SORINT.TEK

Come funzionano le

Reti Neurali Convoluzionali
Convolutional Neural Network

Gianluca Gerard, PhD
CTO @ SORINT.tek

https://ai-4-all.org/resources/

Open Learning Curriculum

- Il materiale presentato in questa lezione, nella sua versione originale in inglese, è liberamente disponibile dal sito https://ai-4-all.org/resources/
  - La presentazione originale contiene diverse attività e contenuti multimediali ad integrazione delle slide

- AI4ALL è un'organizzazione no-profit con sede negli Stati Uniti impegnata ad aumentare la diversità e l'inclusione nell'istruzione, nella ricerca, nello sviluppo e nelle politiche sull'intelligenza artificiale.

Obiettivi

Domande Fondamentali:

- Perché le Convolutional Neural Network (CNN) sono utili?
- Come funzionano?

Sarete capaci di:

- Spiegare come i computer vedono le immagini.
- Descrivere le parti e gli usi di una rete neurale convoluzionale.

Riconoscimento degli oggetti

- La maggior parte degli esseri umani può dire che tutte le immagini sottostanti sono immagini di cani nonostante le diverse angolazioni da cui sono state scattate le immagini e nonostante ogni cane appartenga a una razza diversa.
- Questo è difficile da eseguire per un computer utilizzando i metodi tradizionali.

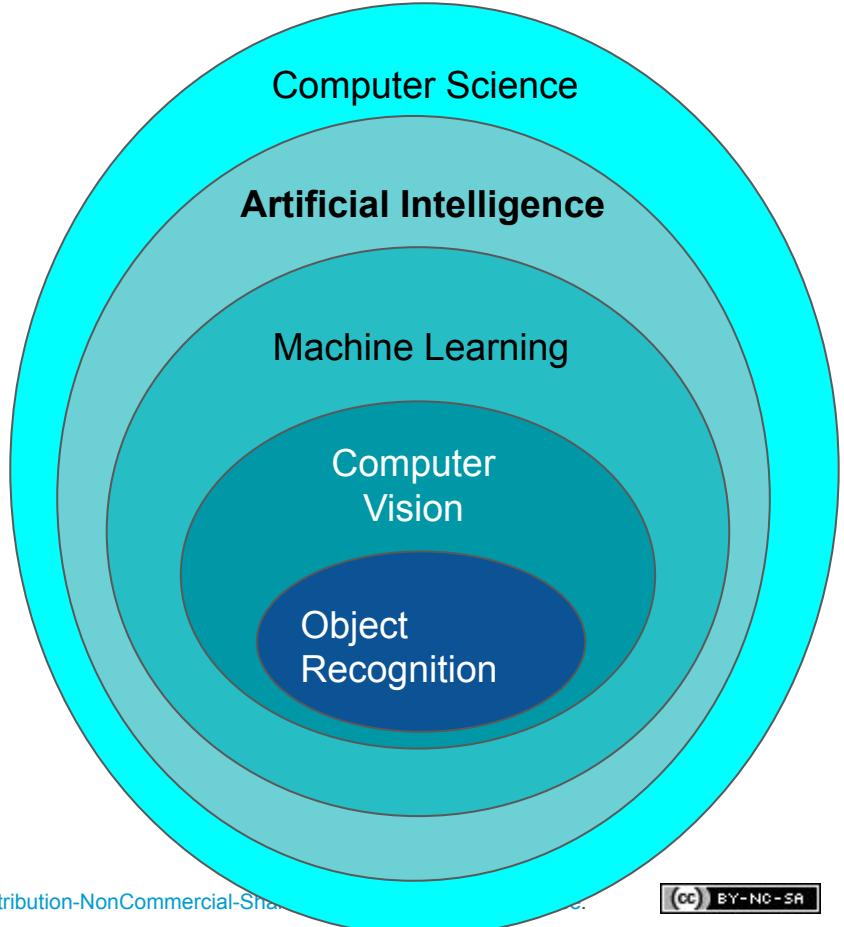
Autore: leisergu

Autore: Sp..andreea

Autore: Leo_65

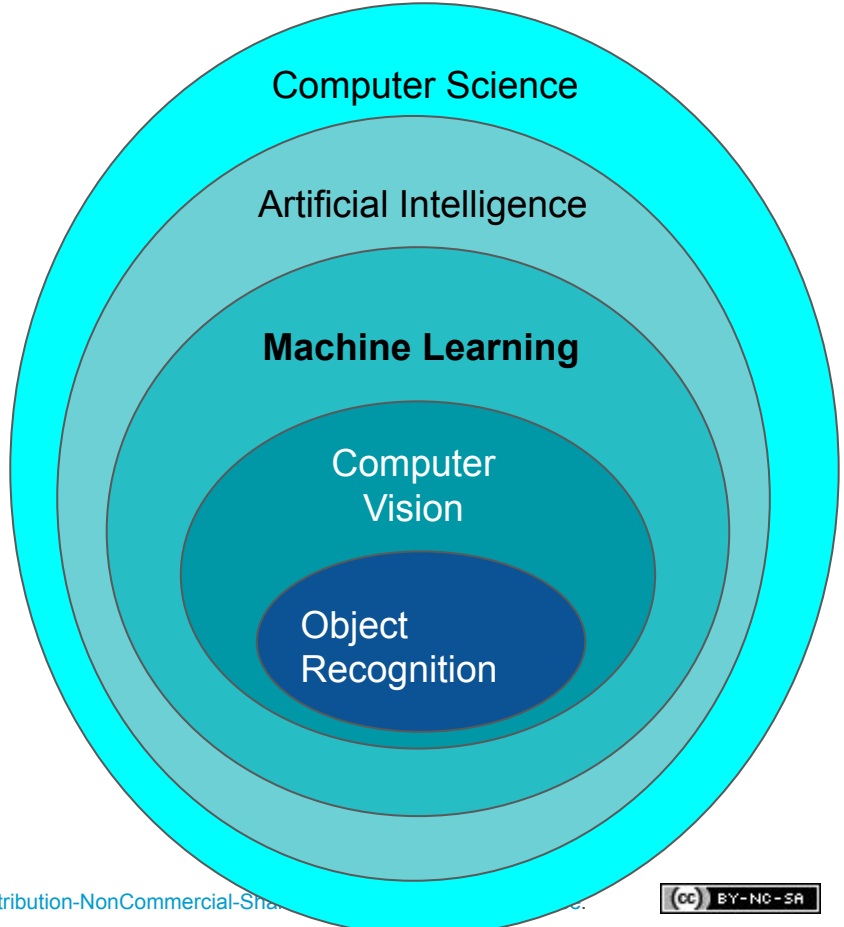
Intelligenza Artificiale

- L'**intelligenza artificiale** (***Artificial Intelligence* - AI**) è una branca dell'informatica che si occupa della progettazione di computer in grado di effettuare previsioni e decisioni.



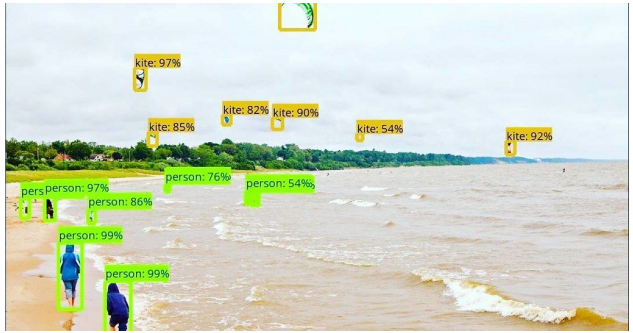
Machine Learning

- Il **Machine Learning** è una branca dell'Intelligenza Artificiale.
- Utilizzando metodi tradizionali, un programmatore dovrebbe tentare di descrivere come potrebbe apparire ogni immagine di un cane nel codice.
- Con l'apprendimento automatico il programmatore fornisce al programma molti esempi e il programma apprende da solo quale tipo di dati immagine appartiene all'immagine di un cane.

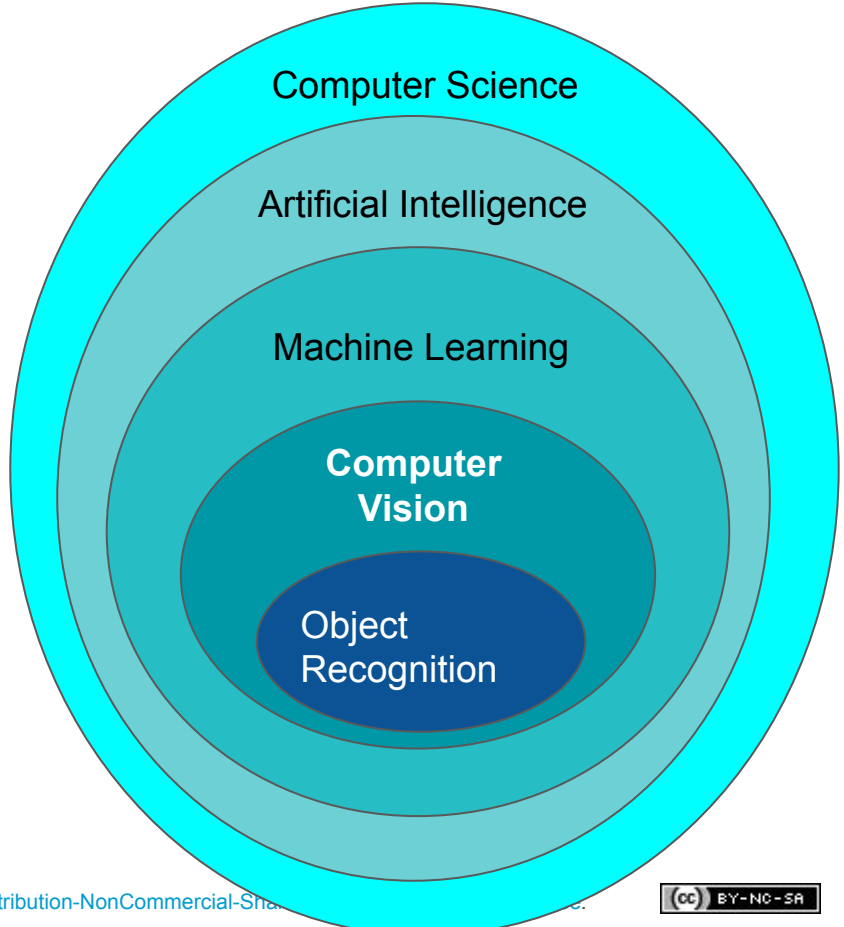


Terminologia

- Il rilevamento degli oggetti è una parte della **visione artificiale**.
- **Visione artificiale**: una branca dell'intelligenza artificiale che studia e sviluppa programmi per comprendere informazioni visive come immagini o video.
- Utilizza spesso il machine learning.



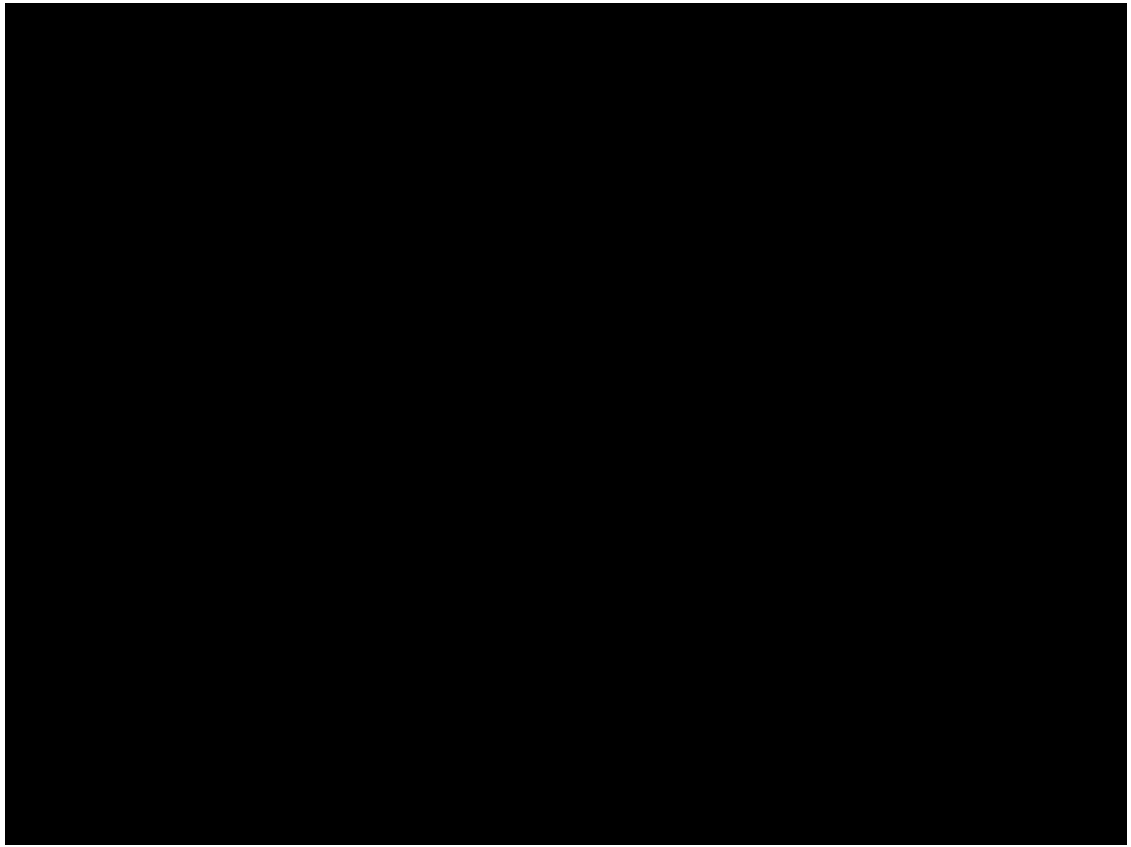
Fonte: Google



Usi della Visione Artificiale

- Riconoscimento facciale
  - Filtri su Instagram, videosorveglianza
- Stima della posa
  - Strumenti per gli sportivi, video-giochi
- Identificazione degli animali selvatici
- Diagnosi delle immagini mediche
  - Identificazione dei tumori
- Auto a guida autonoma

Openai Sora

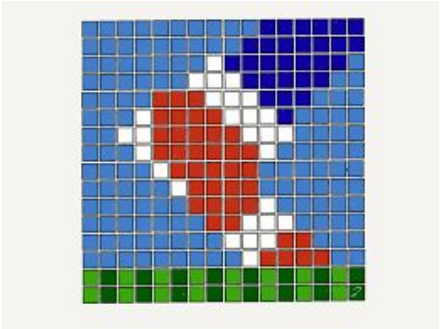


Prompt: The camera directly faces colorful buildings in burano italy. An adorable dalmation looks through a window on a building on the ground floor. Many people are walking and cycling along the canal streets in front of the buildings.

Computer e visione

Pixel

- **Pixel:** L'unità più piccola e indivisibile che costituisce un'immagine, come un minuscolo quadrato.

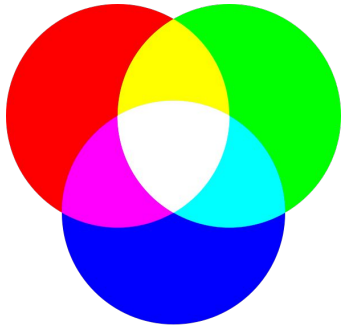


Ogni quadrato rappresenta un pixel
Autore: Raul Carrillo Garrido

RGB

- I numeri su ciascuno dei pixel sono stati suddivisi in valori R, G e B.
- **R** sta per rosso, **G** sta per green/verde e **B** sta per blu
- Ogni colore mostrato sullo schermo del computer è un mix di valori rosso, verde e blu.
- I valori saranno compresi tra 0 e 255, con i numeri più alti che saranno più di quel colore.

R 145 G 149 B 152	R 148 G 148 B 147	R 147 G 150 B 157
R 59 G 61 B 68	R 24 G 20 B 20	R 30 G 37 B 40
R 36 G 47 B 64	R 60 G 73 B 90	R 53 G 54 B 63

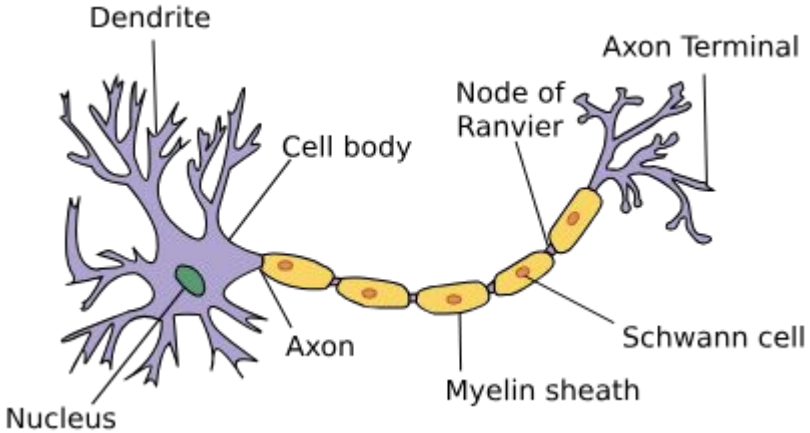


Reti Neurali

Neurone

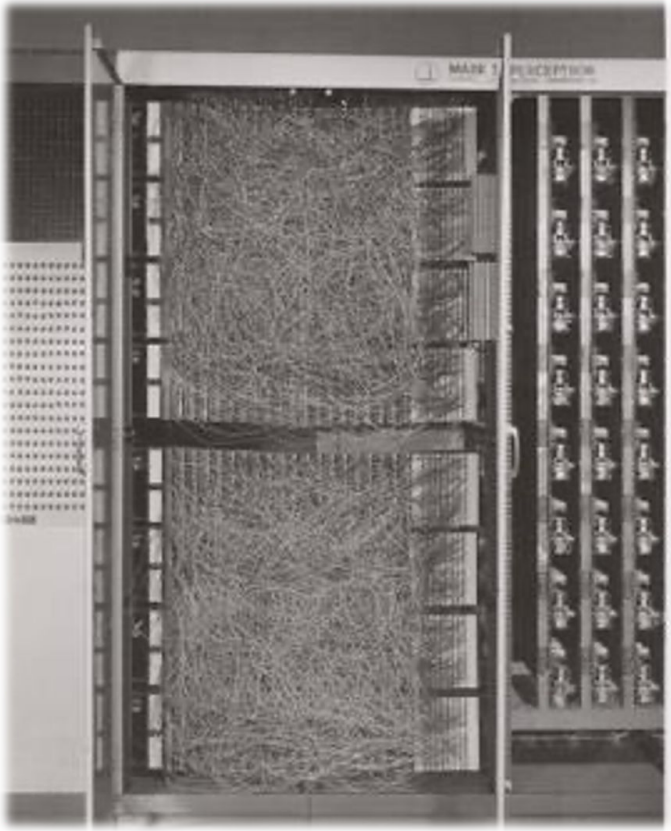
- **Un'unità computazionale**
  - riceve degli input *(x1, x2, …, xN)*
  - applica una ***trasformazione lineare***, *x1*w1 + x2*w2 + … xN*wN*
  - seguita da una ***funzione di attivazione*** **non lineare**, *f*
  - e produce un output
    - *o = f(x1*w1+x2*w2+...+xN*wN)*

- Questo processo simula l'elaborazione di un segnale da parte di un neurone biologico.

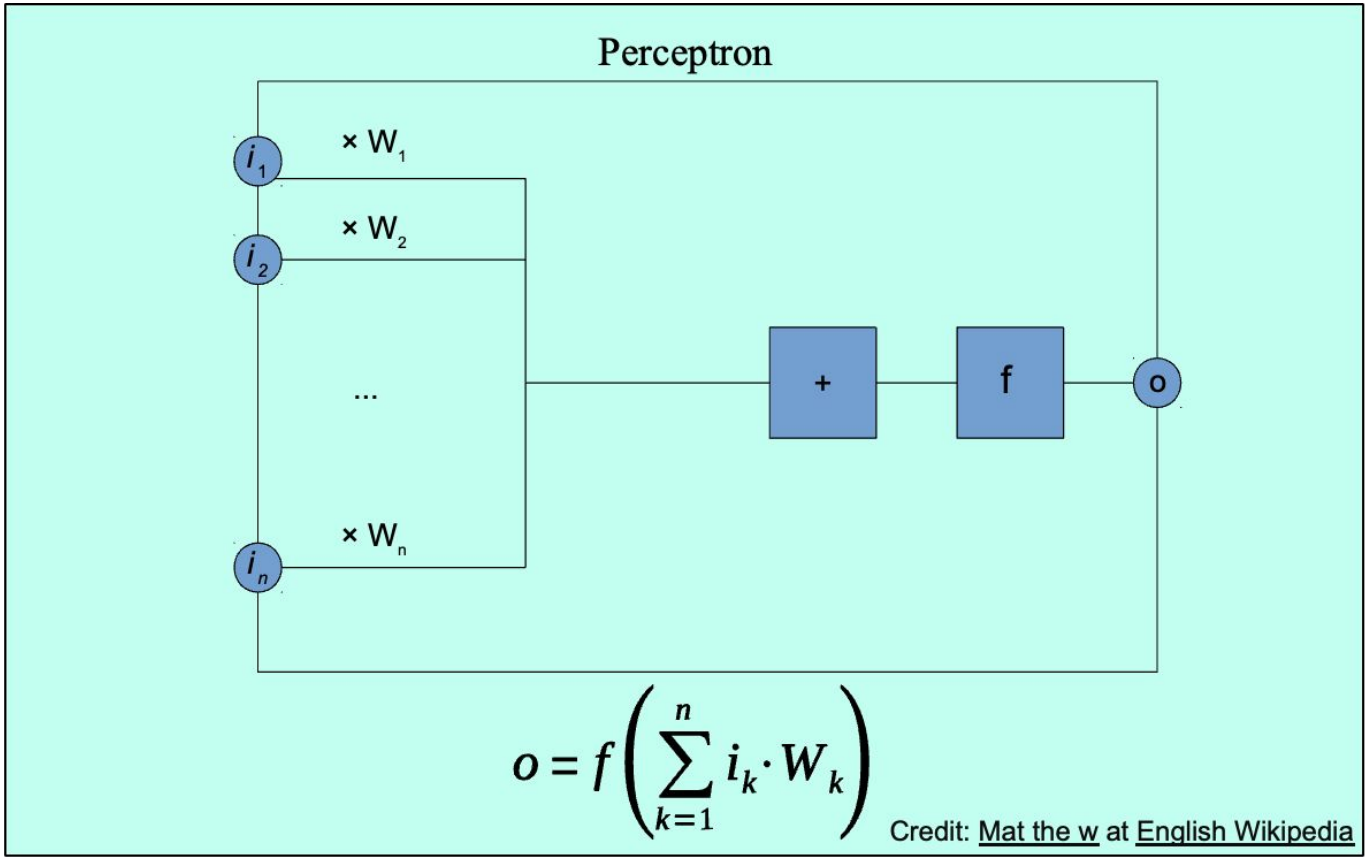


"Anatomy and Physiology" - US National Cancer Institute's Surveillance, Epidemiology and End Results (SEER) Program

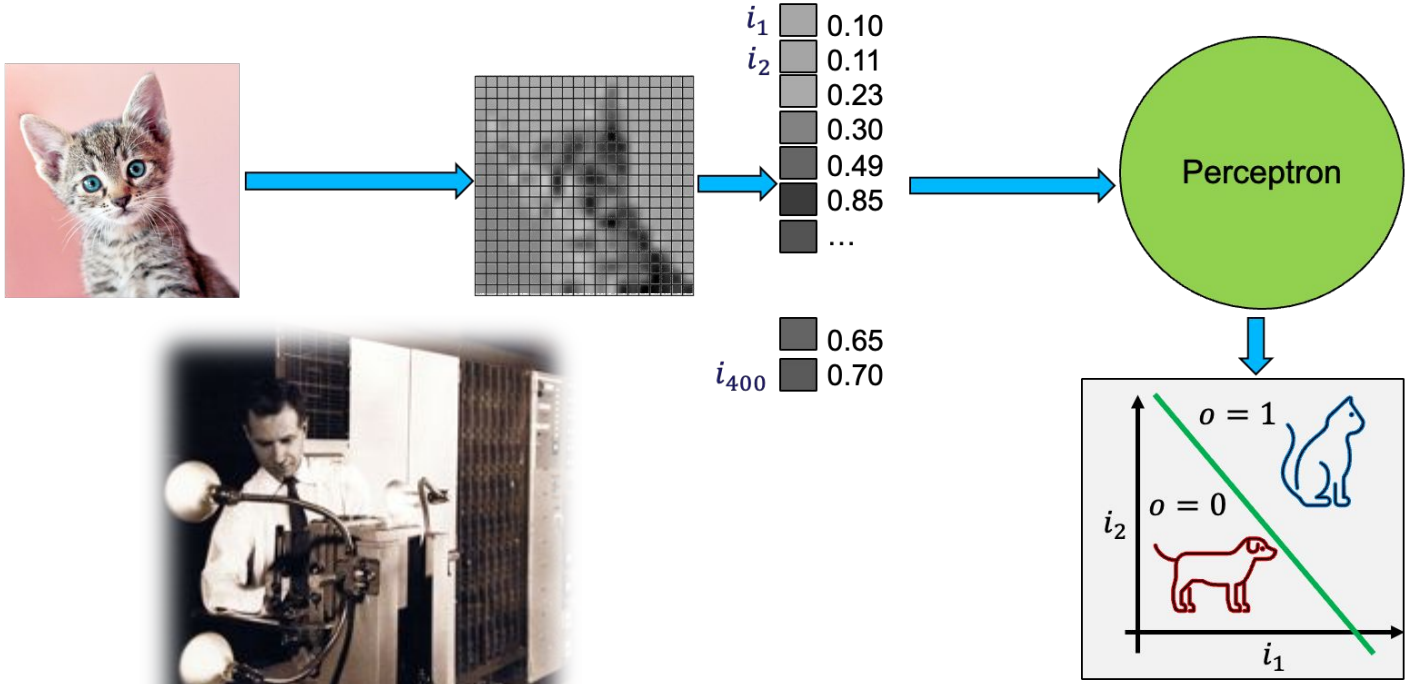
1958 - Il perceptron



1958 - L'architettura



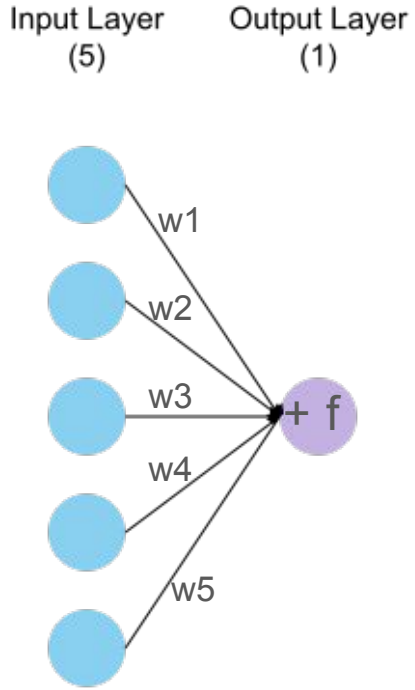
Frank Rosenblatt



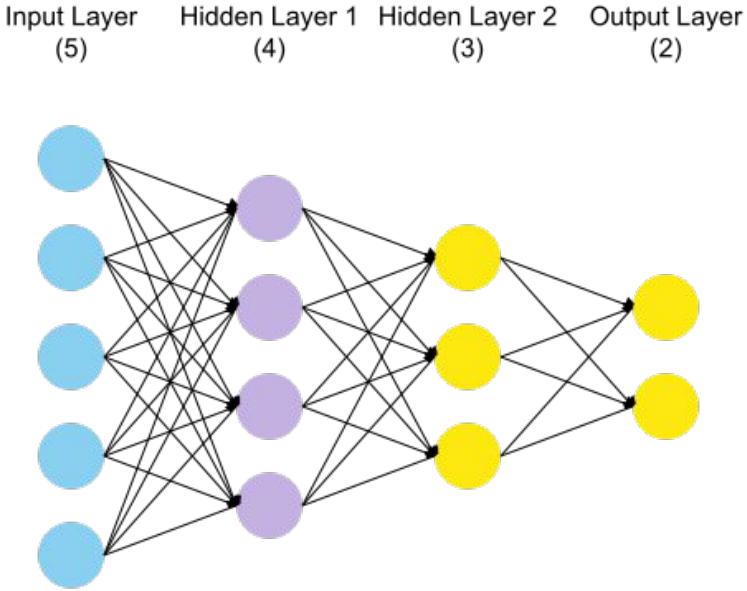
Strati di ingresso / uscita

- **Strato di ingresso (Input Layer):** La prima interfaccia della rete neurale che riceve le informazioni iniziali per il processo di apprendimento.
  - Per le immagini, questo strato accetta come input i valori dei pixel; per il Natural Language Processing (NLP), accetta sequenze di testo codificate.
- **Strato di uscita (Output Layer):** L'ultimo strato della rete neurale che fornisce il risultato dell'elaborazione.
  - Il tipo di output varia a seconda del compito specifico: classificazione dell'immagine, generazione di immagini, predizione del testo successivo, ecc.

Rappresentazione moderna del percettrone



Multilayer Perceptron



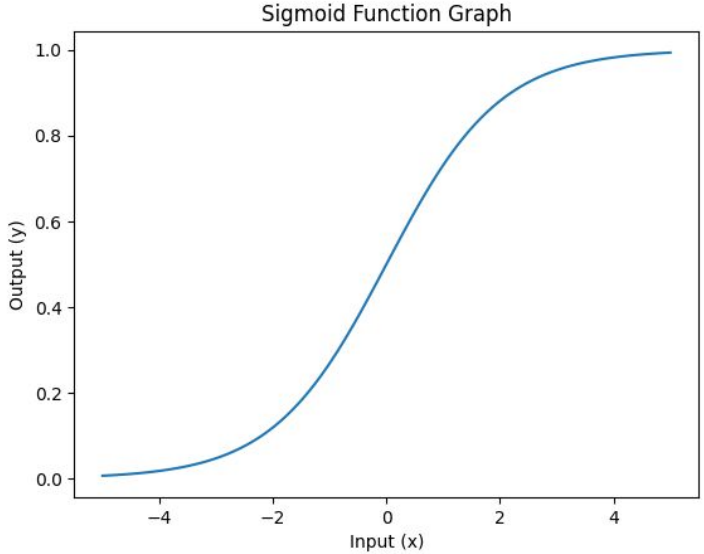
- **Strati nascosti (*hidden layers*)**: Gli strati intermedi tra l'input e l'output. Ogni strato trasforma gli output del precedente in nuovi input per il successivo, estraendo rappresentazioni sempre più astratte dei dati iniziali.

Rete neurale

- Una **struttura** computazionale **composta da *neuroni*** artificiali **organizzati in *strati***, progettata per **apprende**re **rappresentazioni dei dati attraverso l'addestramento**.

Funzione di attivazione

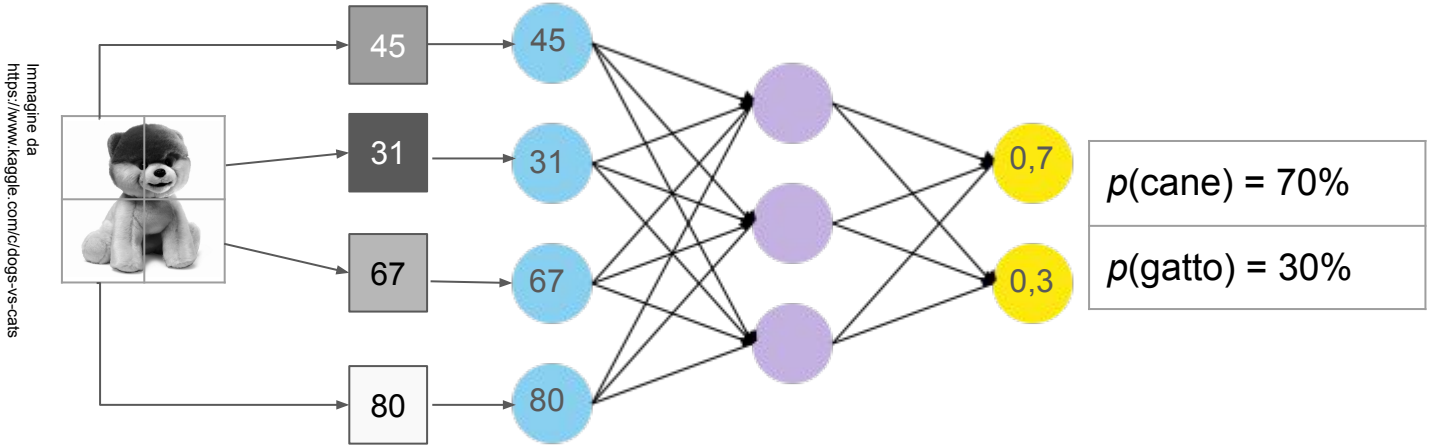
- La **funzione sigmoide** è un esempio di funzione di attivazione



Non è lineare perché non esiste una combinazione di numeri fatta da semplici somme e moltiplicazioni che ricostruisca questa funzione

Funzione sigmoide in uscita

- La **funzione sigmoide** nello strato di output, fornendo valori sempre compresi tra 0 e 1, permette di leggere il valore di ogni neurone di questo strato come la probabilità di classificazione dell'input.



Reti Neurali Convoluzionali - Introduzione

Limiti dei Multi Layer Perceptron

- Ogni strato è collegato a ogni altro strato.
  - Questo occupa molto spazio.
    Più un'immagine è grande, più connessioni ci sono nella rete.
  - Non fornisce il contesto per dove si trova qualcosa in un'immagine.
- **Le reti neurali convoluzionali esaminano sezioni di un'immagine alla volta, risolvendo questi problemi.**

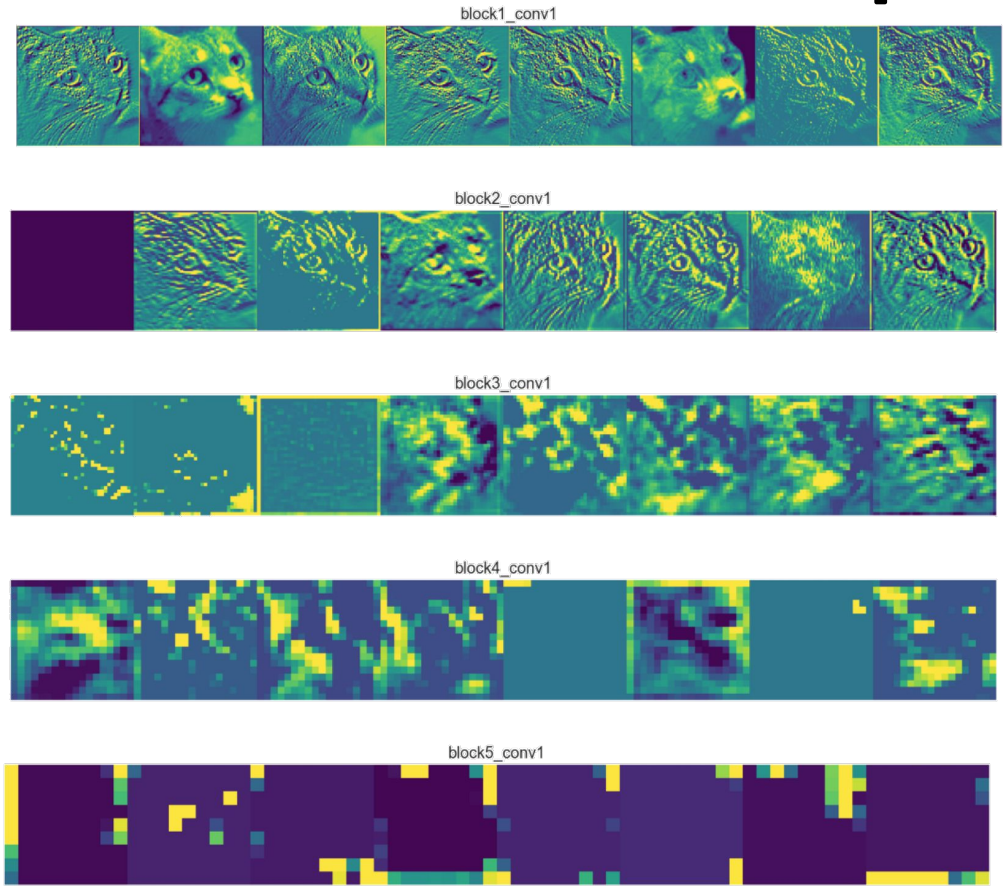
Reti Neurali Convoluzionali

- Una rete neurale convoluzionale trova **motivi** (***pattern***) in un'immagine (come orecchie a punta, occhi o baffi) chiamati **caratteristiche** (***feature***).
- In una rete neurale convoluzionale esistono **due tipi speciali di strati nascosti:**
  - **Strato convoluzionale:** trova modelli nelle immagini applicando **filtri** o ***kernel*.** Il processo per fare ciò è chiamato **convoluzione.**
    - Ingresso: un'immagine (in pixel RGB)
    - Uscita: una versione trasformata dell'immagine, che mostra parti specifiche dell'immagine
  - **Strato di aggregazione** (***pooling***)**:** semplifica le immagini

Che cosa fanno i livelli convoluzionali?

- I livelli convoluzionali delle CNN rilevano, usando i filtri, dei *pattern* in un'immagine
- I *pattern* includono inizialmente strutture semplici, come per esempio i bordi, e successivamente rappresentazioni sempre più astratte

Esempio



Strato 1

Strato 2

Strato 3

Strato 4

Strato 5

Reti Neurali Convoluzionali - Livelli convoluzionali

Definizioni

- I computer trovano i *pattern* in un'immagine applicando **filtri / kernel** nella forma di matrici
  - **Che cos'è una matrice?**
    Un gruppo di numeri organizzati in righe e colonne.
  - Si possono fare delle operazioni numeriche parallele sulle matrici (come la somma, la moltiplicazione, la divisione, ecc.)

-4	-3	-2
-1	0	1
2	3	4

$\odot$

-1	0	1
-2	0	2
-1	0	1

$=$

4	0	-2
2	0	2
-2	0	4

Demo

- https://setosa.io/ev/image-kernels/

Here's a playground were you can select different kernel matrices and see how they effect the original image or build your own kernel. You can also upload your own image or use live video if your browser supports it.

Choose File No file chosen Live video

-1	-1	-1
-1	8	-1
-1	-1	-1

outline

Che cos’è un filtro?

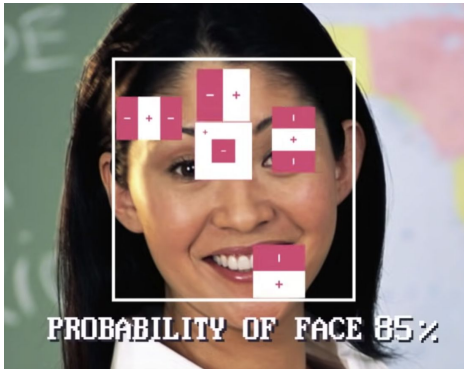
- Un **filtro** è una matrice di numeri
- La matrice decide cosa aggiungere o sottrarre ai pixel, facendo emergere diverse parti dell'immagine.
- Ciò aiuta a rilevare i bordi (orizzontali, verticali, obliqui).
- Una rete neurale convoluzionale applica i filtri molte volte e a livelli diversi.

Che funzione assolvono i filtri?

- Ogni filtro contiene numeri che ***estraggono tipi specifici di pattern***

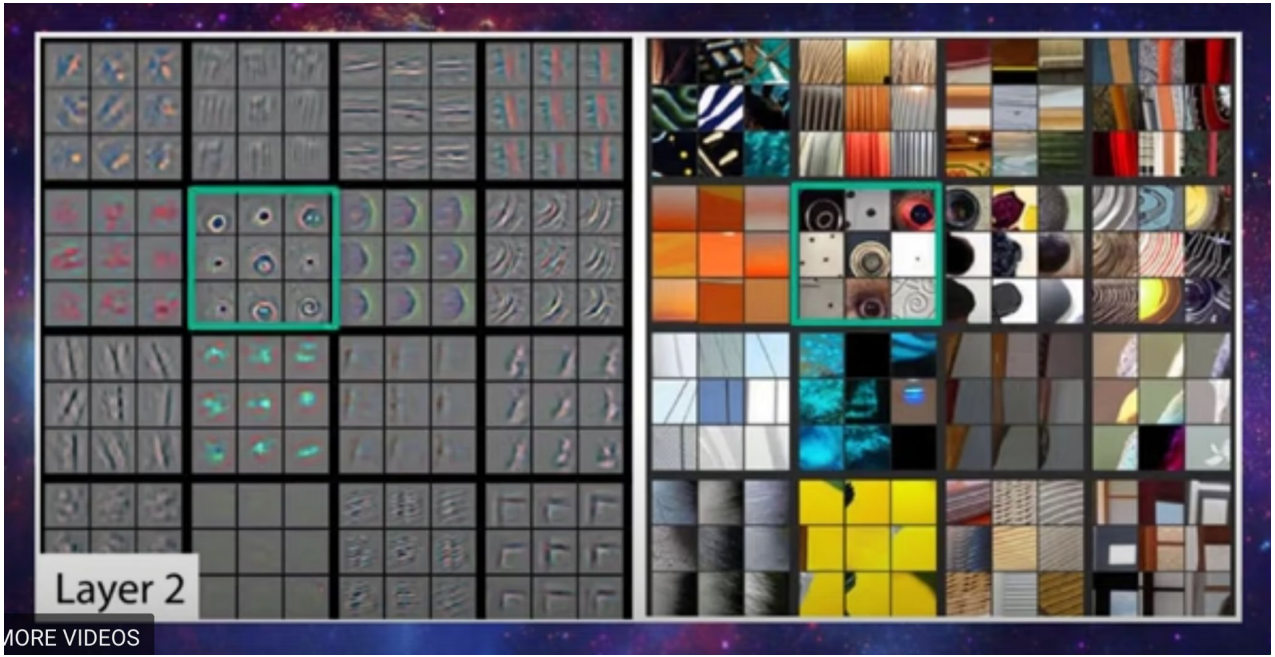
Come aiutano a riconoscere?

- I filtri possono trovare i bordi, ma i filtri più grandi possono trovare caratteristiche più complesse, come linee e cerchi
- La presenza di forme tipiche in una faccia (es. i cerchi degli occhi, le linee di bocca e naso) aumentano la probabilità che si tratti di un'immagine di un volto



Filtri a strati profondi

- Man mano che gli strati convoluzionali diventano più profondi, si rilevano caratteristiche più complesse.



Scelta dei filtri

- Come fa una rete neurale convoluzionale a scegliere quali filtri sono utili per identificare un'immagine?

- *Le reti neurali convolutonali apprendono i filtri che aiutano provando molti filtri diversi e valutandoli in modo diverso.*

La fase di addestramento

Come imparano le reti neurali?

- All’inizio della presentazione abbiamo detto che la Visione Artificiale usa spesso il “Machine Learning”, cioè l’Apprendimento Automatico.
- Approfondiamo ora come avviene l’“Apprendimento”, in particolare come apprendono le reti neurali.

Flashback - i filtri

- Le reti convoluzionali applicano dei filtri in sequenza.
- Ogni filtro è una matrice di numeri, per esempio il filtro per identificare i bordi verticali è il seguente (a destra un esempio di applicazione)

+1	0	-1
+2	0	-2
+1	0	-1

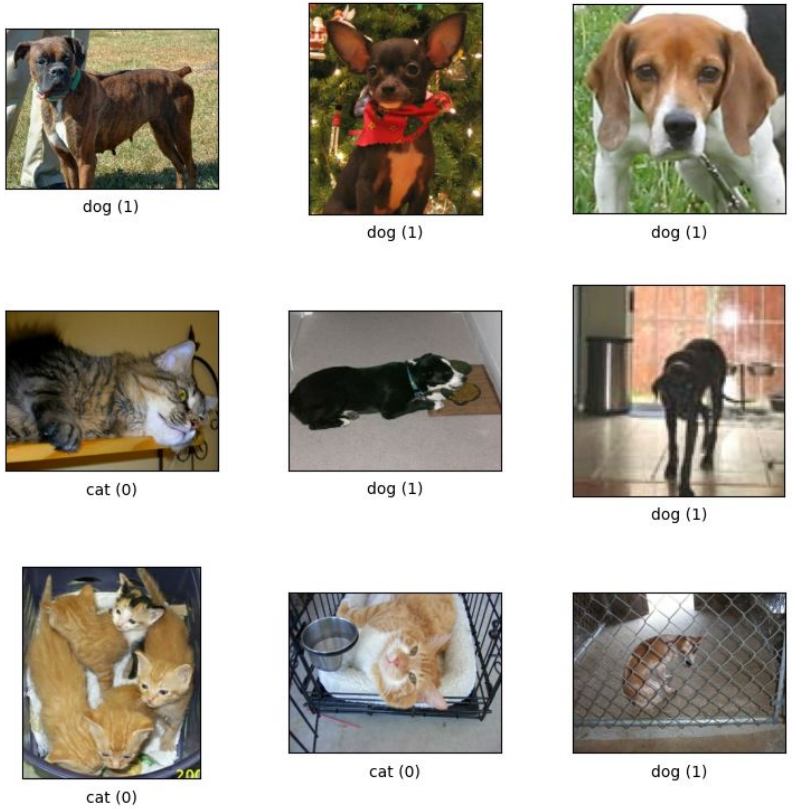
Apprendere i filtri

- I filtri nelle reti neurali non sono pre-impostati da chi sviluppa la rete neurale ma vengono appresi a partire da dati di esempio.

Apprendimento Automatico = Apprendere dagli Esempi

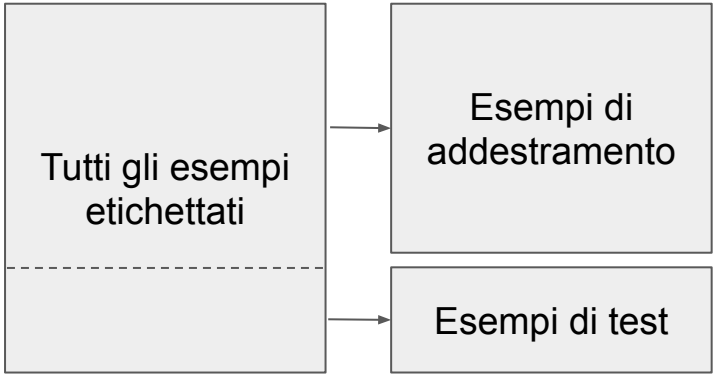
Come si impara dagli esempi?

- **Fase 1**:
  raccolta e annotazione degli esempi
- A destra esempi dal *dataset* “Cats vs. Dogs”
- Ogni immagine di cane o Gatto è *annotata* con la sua *etichetta*



Come si impara dagli esempi?

- **Fase 2**:
  Si divide il dataset in due Gruppi
  - Il dataset di addestramento
  - Il dataset di test



- Il **dataset di addestramento** è usato per l'apprendimento
- Il **dataset di test** è usato al termine dell'apprendimento per valutare le prestazioni del modello nel distinguere i cani dai gatti

Come si impara dagli esempi?

- **Fase 3.a**:
  Gli esempi di addestramento sono mostrati iterativamente alla CNN
- **Fase 3.b**:
  La CNN classifica l'immagine con una probabilità associata a ciascuna classe

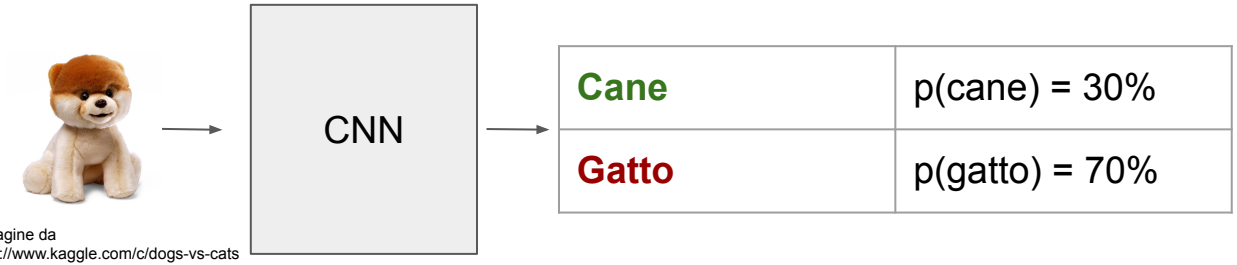
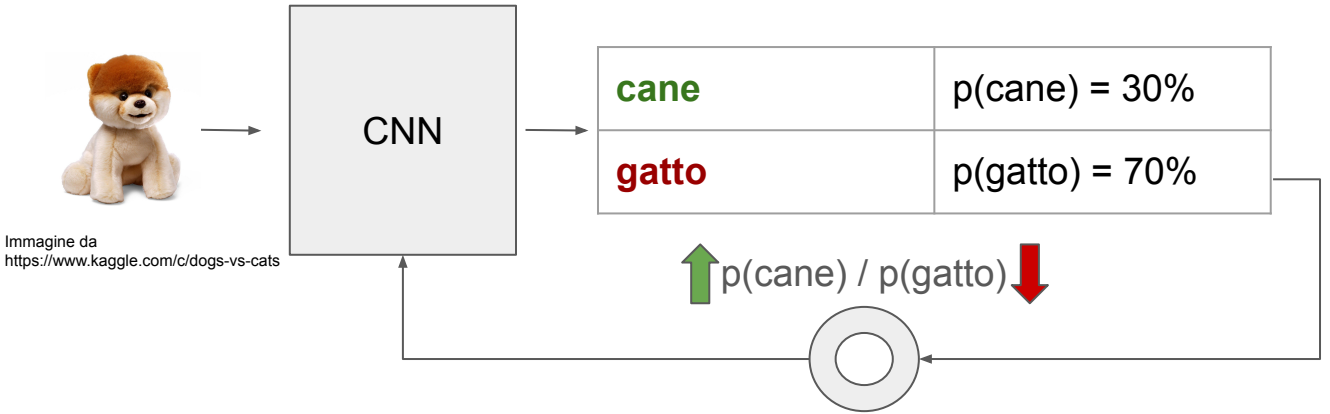


Immagine da
https://www.kaggle.com/c/dogs-vs-cats

Come si impara dagli esempi?

- **Fase 4**:
  I filtri vengono modificati per aumentare la probabilità dell'etichetta attesa e per diminuire la probabilità dell'etichetta errata
- Il processo si ripete fino a quando le probabilità non cambiano più in modo significativo



Calcolo delle prestazioni

- Al termine dell'addestramento occorre sempre calcolare le prestazioni del modello
- Il metodo più semplice è quello di costruire una **matrice di confusione**
- **Questa matrice va calcolata sugli esempi di test non visti durante l'addestramento**

		Classi previste	
		Cane	**Gatto**
Classi osservate	**Cane**	**TRUE DOG**	**FALSE CAT**
	Gatto	**FALSE DOG**	**TRUE CAT**

Calcolo delle prestazioni

- Dalla matrice di confusione si derivano le *metriche di prestazione*

		Classi previste	
		Positiva	**Negativa**
Classi osservate	**Positiva**	**TRUE POSITIVE (TP)**	**FALSE NEGATIVE (FN)**
	Negativa	**FALSE POSITIVE (FP)**	**TRUE NEGATIVE (TN)**

Accuratezza = TP + TN / TP + TN + FP + FN

Precisione = TP / TP + FP

Recall o sensitività = TP / TP + FN

- **Un buon modello ha alta precisione e sensitività**

I filtri: valori iniziali

- I filtri di una rete neurale sono matrici. Per esempio un filtro 3x3 conterrà 9 valori, che possiamo chiamare genericamente w11, w12, … w21, … w33

w11	w12	w13
w21	w22	w23
w31	w32	w33

- **Prima dell'addestramento i valori in queste matrici sono casuali**
  - Seguono una distribuzione normale centrata a 0

I filtri e l'addestramento

- Man mano che la rete apprende i valori dei filtri cambiano.
- L'obiettivo dell'addestramento è duplice:
  - massimizzare la probabilità assegnata dalla rete alle classi predette corrette
  - minimizzare la probabilità assegnata dalla rete alle classi predette sbagliate
- Per raggiungere l'obiettivo serve definire una **metrica unica** che guidi le modifiche dei filtri nella giusta direzione**:**

la **funzione di costo**

La funzione di costo

La funzione di costo per una classificazione binaria calcola il **logaritmo negativo della probabilità assegnata alla classe corretta osservata**.

  - Se la classe vera è 'cane', la funzione considera il logaritmo negativo della probabilità che la rete assegna a 'cane'.
  - Se la classe vera è 'gatto', si utilizza il logaritmo negativo di 1 meno la probabilità assegnata a 'cane', che equivale alla probabilità di 'gatto'.

- Se alla classe vera è assegnata alta probabilità il costo è basso ($\log(1) = 0$)
- Se invece alla classe vera è assegnata bassa probabilità il costo è alto ($-\log(0) \rightarrow \infty$).

Questa funzione di costo si chiama ***Binary Cross Entropy,*** **BCE**

Funzione di costo: esempio

La CNN per ogni immagine assegna una probabilità all'una e all'altra classe

- Per ogni predizione della CNN si calcola il "costo"

Immagine da https://www.kaggle.com/c/dogs-vs-cats

CNN

Cane	p(cane) = 30%	**Costo (BCE)** = - log(0,3) **= 1,20**
Gatto	p(gatto) = 70%	

Copyright: mmsopen3

CNN

Cane	p(cane) = 20%	**Costo (BCE)** = - log(0,8) **= 0,22**
Gatto	p(gatto) = 80%	

Funzione di costo e filtri

Quindi

BCE (“cane”,*p*()) = -log (*p*()) = - log (CNN())

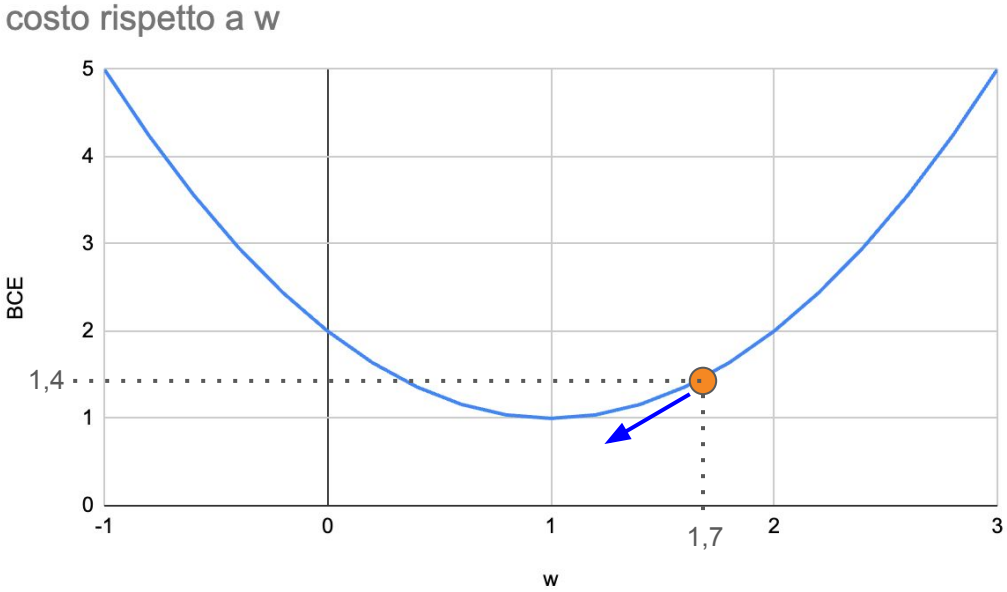
Il risultato della CNN dipende però non solo dall’immagine ma **anche dal valore dei filtri della CNN**

Possiamo perciò scrivere

BCE (“cane”,*p*())= - log (CNN( , {w11, w12, …, w21, …, w33, …}))

Funzione di costo e filtri

BCE (“cane”, *p*()) = - log (CNN( , {w11, w12, ..., w21, ..., w33, ...}))



BCE(..., {w=1,7}) = 1,4

Cambio w nella direzione della freccia affinché il costo tenda al minimo

L’algoritmo di *back propagation*

Quanto abbiamo visto è il metodo più comune di addestramento delle reti neurali: l’algoritmo di back-propagation

Con questo algoritmo i pesi vengono aggiustati con il calcolo del ***gradiente*** della funzione di costo rispetto al peso.

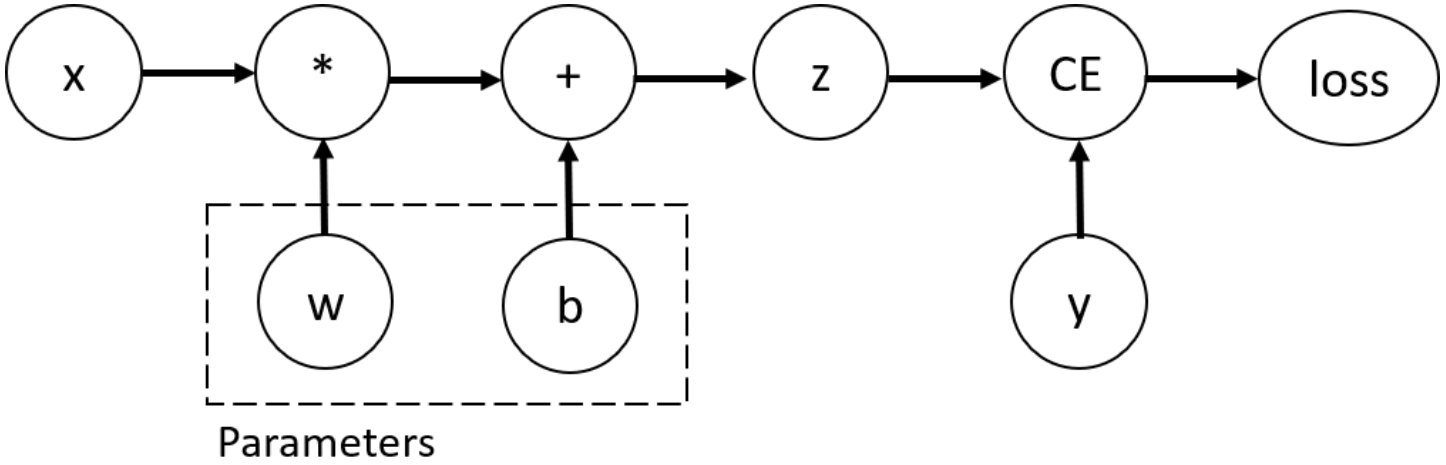
- Il gradiente è solo il nome elaborato per indicare la direzione e la quantità di aggiustamento

Questo valore viene calcolato automaticamente dai framework con cui si eseguono le reti neurali grazie al sistema di “differenziazione automatica”.

- Per esempio, in PyTorch, vi è il motore torch.autograd

Il grafo computazionale

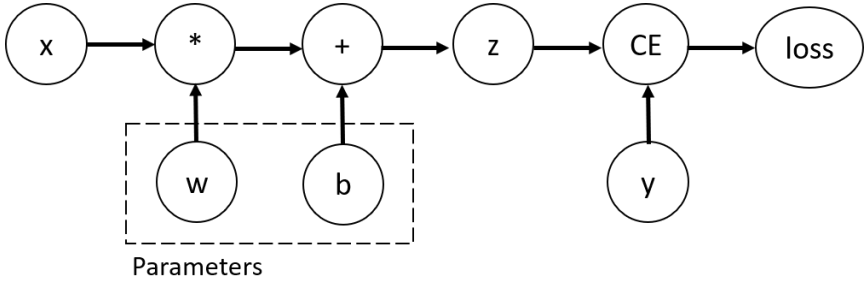
La differenziazione automatica si esegue con la compilazione del *grafo computazionale della rete neurale*. Per esempio il grafico



è la formula *CE(y, z) = CE(y, x*w+b)*, dove
CE è la Cross-Entropy, *x* l'input (l'immagine), *y* è l'etichetta associata all'input *x* (cano o gatto, 0 o 1), il calcolo *x*w+b* è il percettrone di cui *w* e *b* sono i suoi pesi (o parametri)

La funzione *backward*

- Creiamo il grafo in PyTorch




```
x = torch.ones(4)  # input
y = torch.zeros(2)  # output atteso
w = torch.randn(4, 2, requires_grad=True)
b = torch.randn(2, requires_grad=True)
z = torch.matmul(x, w)+b
loss = F.binary_cross_entropy_with_logits(z, y)
```

- Per ogni nodo del grafo (un'operazione) abbiamo una funzione associata che ci permette di calcolare il gradiente

```
print(f"Funzione di calcolo del gradiente per z z = {z.grad_fn}")
print(f"Funzione di calcolo del gradiente per la loss = {loss.grad_fn}")
```

Funzione di calcolo del gradiente per z = <AddBackward0 object at 0x7de5…f0>
Funzione di calcolo del gradiente per la loss = <BinaryCrossEntropyWithLogitsBackward0 object at 0x7de6…d0>

- Ora possiamo calcolare i gradienti dei parametri *w* e *b* e otteniamo:

```
loss.backward()
print(w.grad)
print(b.grad)
```

```
tensor([[0.0033, 0.0101],
        [0.0033, 0.0101],
        [0.0033, 0.0101],
        [0.0033, 0.0101]])
tensor([0.0033, 0.0101])
```

Reti Neurali Convoluzionali - Strati di pooling

Strati di pooling

- Anche con l'uso dei filtri rimangono molte informazioni da elaborare per il computer.
- Sarebbe utile prendere solo le parti importanti.
  - L'idea è che la CNN possa guardare sezioni più grandi dell'immagine in una volta sola.
- Per evidenziare solo le parti più importanti dell'immagine, le reti neurali convoluzionali eseguono una funzione chiamata ***pooling***.
- La forma più popolare di pooling nelle reti neurali convoluzionali è chiamata ***max pooling***.
  - Una sezione quadrata di pixel viene condensata in un pixel, quello con il valore più alto in quella sezione.

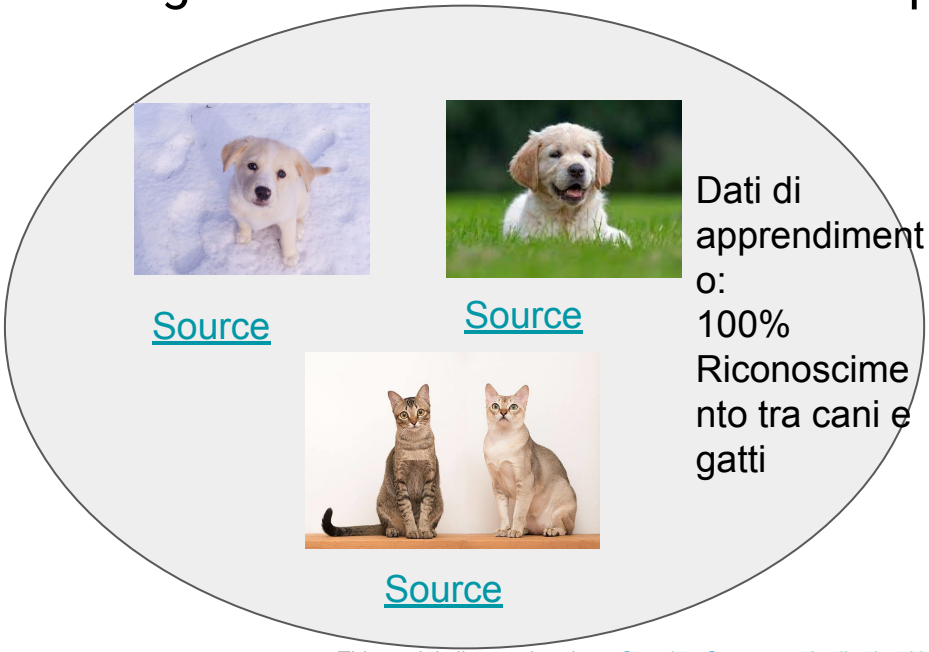
Overfitting

- Il pooling aiuta a impedire l'***overfitting*** dell'algoritmo

- **Che cos'è l'overfitting?**
  - Il sistema impara dai dati che le vengono forniti.
  - A volte **si adatta eccessivamente**:
    funzionerà per i dati forniti, ma non generalizzerà ai dati su cui non è stato addestrato in precedenza.
    - Esempio: se un sistema per i consigli suggerisce solo cose che hai già visto e apprezzato, potrebbe essere affetto dal problema di overfitting.

Overfitting

- Ad esempio, una rete neurale convoluzionale addestrata su 10 immagini di cani dovrebbe classificare correttamente quei 10 cani. Ma, se la rete overfitta, classificherà erroneamente qualsiasi immagine di un cane che non sia tra quelle 10 che ha già visto.



Non riconosciuto come cane perché non era nel set di dati originale

Source

Riconosciuto come cane; nel set originale

Source

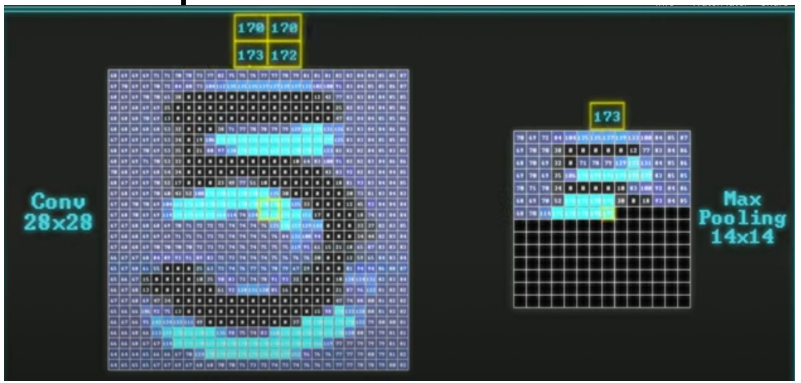
Overfitting

- Il raggruppamento aiuta a fermare l'adattamento eccessivo rendendo l'immagine stessa più generalizzabile.

- Inoltre, molte reti neurali convoluzionali rimuovono alcuni neuroni per evitare un adattamento eccessivo

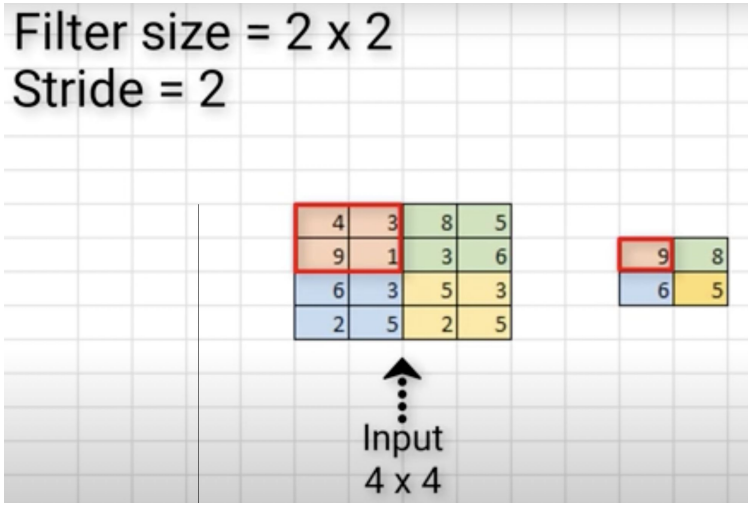
Cosa fanno i livelli di pooling?

- I livelli di pooling aiutano a ridurre l'overfitting
- I livelli di raggruppamento mantengono le parti più importanti di un'immagine e scartano il resto
- Ciò rende più veloce l'esecuzione dei calcoli sull'immagine
- Un kernel scorre l'immagine, sceglie il valore massimo e viene inserito nella nuova immagine.
- Le informazioni sull'immagine non sono perse.



Come funziona il max pooling

- Il max pooling sceglie una dimensione del filtro (per esempio 2x2) e forma una nuova matrice prendendo il valore massimo dalla sezione coperta dal filtro e applicandolo alla nuova matrice



MAX POOLING

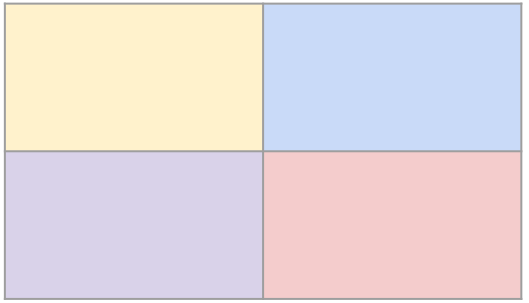
3	3	3	5	6	7
3	3	3	5	6	7
3	3	3	7	7	7
1	1	1	9	5	3
1	7	1	7	2	1
1	1	1	3	6	4

Filter Size: 3x3
Stride 3

MAX POOLING

3	3	3	5	6	7
3	3	3	5	6	7
3	3	3	7	7	7
1	1	1	9	5	3
1	7	1	7	2	1
1	1	1	3	6	4

Filter Size: 3x3
Stride 3



MAX POOLING

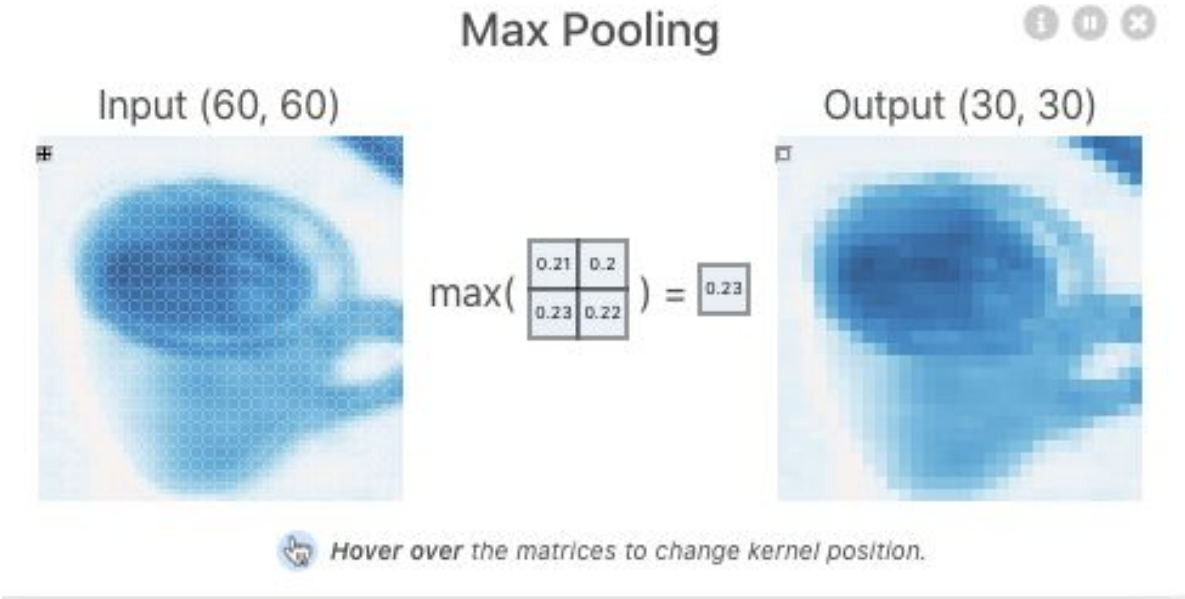
3	3	3	5	6	**7**
3	3	3	5	6	7
3	3	3	7	7	7
1	1	1	4	5	3
1	**7**	1	7	2	1
1	1	1	3	6	**9**

Filter Size: 3x3
Stride 3

3	7
7	9

DEMO

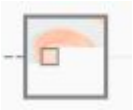
- https://poloclub.github.io/cnn-explainer/.



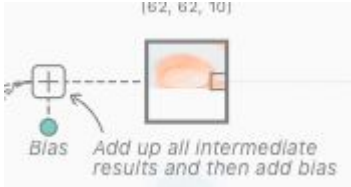
DEMO

- Visit https://poloclub.github.io/cnn-explainer/
- Look at each of the convolutional layers. What is happening at each level?
- Click on the first image under conv 1_1

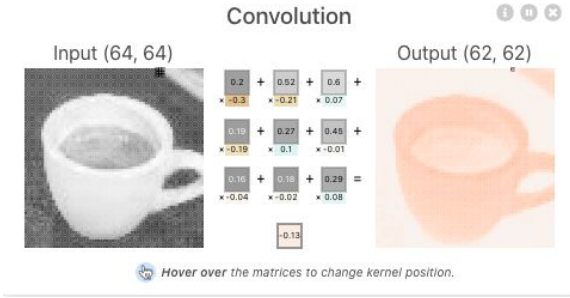
- It will show you the convolutional process of building that image.



  - It builds the image from adding the filters on the R, G, and B values together.



- Click on either the R, G, or B image to see the convolution in detail. What do you notice?



Preparazione alla pratica

Google Colab

- Per le successive esercitazioni dovrete poter accedere ad un ambiente di prototipazione Python gratuito di Google Research
- Il servizio è disponibile a questo indirizzo

https://colab.research.google.com/

Per tutte le informazioni relative al prodotto qui trovate le FAQ:
https://research.google.com/colaboratory/faq.html

Sessione pratica mercoledì 21/02/24 alle 16