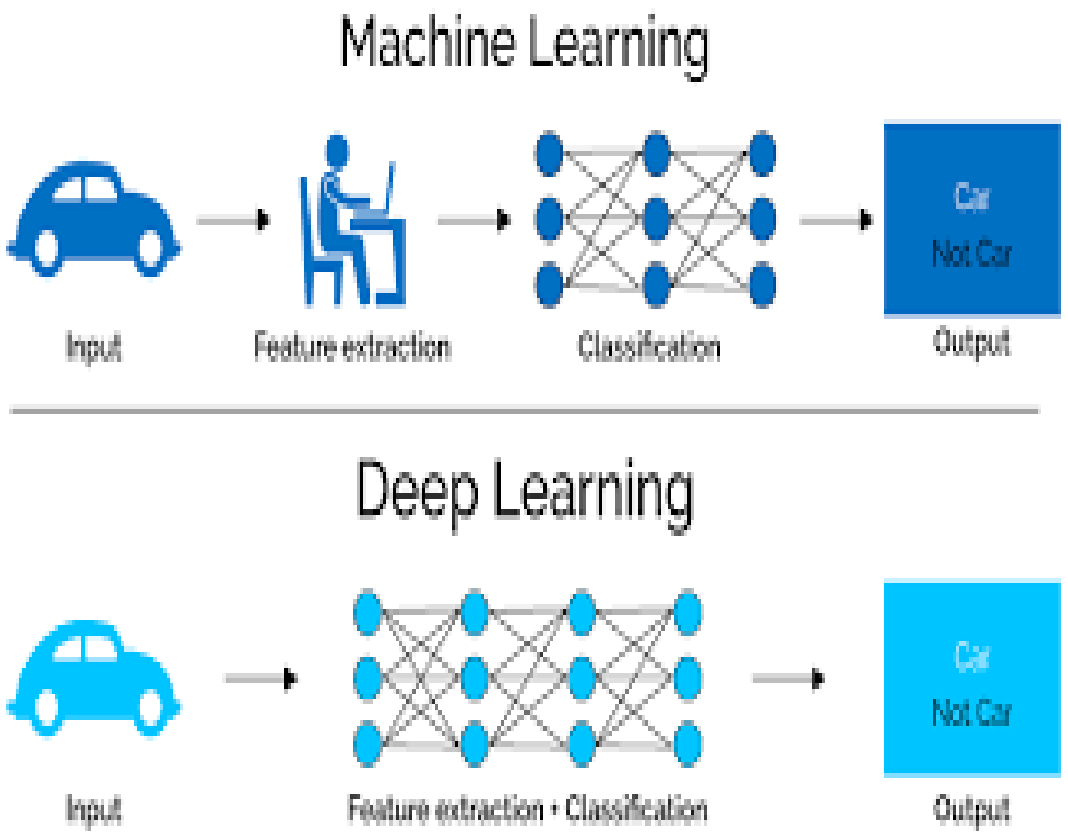
[Kruger et al. 2013]

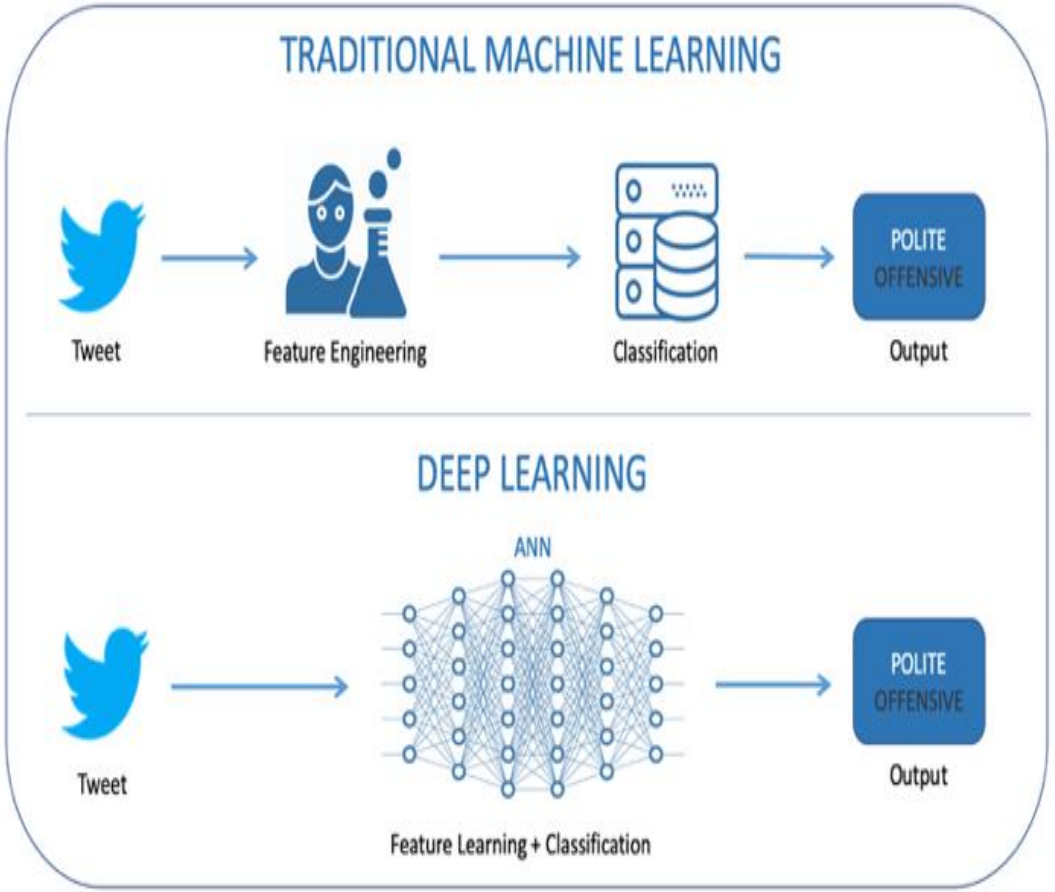


- L’organizzazione gerarchica consente di condividere e riusare informazioni (un po' come la programmazione strutturata). Lungo la gerarchia è possibile selezionare feature specifiche e scartare dettagli inutili (al fine di massimizzare l’invarianza).

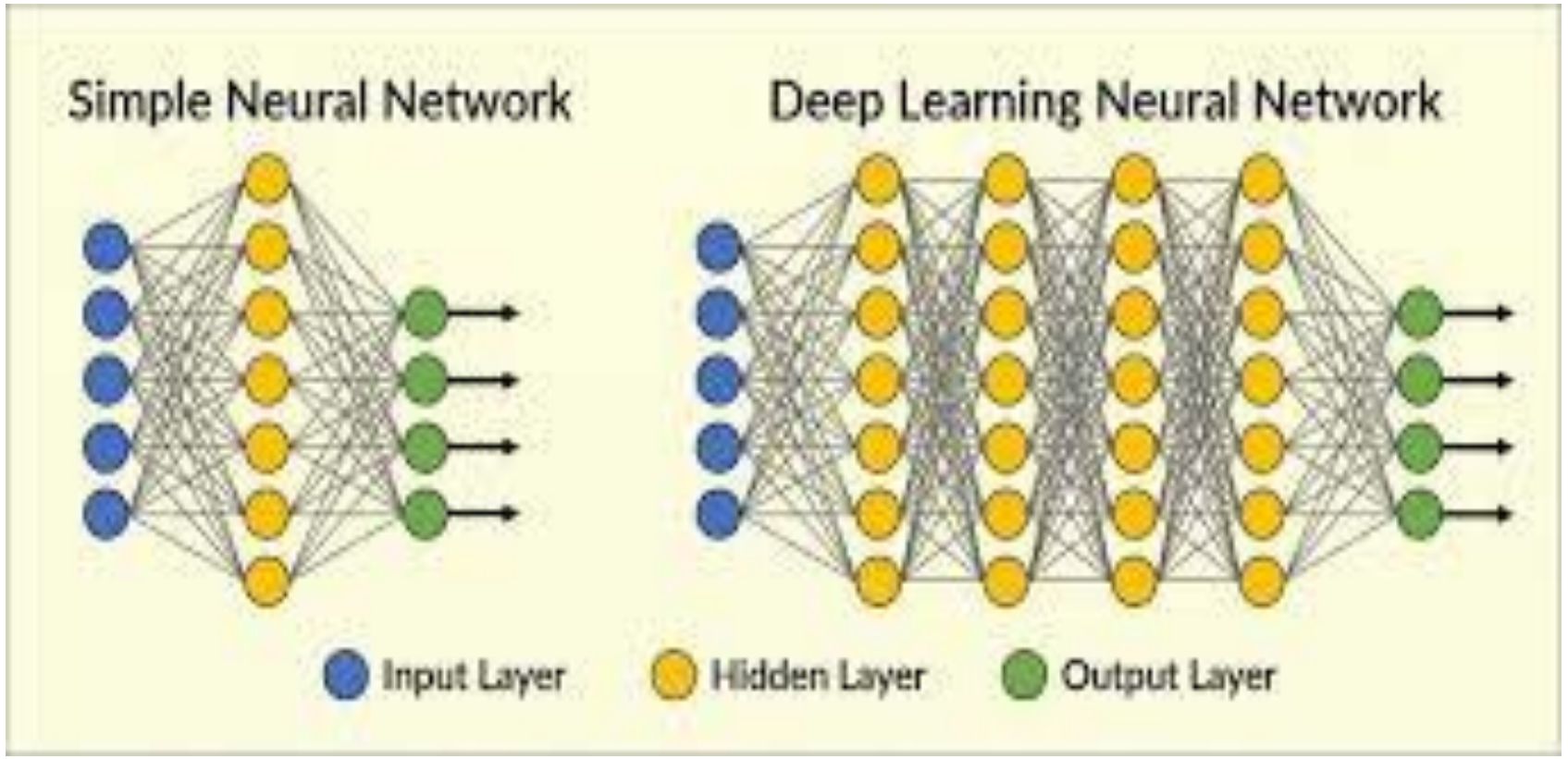


MACHINE LEARNING vs DEEP LEARNING





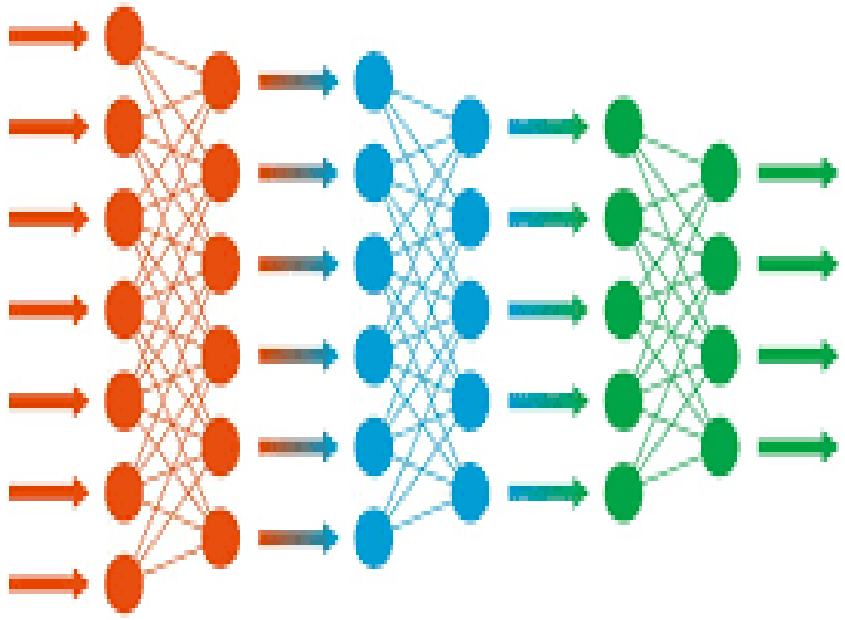
DEEP LEARNING



DEEP LEARNING

ma quanto deep?

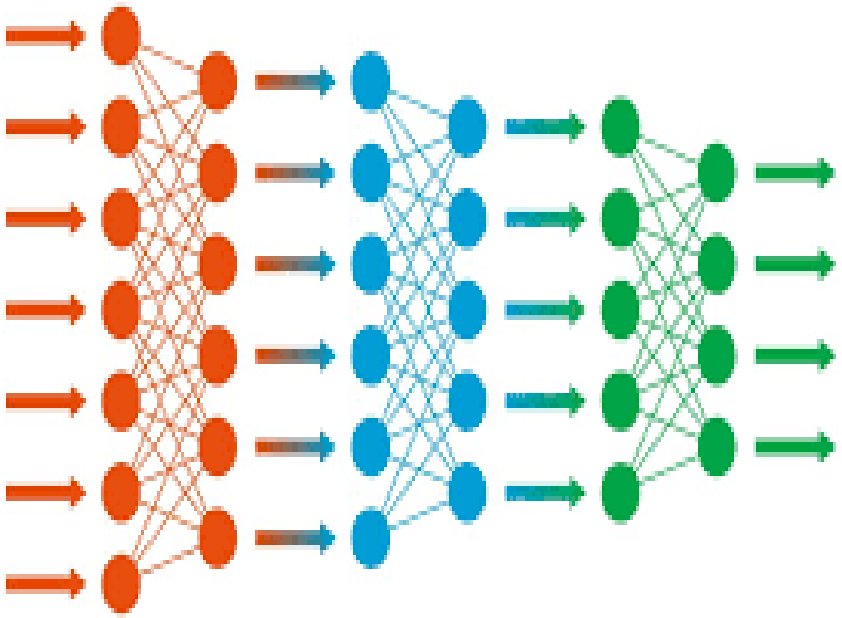
- Le DNN oggi maggiormente utilizzate consistono di un numero di livelli compreso tra 7 e 50.
  - Reti più profonde (100 livelli e oltre) hanno dimostrato di poter garantire prestazioni leggermente migliori, a discapito però dell'efficienza.



DEEP LEARNING

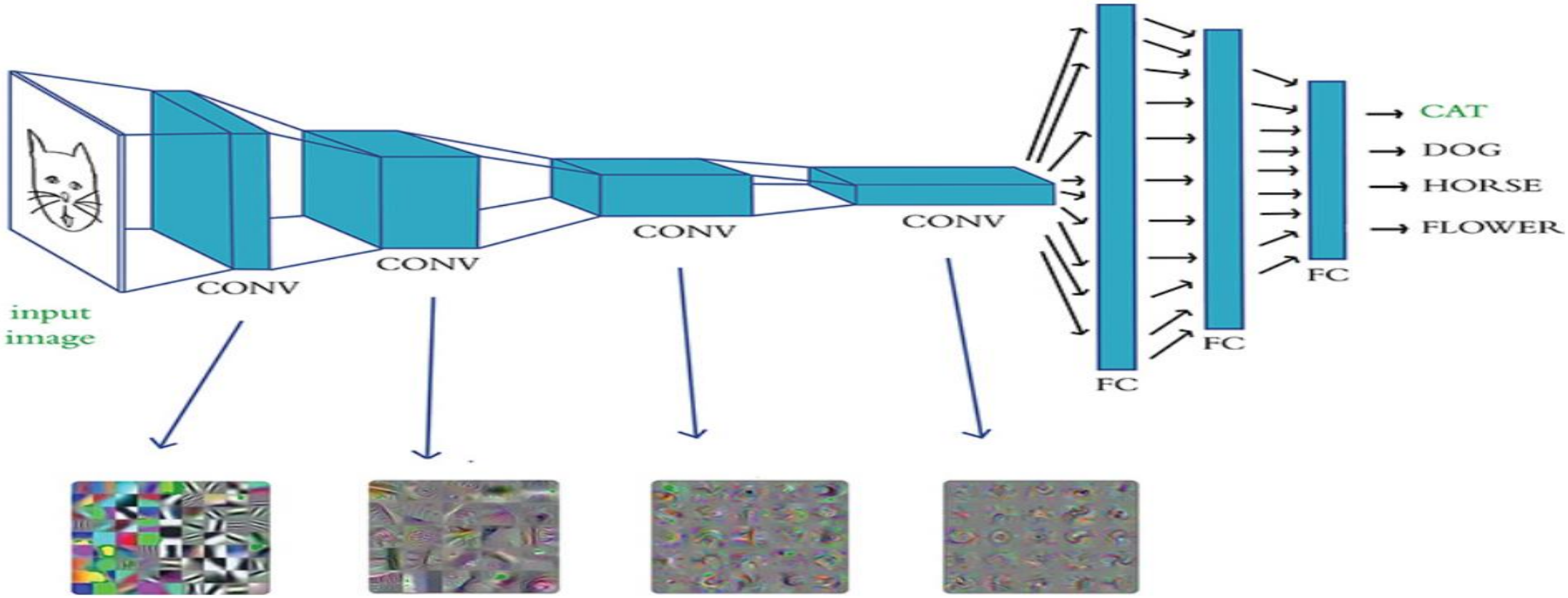
Livelli e Complessità

- La profondità (numero di livelli) è solo uno dei fattori di complessità. Numero di neuroni, di connessioni e di pesi caratterizzano altresì la complessità di una DNN.
- Maggiore è il numero di pesi (ovvero di parametri da apprendere) maggiore è la complessità del training. Al tempo stesso un elevato numero di neuroni (e connessioni) rende forward e back propagation più costosi, poiché aumenta il numero (G-Ops) di operazioni necessarie.
  - AlexNet: 8 livelli, 650K neuroni e 60M parametri
  - VGG-16: 16 livelli, 15M neuroni e 140M parametri
  - Corteccia umana: 10^{11} neuroni e 10^{14} sinapsi

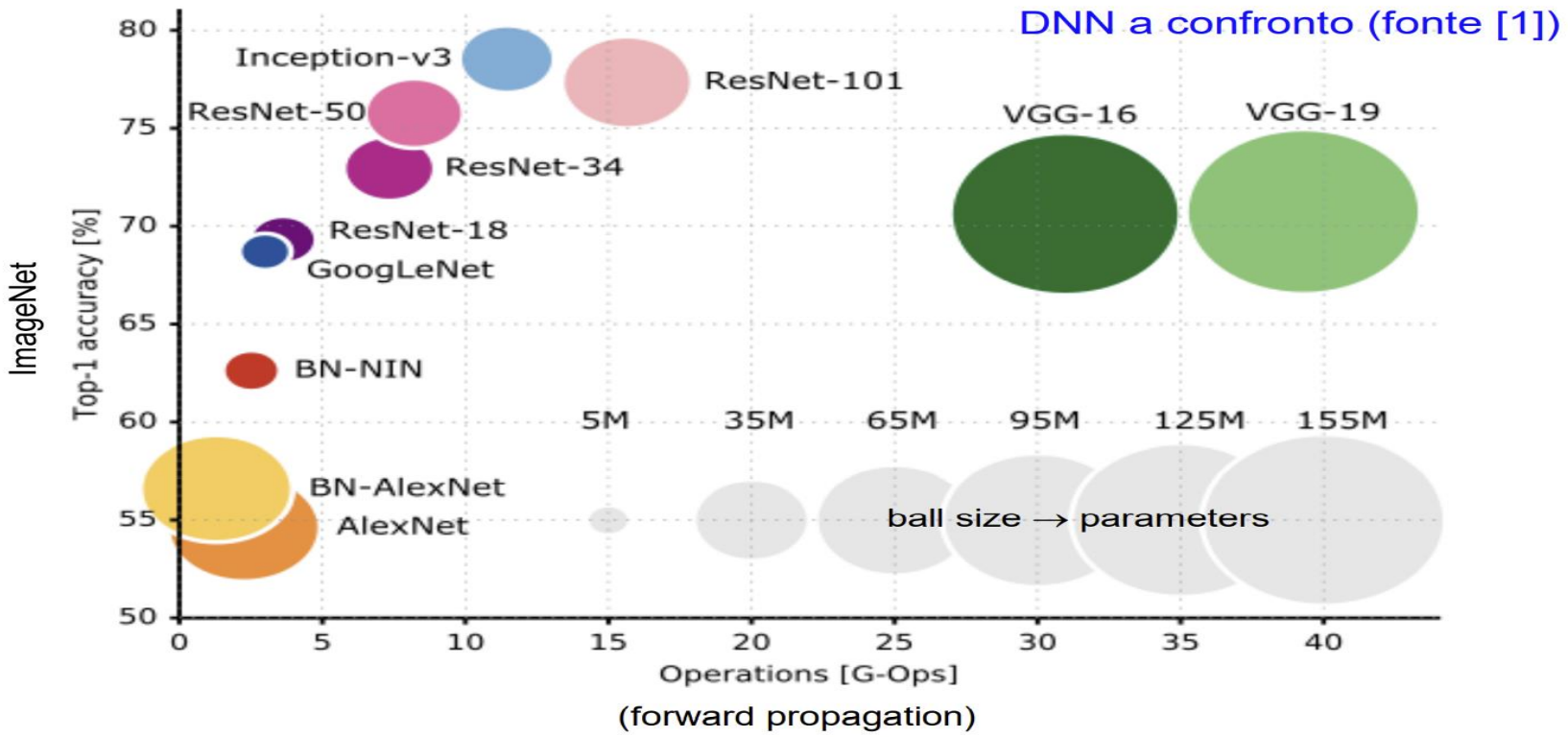


DEEP LEARNING

ALEXA NET



DEEP LEARNING



[1] Canziani et al. 2016, *An Analysis of Deep Neural Network Models for Practical Applications*

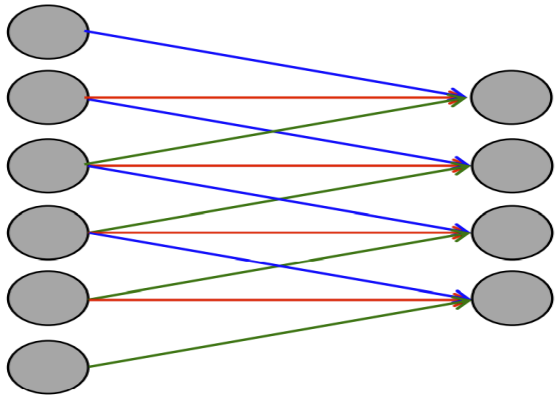
Principali tipologie di DNN

- Modelli feedforward «discriminativi» per la classificazione (o regressione) con training prevalentemente supervisionato:
  - CNN - Convolutional Neural Network (o ConvNet)
  - FC DNN - Fully Connected DNN (MLP con almeno due livelli hidden)
  - HTM - Hierarchical Temporal Memory
- Training non superivisionato (modelli «generativi» addestrati a ricostruire l'input, utili per pre-training di altri modelli e per produrre feature salienti):
  - Stacked (de-noising) Auto-Encoders
  - RBM - Restricted Boltzmann Machine
  - DBN - Deep Belief Networks

DA MULTILAYER PERCEPTRON (MLP) A CONVOLUTIONAL NEURAL NETWORK

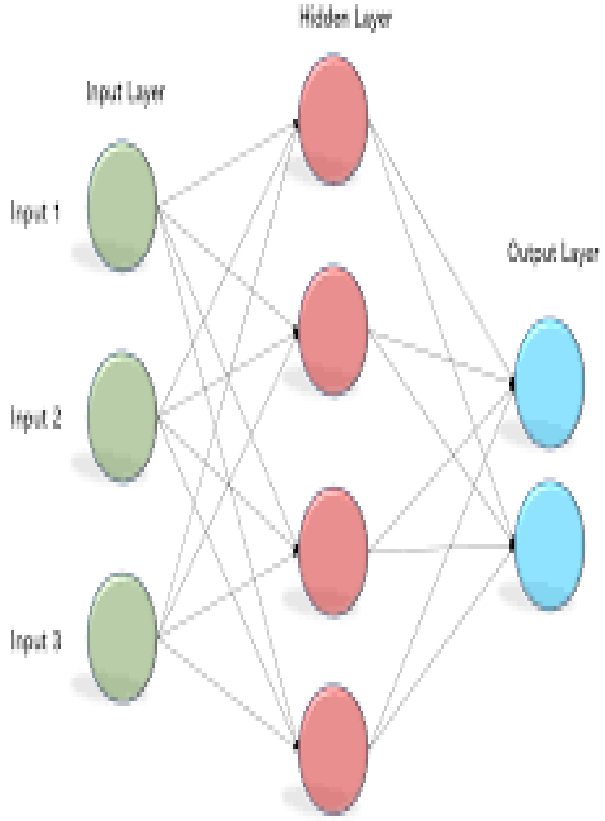
Convolutional Neural Networks (CNN) introdotte da LeCun et al., a partire dal 1998. Le principali differenze rispetto a MLP:

- processing locale: i neuroni sono connessi solo localmente ai neuroni del livello precedente. Ogni neurone esegue quindi un'elaborazione locale. Forte riduzione numero di connessioni.
- pesi condivisi: i pesi sono condivisi a gruppi. Neuroni diversi dello stesso livello eseguono lo stesso tipo di elaborazione su porzioni diverse dell'input. Forte riduzione numero di pesi.



Esempio: ciascuno dei 4 neuroni a destra è connesso solo a 3 neuroni del livello precedente. I pesi sono condivisi (stesso colore stesso peso). In totale 12 connessioni e 3 pesi contro le 24 connessioni + 24 pesi di una equivalente porzione di MLP.

MLP



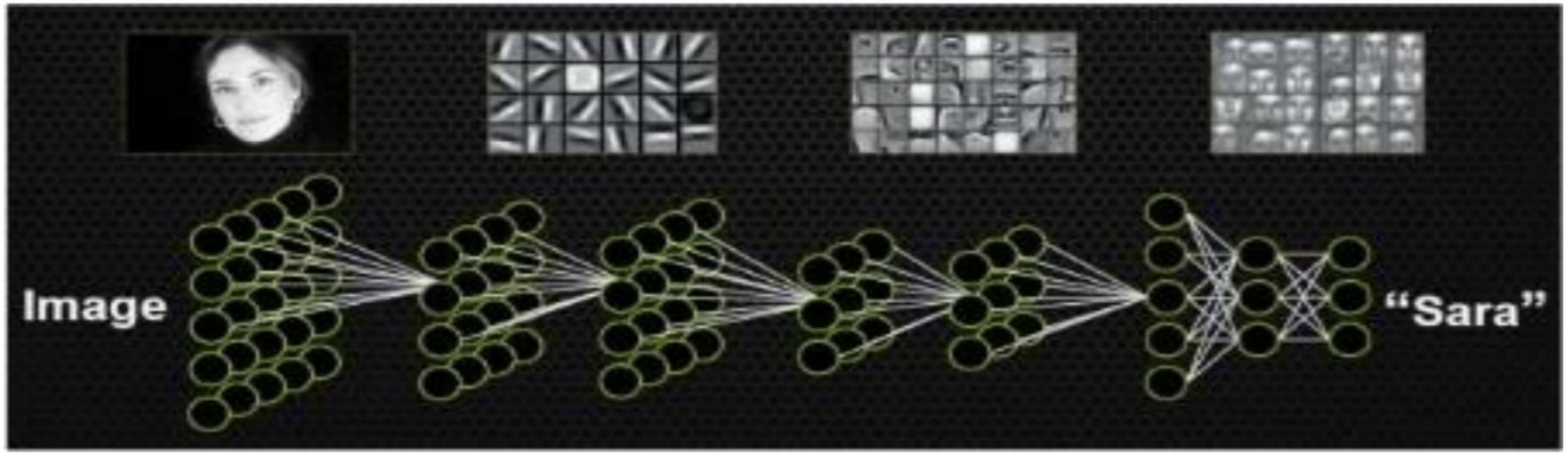
CONVOLUTIONAL NEURAL NETWORK

CNN: Architettura

Esplicitamente progettate per processare immagini, per le quali elaborazione locale, pesi condivisi, e pooling non solo semplificano il modello, ma lo rendono più efficace rispetto a modelli fully connected. Possono essere utilizzate anche per altri tipi di pattern (es. speech).

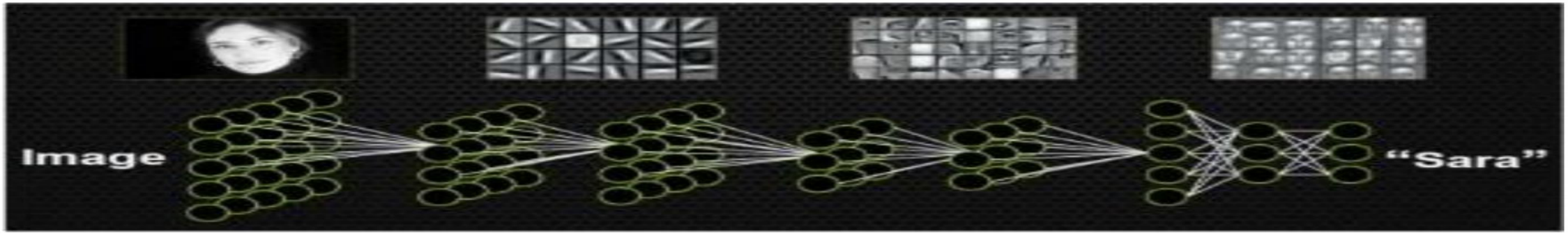
CONVOLUTIONAL NEURAL NETWORK

- Architettura: una CNN è composta da una gerarchia di livelli. Il livello di input è direttamente collegato ai pixel dell'immagine, gli ultimi livelli sono generalmente fully-connected e operano come un classificatore MLP, mentre nei livelli intermedi si utilizzano connessioni locali e pesi condivisi.



CONVOLUTIONAL NEURAL NETWORK

- Architettura: una CNN è composta da una gerarchia di livelli. Il livello di input è direttamente collegato ai pixel dell'immagine, gli ultimi livelli sono generalmente fully-connected e operano come un classificatore MLP, mentre nei livelli intermedi si utilizzano connessioni locali e pesi condivisi.



- Il campo visivo (receptive field) dei neuroni aumenta muovendosi verso l'alto nella gerarchia.
- Le connessioni locali e condivise fanno sì che i neuroni processino nello stesso modo porzioni diverse dell'immagine. Si tratta di un comportamento desiderato, in quando regioni diverse del campo visivo contengono lo stesso tipo di informazioni (bordi, spigoli, porzioni di oggetti, ecc.).
- Visualizzazione 3D di una CNN: *http://www.cs.cmu.edu/~aharley/vis/*

CONVOLUTIONAL NEURAL NETWORK

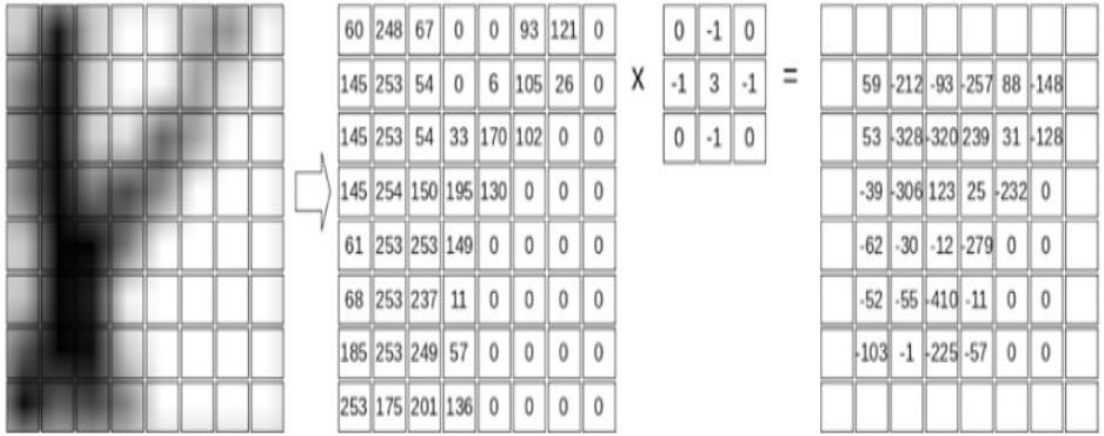


Figure 12-1: Convolving a kernel with an image

$$\begin{bmatrix} 248 & 67 & 0 \\ 253 & 54 & 0 \\ 253 & 54 & 33 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -67 & 0 \\ -253 & 162 & 0 \\ 0 & -54 & 0 \end{bmatrix}$$

212

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 60 & 248 & 67 \\ 145 & 253 & 54 \\ 145 & 253 & 54 \end{bmatrix}$$

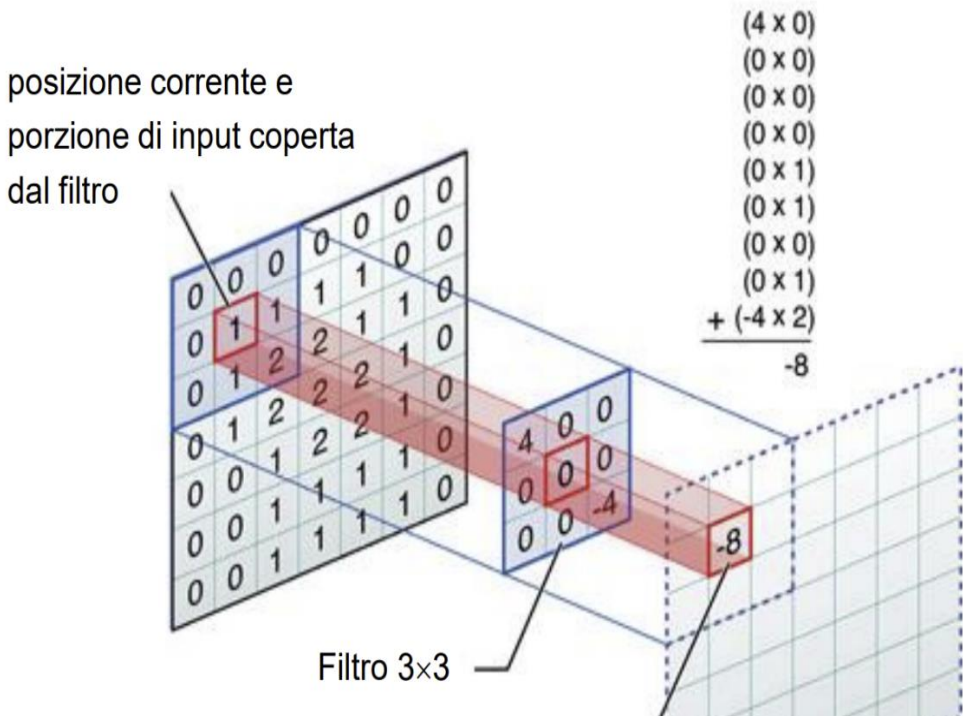
$$\begin{bmatrix} 60 & 248 & 67 \\ 145 & 253 & 54 \\ 145 & 253 & 54 \end{bmatrix} \times \begin{bmatrix} 0 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -248 & 0 \\ -145 & 759 & -54 \\ 0 & -253 & 0 \end{bmatrix}$$

0 + (–248) + 0 + (–145) + 759 + (–54) + 0 + (–253) + 0 = 59

CONVOLUTIONAL NEURAL NETWORK

- La convoluzione è una delle più importanti operazioni di image processing attraverso la quale si applicano filtri digitali.
- Un filtro digitale (un piccola maschera 2D di pesi) è fatta scorrere sulle diverse posizioni di input; per ogni posizione viene generato un valore di output, eseguendo il prodotto scalare tra la maschera e la porzione dell'input coperta (entrambi trattati come vettori).

CONVOLUTIONAL NEURAL NETWORK



: pesi del filtro in

rosso, porzione coperta in giallo

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

4		

CONVOLUTIONAL NEURAL NETWORK

Esempio Filtri a immagine

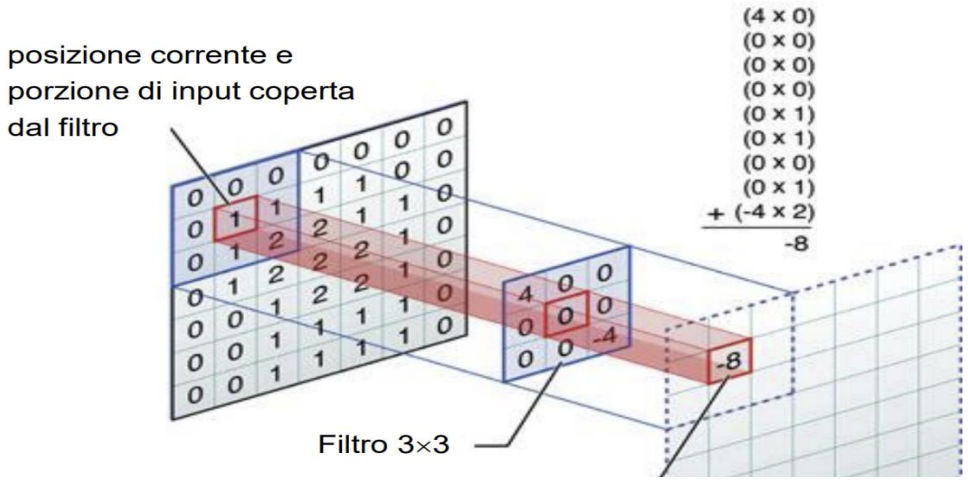


Immagine input

Filtro

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Immagine output

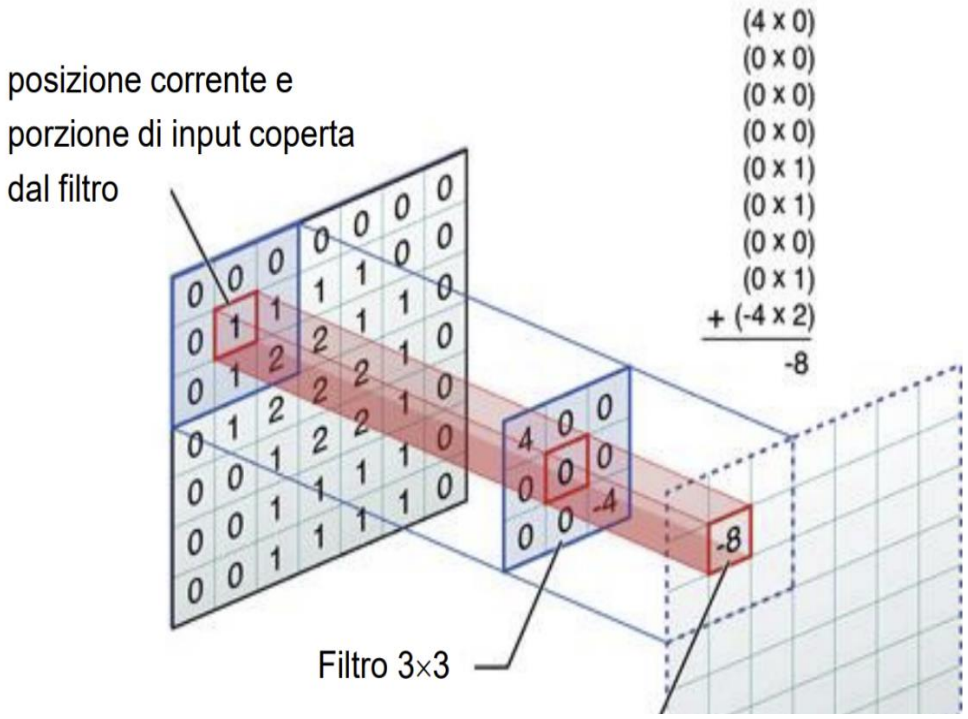
$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

4		

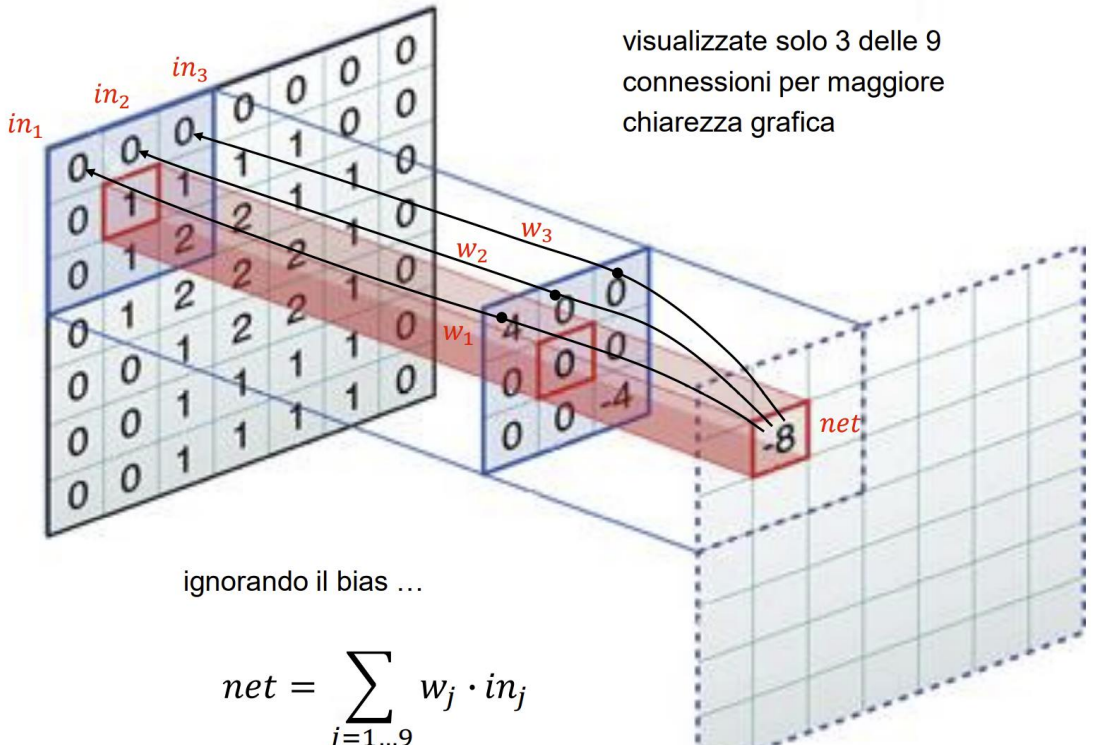
$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

CONVOLUTIONAL NEURAL NETWORK

CONVOLUZIONE



OUT



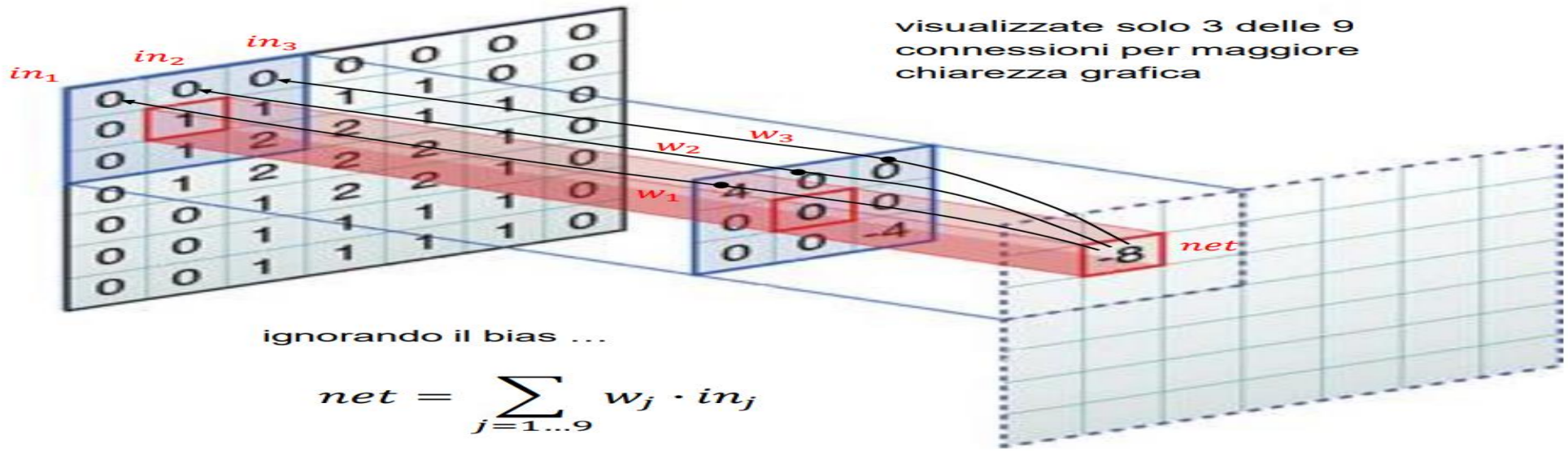
visualizzate solo 3 delle 9 connessioni per maggiore chiarezza grafica

ignorando il bias …

$$net = \sum_{j=1..9} w_j \cdot in_j$$

CONVOLUTIONAL NEURAL NETWORK

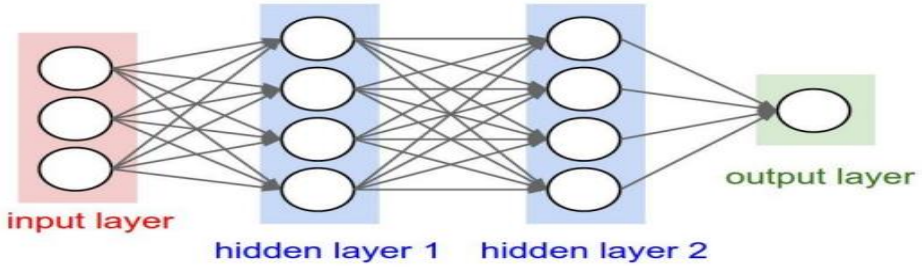
- Consideriamo i pixel come neuroni e le due immagini di input e di output come livelli successivi di una rete. Dato un filtro 3×3, se colleghiamo un neurone ai 9 neuroni che esso «copre» nel livello precedente, e utilizziamo i pesi del filtro come pesi delle connessioni w, notiamo che un classico neurone (di una MLP) esegue di fatto una convoluzione.



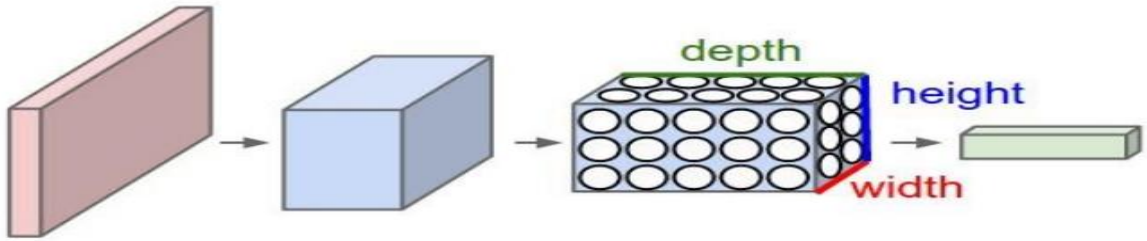
ignorando il bias ...

$$net = \sum_{j=1\ldots9} w_j \cdot in_j$$

CONVOLUTIONAL NEURAL NETWORK



MLP: organizzazione lineare dei neuroni nei livelli

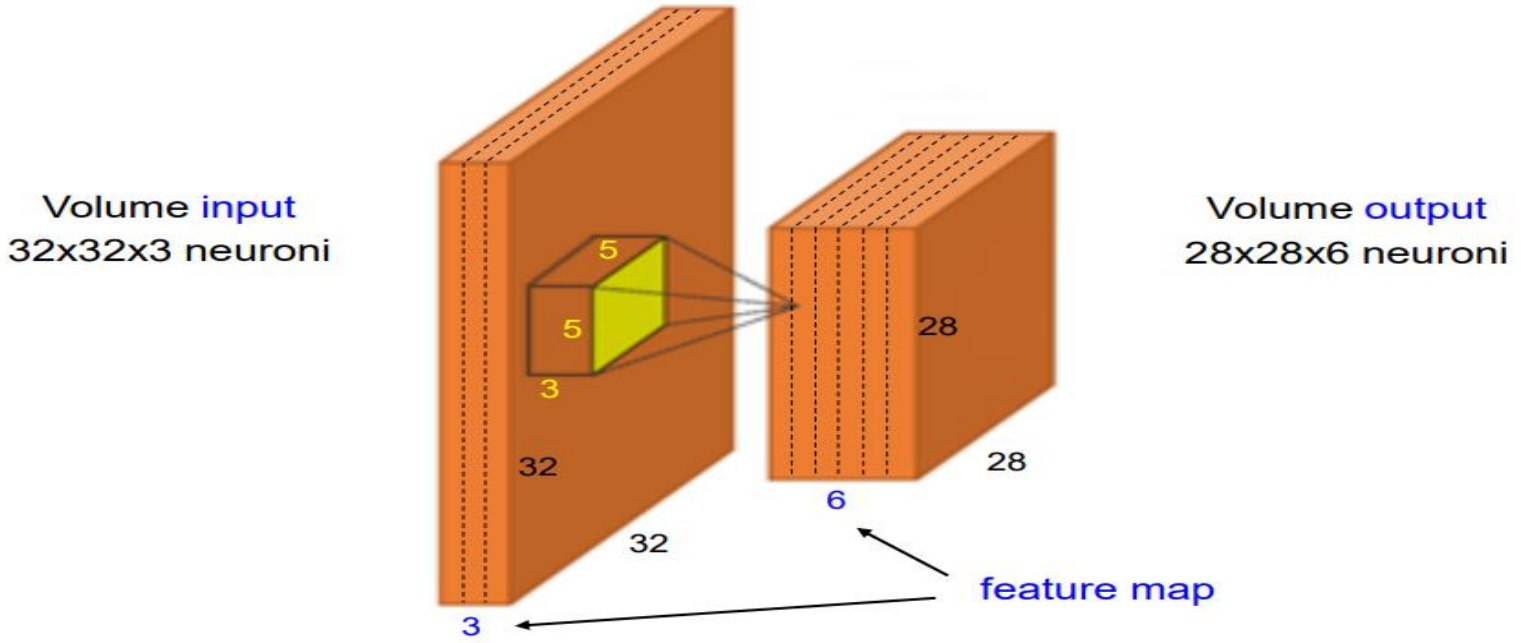


CNN: i livelli sono organizzati come griglie 3D di neuroni

sui piani width - height si conserva l'organizzazione spaziale «retinotipica» dell'immagine di input.

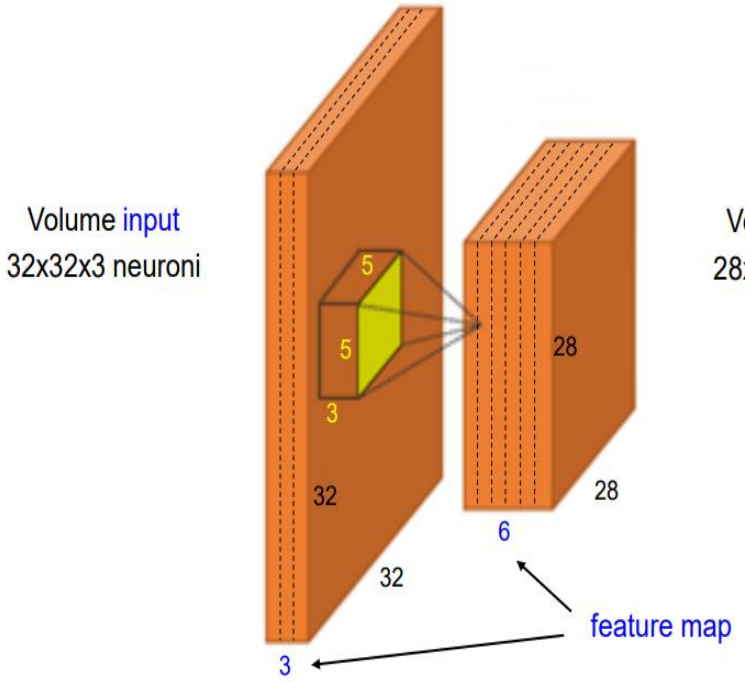
CONVOLUTIONAL NEURAL NETWORK – 3 D

- Il filtro opera su una porzione del volume di input. Nell'esempio ogni neurone del volume di output è connesso a 5x5x3=75 neuroni del livello precedente.



CONVOLUTIONAL NEURAL NETWORK – 3 D

- Il filtro opera su una porzione del volume di input. Nell'esempio ogni neurone del volume di output è connesso a 5x5x3= neuroni del livello precedente.



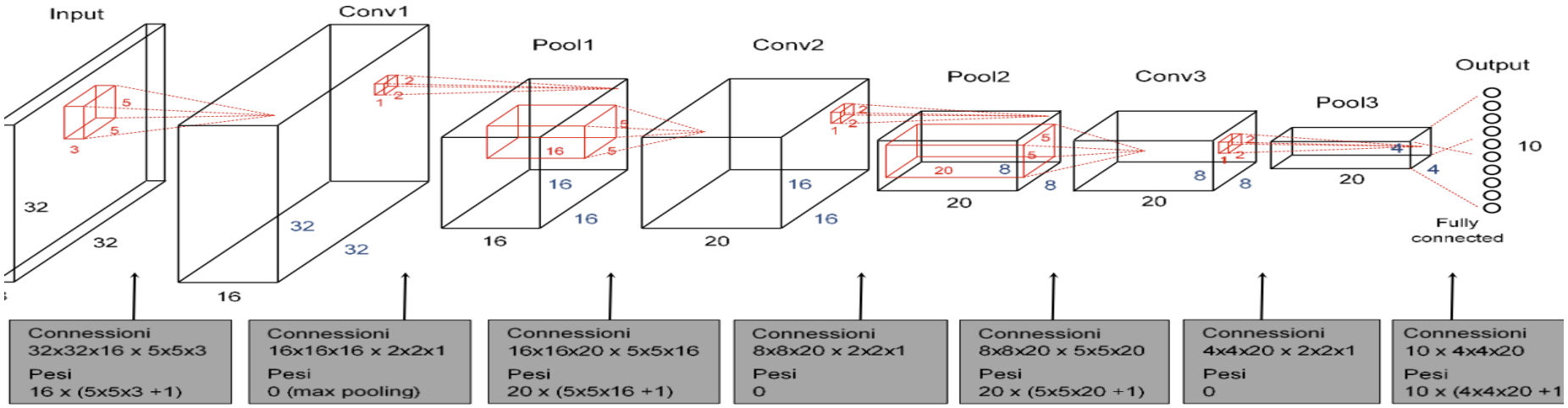
- Ciascuna «fetta» di neuroni (stessa depth) denota una feature map. Nell'esempio troviamo:
  - 3 feature map (dimensione 32x32) nel volume di input.
  - 6 feature map (dimensione 28x28) nel volume di output.
- I pesi sono condivisi a livello di feature map. I neuroni di una stessa feature map processano porzioni diverse del volume di input nello stesso modo. Ogni feature map può essere vista come il risultato di uno specifico filtraggio dell'input (filtro fisso).
- Nell'esempio il numero di connessioni tra i due livelli è (28x28x6) x (5x5x3) = 352800, ma il numero totale di pesi è 6 x (5x5x3 + 1) = 456. *In analoga porzione di MLP quanti pesi?*

CNN

Architettura

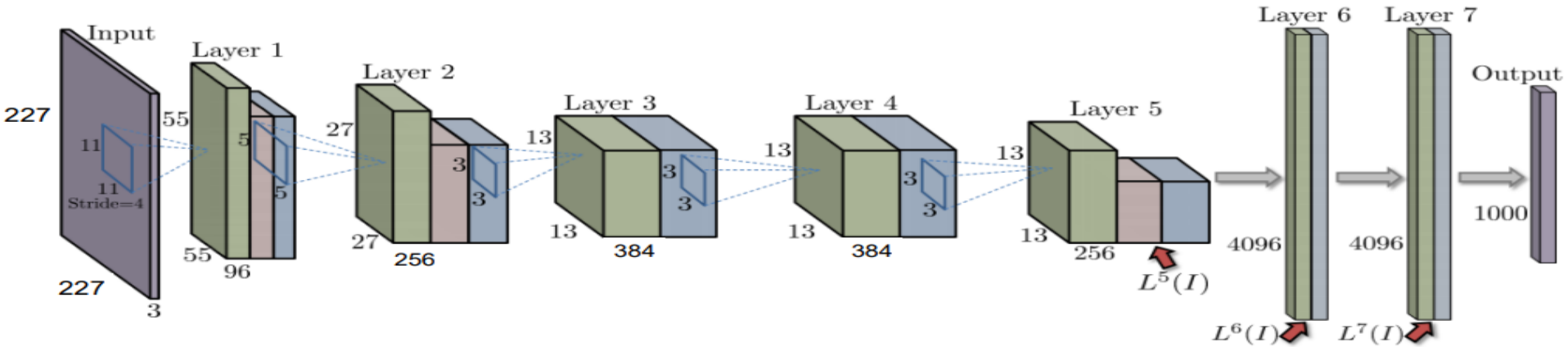
- Input: Immagini RGB 32x32x3;
- Conv1: Filtri:5x5, FeatureMaps:16, stride:1, pad:2, attivazione: Relu
- Pool1: Tipo: Max, Filtri 2x2, stride:2
- Conv2: Filtri:5x5, FeatureMaps:20, stride:1, pad:2, attivazione: Relu
- Pool2: Tipo: Max, Filtri 2x2, stride:2
- Conv3: Filtri:5x5, FeatureMaps:20, stride:1, pad:2, attivazione: Relu
- Pool3: Tipo: Max, Filtri 2x2, stride:2
- Output: Softmax, NumClassi: 10

Disegniamo la rete e calcoliamo neuroni sui livelli, connessioni e pesi



Neuroni totali: 31.562 (incluso livello input)
Connessioni totali: 3.942.784
Pesi totali: 22.466 (inclusi bias)

CNN

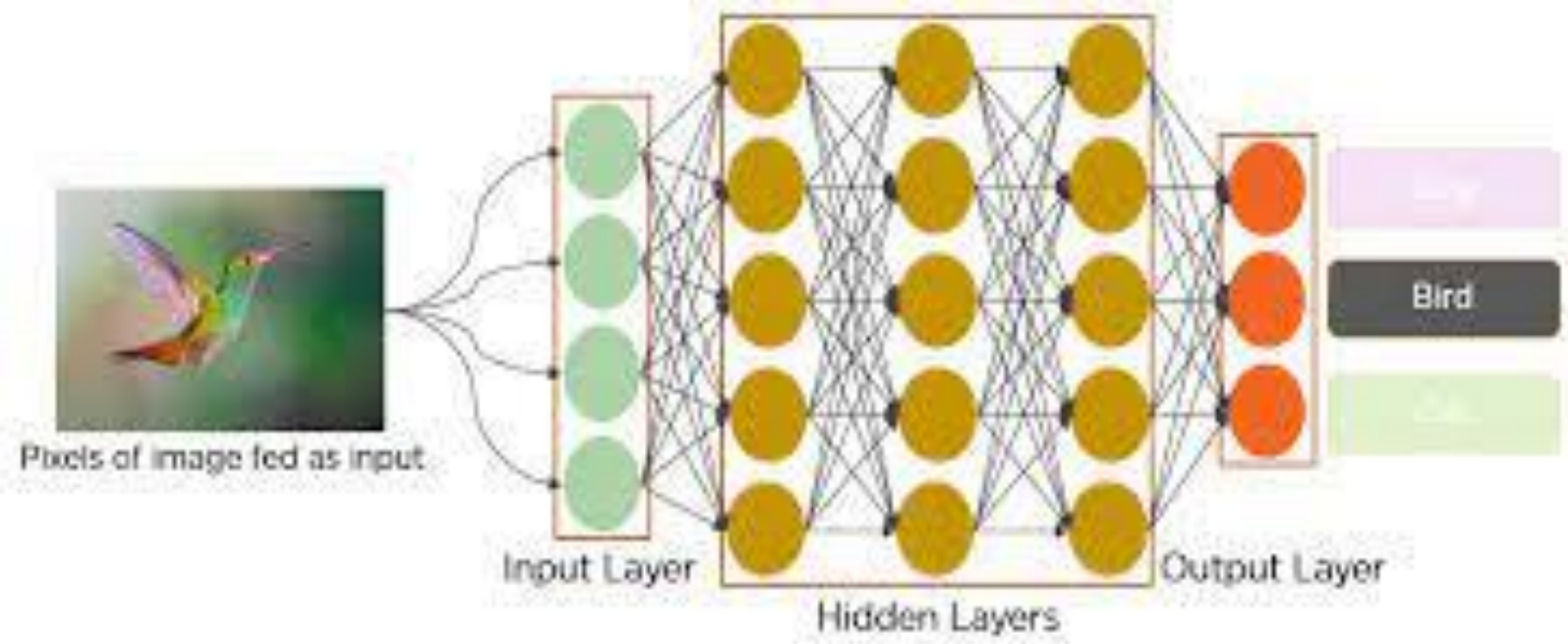


Codici colore

- Viola: Input (immagini 227x227x3) e Output (1000 classi di ImageNet)
- Verde: Convoluzione
- Rosa: Pooling (max)
- Blu: Relu activation

- Numero totale di parametri: 60M circa

CONVOLUTIONAL NEURAL NETWORK



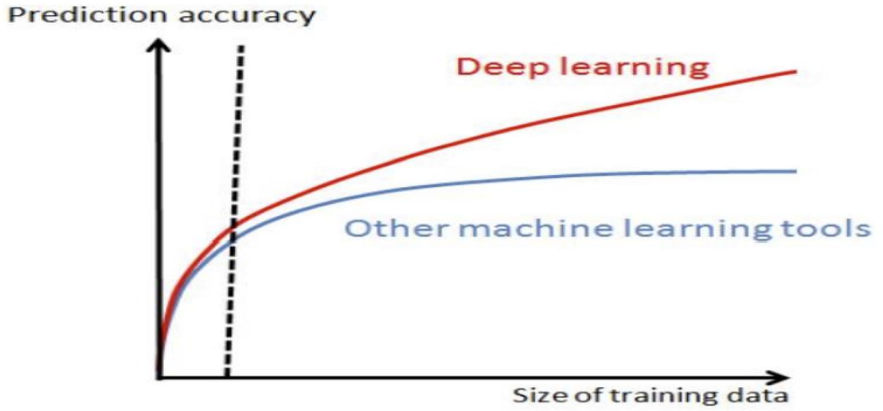
CONVOLUTIONAL NEURAL NETWORK

Ingredienti necessari

CNN ottengono già nel 1998 buone prestazioni in problemi di piccole dimensioni (es. riconoscimento caratteri, riconoscimento oggetti a bassa risoluzione), ma bisogna attendere il 2012 (AlexNet) per un radicale cambio di passo. AlexNet non introduce rilevanti innovazioni rispetto alle CNN di LeCun del 1998, ma alcune condizione al contorno sono nel frattempo cambiate:

- BigData: disponibilità di dataset etichettati di grandi dimensioni (es. ImageNet: milioni di immagini, decine di migliaia di classi).

La superiorità delle tecniche di deep learning rispetto ad altri approcci si manifesta quando sono disponibili grandi quantità di dati di training.



CONVOLUTIONAL NEURAL NETWORK

- GPU computing: il training di modelli complessi (profondi e con molti pesi e connessioni) richiede elevate potenze computazionali. La disponibilità di GPU con migliaia di core e GB di memoria interna ha consentito di ridurre drasticamente i tempi di training: da mesi a giorni.

Hardware per il training

- Il training di modelli complessi (profondi e con molti pesi e connessioni) su dataset di grandi dimensioni (es. ImageNet) richiede elevate potenze computazionali.
- La disponibilità di GPU con migliaia di core e GB di memoria interna è di fatto necessaria per contenere i tempi di training: da mesi a giorni/ore.

Nvidia Titan RTX (2.7K€)

- 4608 Core
- 24 GB RAM
- 16.3 TFLOPS (fp32)

CPU normalmente < 1 TFLOPS

CONVOLUTIONAL NEURAL NETWORK

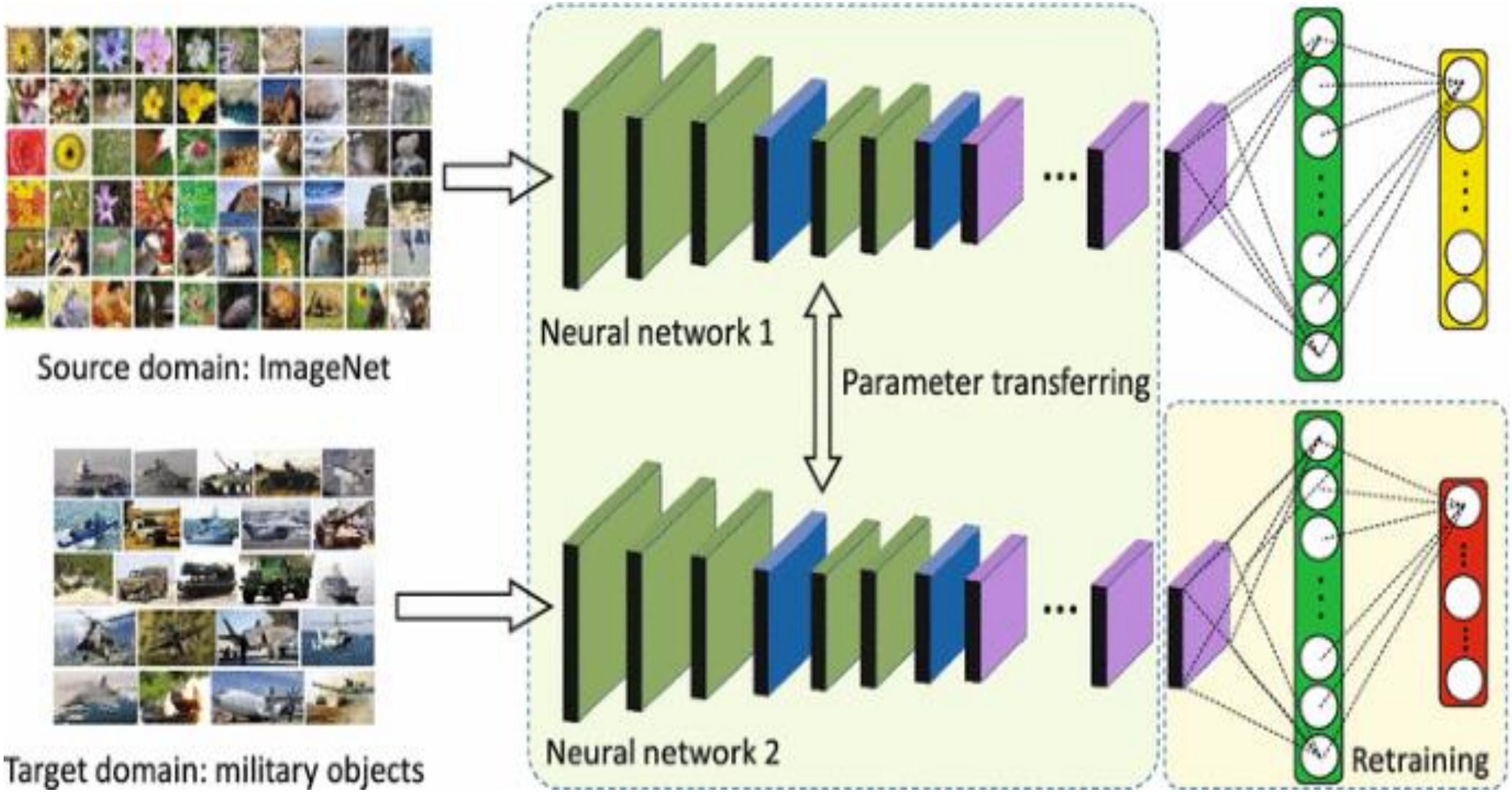
- L'addestramento può essere parallelizzato suddividendo il carico tra diverse GPU (i principali framework lo supportano in modo trasparente):
  - Workstation (es: NVIDIA DXG Station) con 4 GPU collegate tra loro (Nvlink) per massimizzare il trasferimento dei dati senza passare dal bus PCI-express.
  - GPU Cloud (es. Amazon, Google).
  - GPU-Cluster (es. DAVIDE del Cineca) con 180 GPU e una peak performance di circa 1 PFLOPS (1000 TFLOPS)

TOOL PER DEEP LEARNING

I principali framework per il deep learning (tutti con interfaccia Python) sono:

- Tensorflow* (Google). *Il più noto e diffuso*
- PyTorch (Facebook). *Molto apprezzato in ambito ricerca*
- Caffe (Berkeley). *Efficiente per Visione Artificiale*

APPLICAZIONI



APPLICAZIONI

APPLICAZIONI