



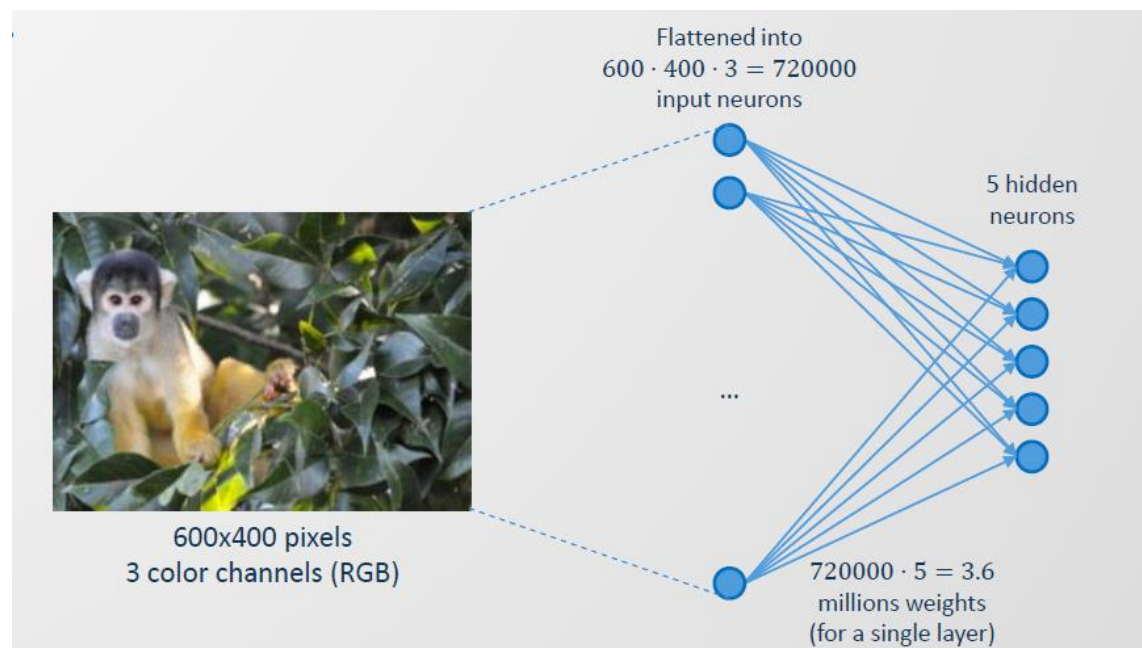
---

# Deep Learning

---

# Perché non bastano gli MLP?

- Le reti di tipo MLP sono computazionalmente troppo pesanti per essere impiegate **nell'elaborazione di immagini**, in quanto sarebbe necessario prevedere **un neurone per ogni pixel dell'immagine**.
- L'immagine viene appiattita e non si tiene conto della sua struttura 2D, si prevede un neurone per ogni pixel che è connesso con tutti i neuroni del layer successivo. Poiché ciascun neurone viene connesso a tutti i neuroni del layer successivo, **il numero di parametri da stimare sarebbe eccessivo per i casi di interesse pratico**.
- Le reti MLP **non hanno alcuna invarianza per traslazione**.



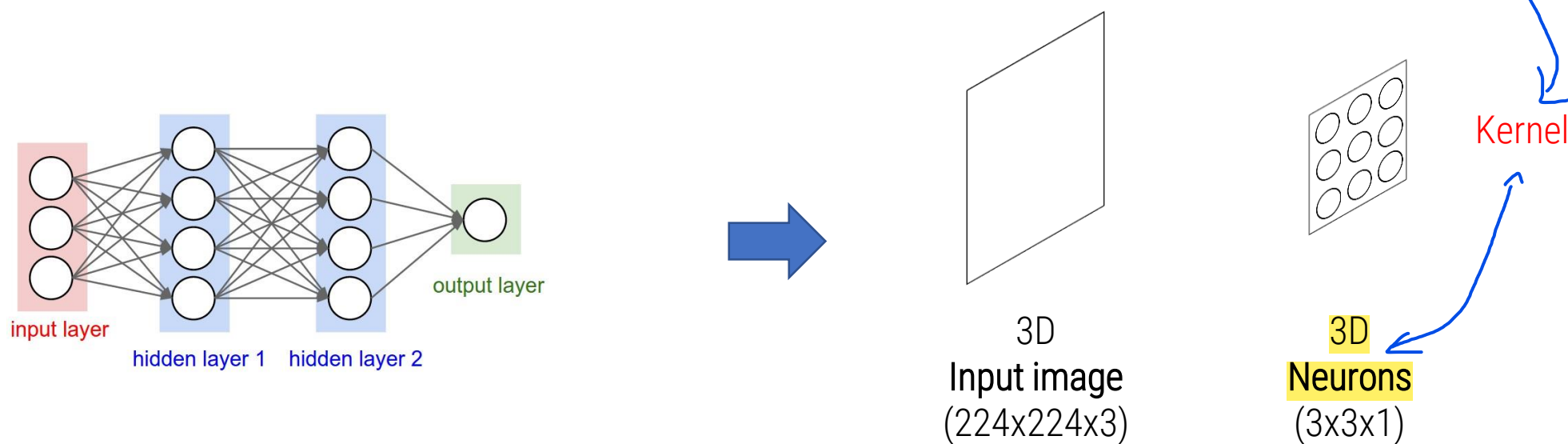
In altre parole, se un oggetto all'interno dell'immagine viene spostato, l'MLP lo tratterà come un nuovo input completamente diverso, anche se l'oggetto è lo stesso. Questo è un problema significativo perché molti compiti di visione artificiale richiedono che il modello riconosca gli oggetti indipendentemente dalla loro posizione nell'immagine.



# Convolutional Neural Network (CNN)

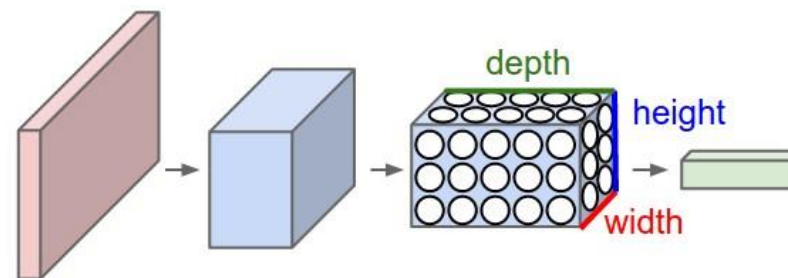
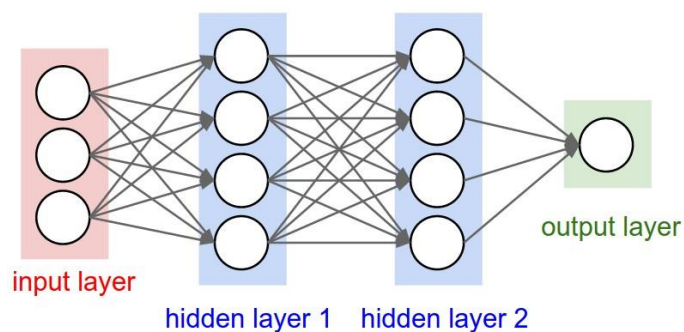
- Le **CNN** sono reti disegnate espressamente per **processare immagini**
  - Idea:** invece di “srotolare” le immagini in input (come potrei fare con un MLP)...

*...perchè non organizzare i neuroni stessi in **3 dimensioni**?*



# Convolutional Neural Network (CNN)

- A differenza del MLP, le CNN hanno una struttura a **3 dimensioni**:
  - **Larghezza** (W)
  - **Altezza** (H)
  - **Profondità** (C) → di solito  $>1$  (per questo è 3D)



- Come è possibile “collegare” il kernel con l’immagine?
  - Tramite una operazione matematica chiamata **convoluzione**!
  - La convoluzione è applicata con un meccanismo *sliding-window*



# Convoluzione

- La convoluzione è una delle più importanti operazioni di **image processing** attraverso la quale si applicano filtri digitali, per estrarre feature dalle immagini.
- Un filtro (kernel)  $h$  (una piccola maschera 2D di pesi, di dimensione  $F \times F$ ) viene fatto scorrere su ogni pixel  $(x, y)$  di un'immagine di input,  $f$ , per ogni posizione viene generato un valore di output  $g(x)$ , eseguendo il prodotto scalare tra la maschera e la porzione dell'input coperta (entrambi trattati come vettori).
- L'output  $g(x)$  prende il nome di **features map**

Presi i Valori della Maschera e dei Pixel coperti da quella maschera, quindi abbiamo due vettori  $F \times F$

$$g(x, y) = (f * h)(x, y) = \sum_{i=0}^{F-1} \sum_{j=0}^{F-1} f(x, y) h(x - i, y - j)$$

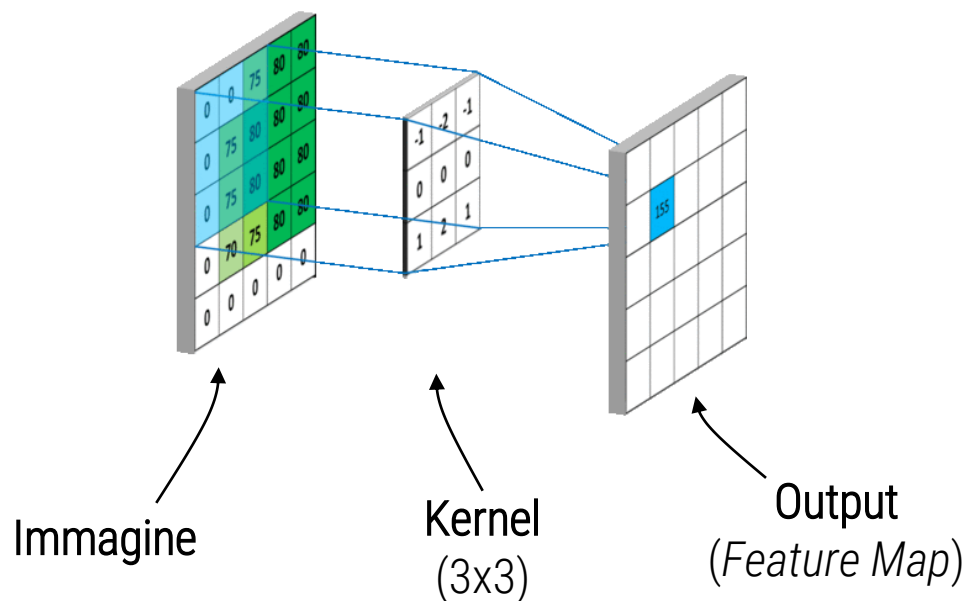
Pixel

Immagine di Input

Filtro (Kernel)

# Convolutional Neural Network (CNN)

- La convoluzione è l'elemento costruttivo di base di una rete CNN
  - Ogni **kernel** è convoluto con i dati in input, generando una **feature map**



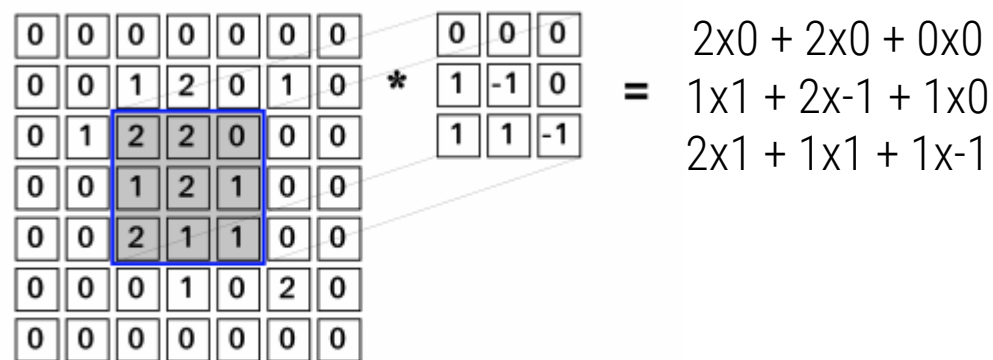


Diagram illustrating the convolution process with a 7x7 input grid, a 3x3 kernel, and the resulting 5x5 output grid. The kernel is highlighted in blue. The calculation for the center value (15) is shown as a dot product of the kernel and the corresponding 3x3 region of the input grid.

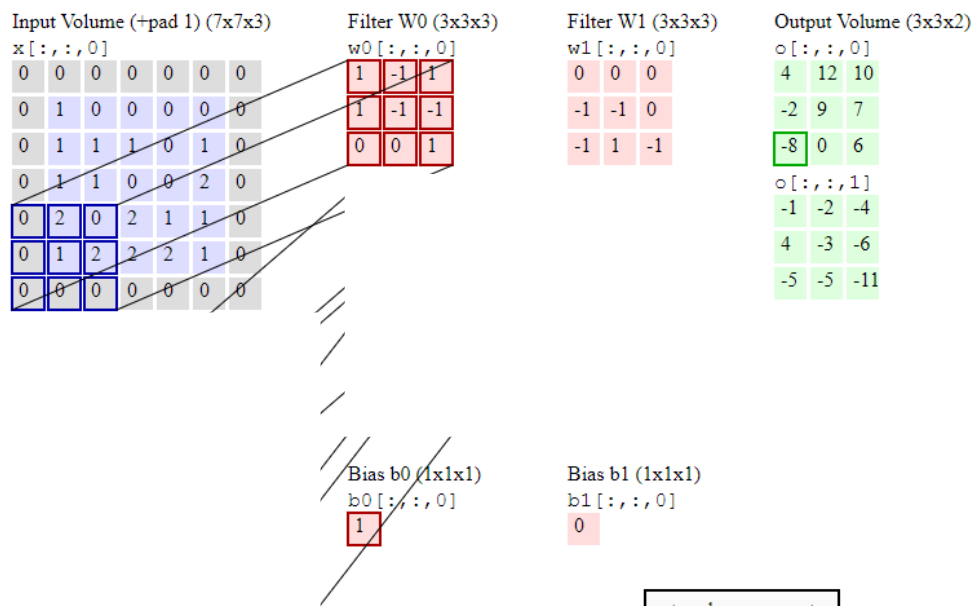
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & 1 & 2 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} 2 \times 0 + 2 \times 0 + 0 \times 0 \\ 1 \times 1 + 2 \times -1 + 1 \times 0 \\ 2 \times 1 + 1 \times 1 + 1 \times -1 \end{bmatrix}$$

- Generalmente, profondità >1 → più kernel → una feature map prodotta per ogni kernel
- Il volume di output è quindi ottenuto mettendo assieme tutte le feature map prodotte

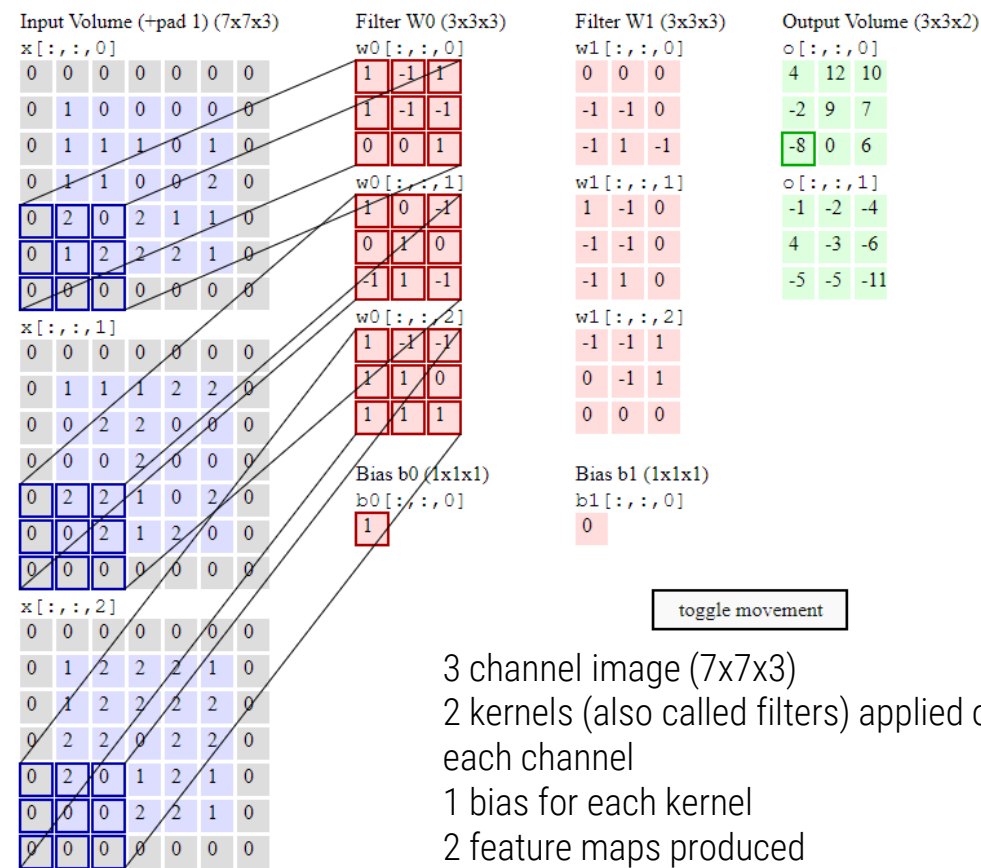


# Convolutional Neural Network (CNN)

- Quindi, ora, i pesi appresi (dell'MLP) sono raggruppati nel kernel!
  - Ogni cella è un singolo peso che viene appreso

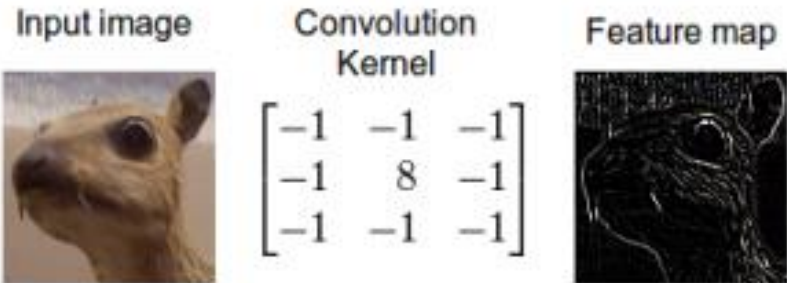


1 channel image (7x7x1)  
2 kernels (also called filters)  
1 bias for each kernel  
2 feature maps produced

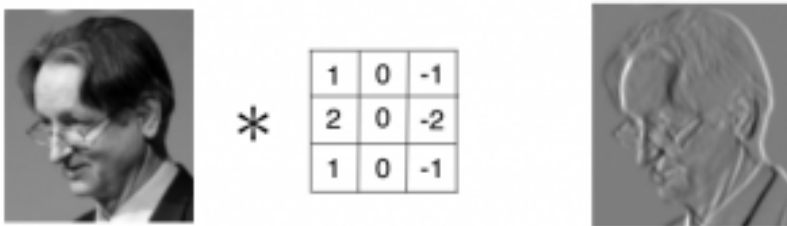


3 channel image (7x7x3)  
2 kernels (also called filters) applied on each channel  
1 bias for each kernel  
2 feature maps produced

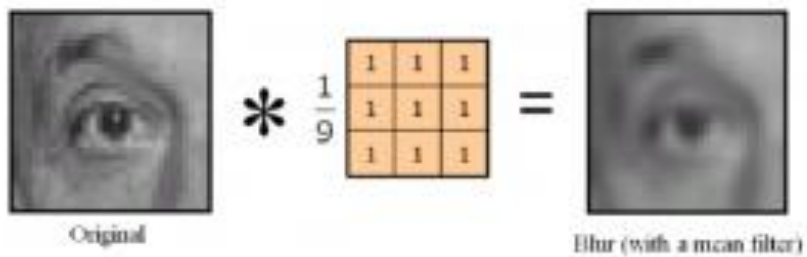
# Convolutional Neural Network (CNN)



Esempio di kernel che evidenzia i contorni di una immagine.



Esempio di kernel che produce un effetto embossed.

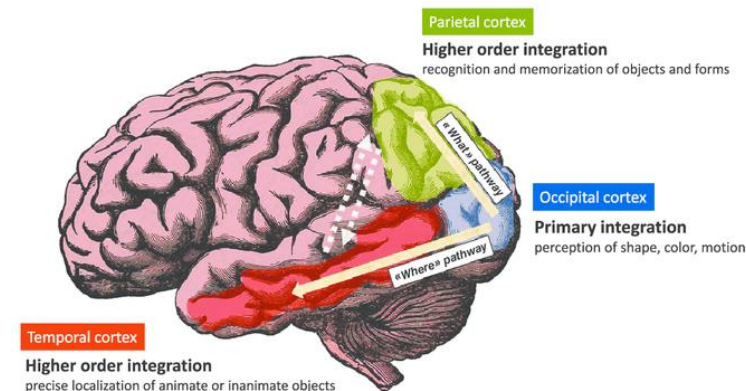
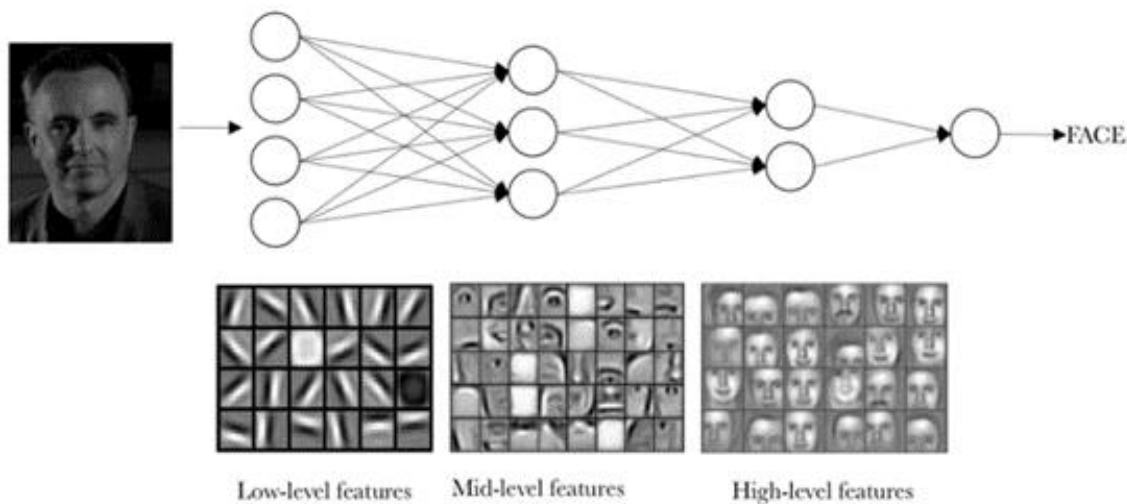


Esempi di kernel che produce un effetto blur



# Convolutional Neural Network (CNN)

- I layer convolutive estraggono vari tipi di informazione visual in maniera gerarchica
  - Nei layer vicini all'input, estraggono informazioni "semplice"
  - Nei layer ~~vicini~~<sup>lontani</sup> all'input, estraggono informazioni "complessa"



- La cosa interessante è che anche nel nostro cervello succede una cosa simile!



## Corteccia visiva

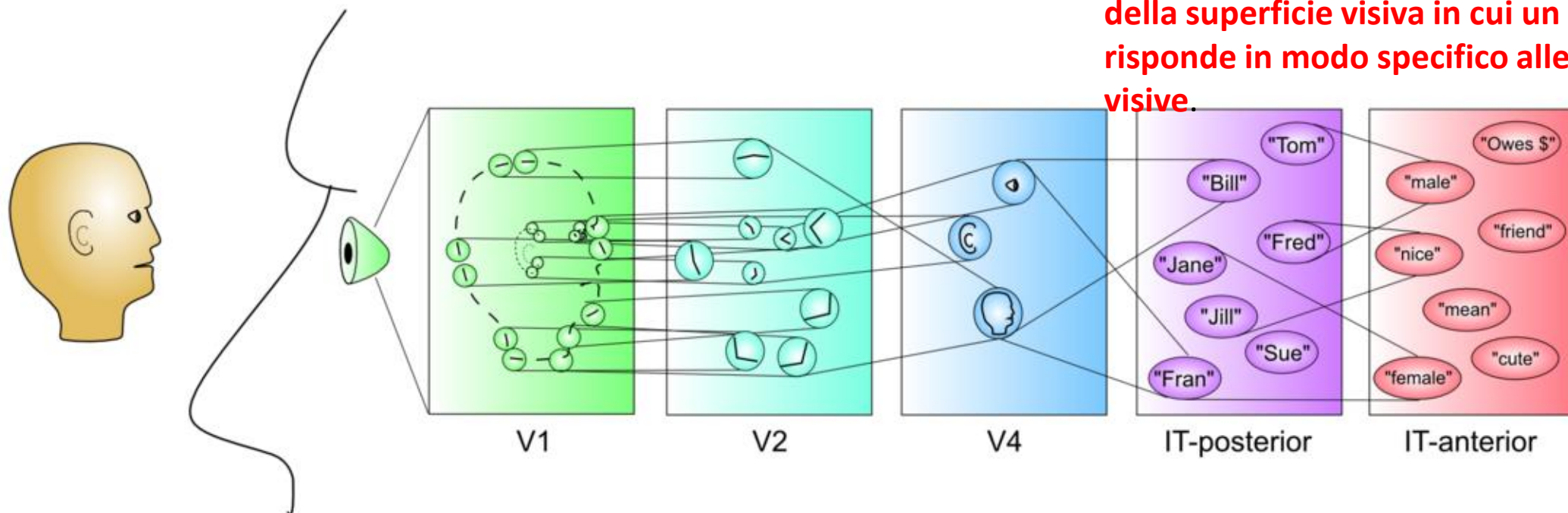
---

- Nel 1965 DH Hubel e TN Wiesel hanno dimostrato che i mammiferi percepiscono visivamente il mondo che li circonda utilizzando un'architettura a strati di neuroni nel cervello.
  - La struttura della **corteccia visiva è a strati. Man mano che le informazioni passano dai nostri occhi al cervello, si formano rappresentazioni di ordine sempre più alto.**
-

## VISIONE UMANA

Il campo recettivo di un neurone visivo è l'area della superficie visiva (ad esempio, la retina nell'occhio) dalla quale quel neurone riceve e risponde a stimoli specifici.

Il **campo recettivo** di un neurone visivo è l'**area della superficie visiva in cui un dato neurone risponde in modo specifico alle informazioni visive**.



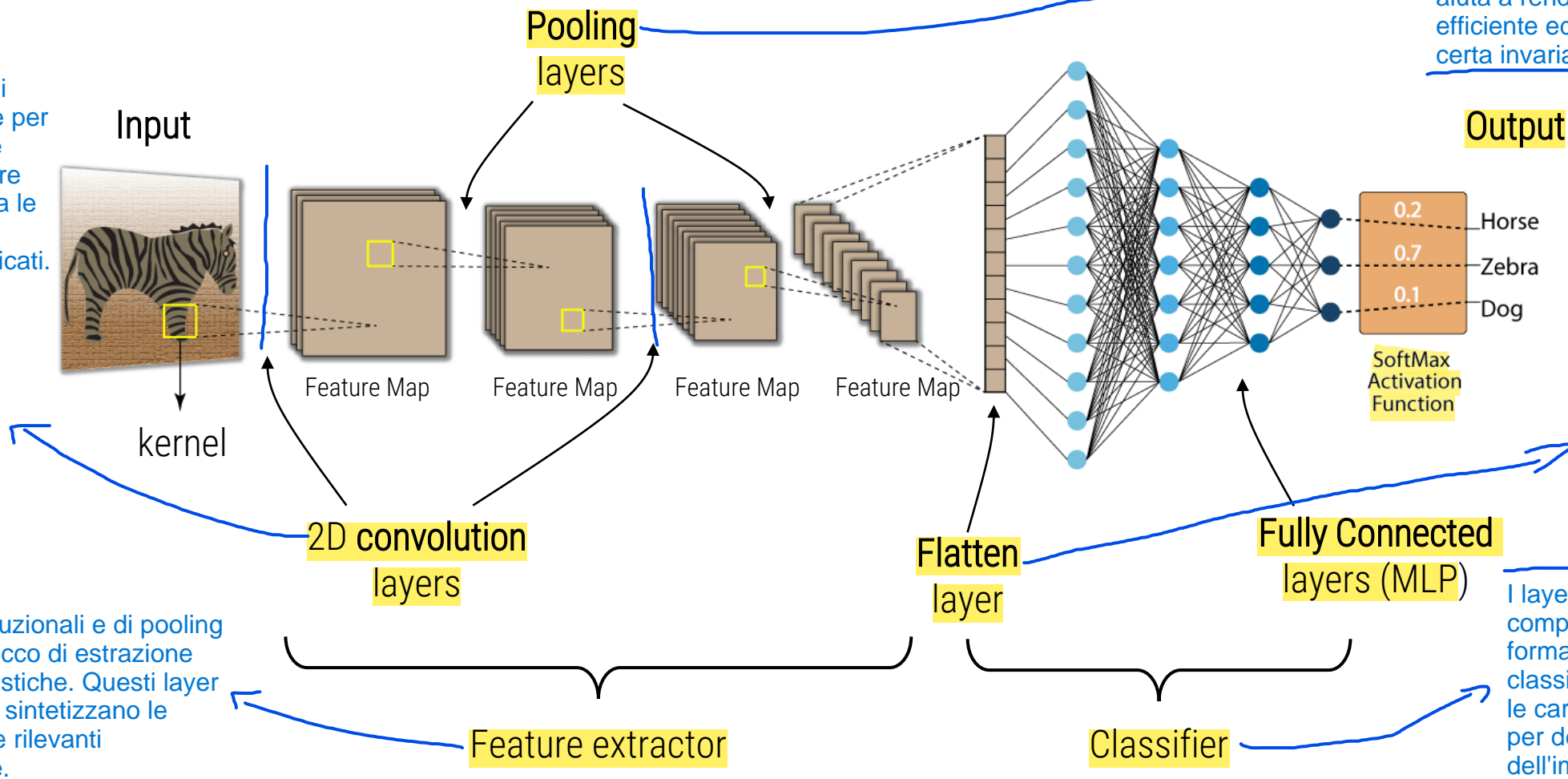
- Il sistema visivo umano elabora le caratteristiche dell'oggetto in un **approccio feed-forward e gerarchico**.
- **La gerarchia inizia dalla Corteccia Visiva Primaria (V1)**, caratterizzata da neuroni con campi recettivi piccoli e specializzati, elabora le informazioni visive di base, bordi e linee.
- **Queste informazioni vengono poi elaborate dalle Aree visive (V2 e V4)**, specializzate nella percezione di informazioni visive più complesse, come forme e oggetti (e sono caratterizzate da neuroni con campi recettivi più estesi).
- Infine, le informazioni visive vengono elaborate dalla **corteccia inferotemporale (IT)**, che è specializzata nella percezione di oggetti complessi, come il riconoscimento facciale (campi recettivi più ampi e completi)

# Convolutional Neural Network (CNN)

- Una CNN “tradizionale” è composta da un insieme di layer sequenziali:

I layer di pooling riducono la dimensionalità delle feature map, combinando i valori delle celle vicine in una sola. Questo è solitamente fatto tramite operazioni come il max pooling o l'average pooling, che riducono le dimensioni spaziali delle feature map mantenendo le informazioni più rilevanti. Questo processo aiuta a rendere il modello più efficiente ed a introdurre una certa invarianza alle traslazioni.

I layer convoluzionali applicano diversi filtri all'immagine per creare le feature map. Ogni feature map rappresenta le risposte dei vari pixel ai filtri applicati. Questi layer convoluzionali rilevano le caratteristiche dell'immagine.

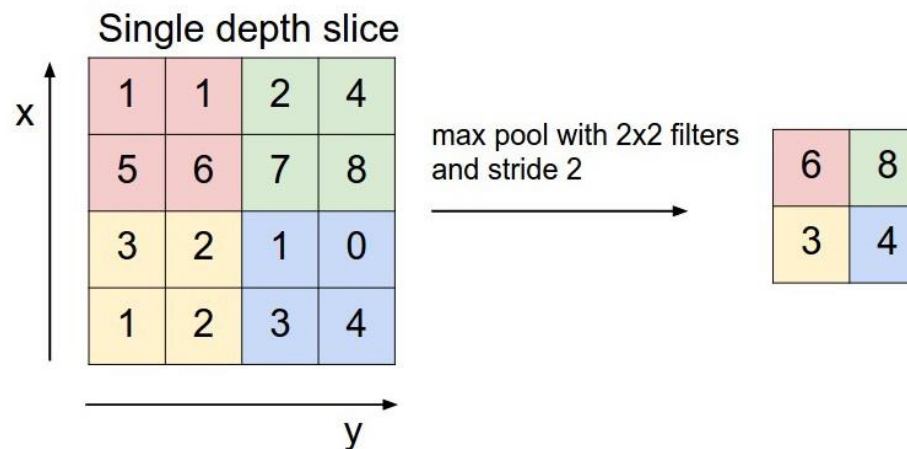
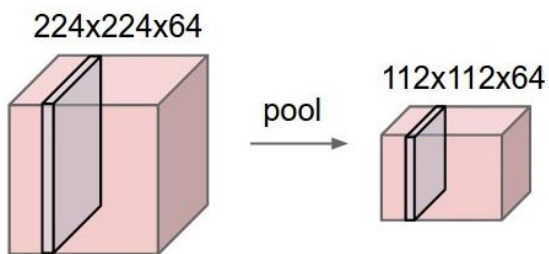


Dopo diversi strati convoluzionali e di pooling, le feature map risultanti vengono appiattite in un vettore unidimensionale. Questo passaggio è necessario per passare le caratteristiche estratte alla parte finale della rete, che è costituita da strati completamente connessi (MLP).

I layer appiattiti e completamente connessi formano il blocco di classificazione. Utilizzano le caratteristiche estratte per determinare la classe dell'immagine di input.

## Altri tipi di layer: pooling layer

- Il layer di pooling **riduce la dimensionalità** del volume di input
- Il layer di pooling viene spesso **utilizzato per una serie di ragioni**:
  - **Invarianza** alla posizione
  - **Riduzione del costo computazionale** (i volumi sono più piccoli)
  - Aiuta a prevenire l'**overfitting**
- Esistono **diverse funzioni di pooling** che possono essere usate: **max** (most used), **average**, ...

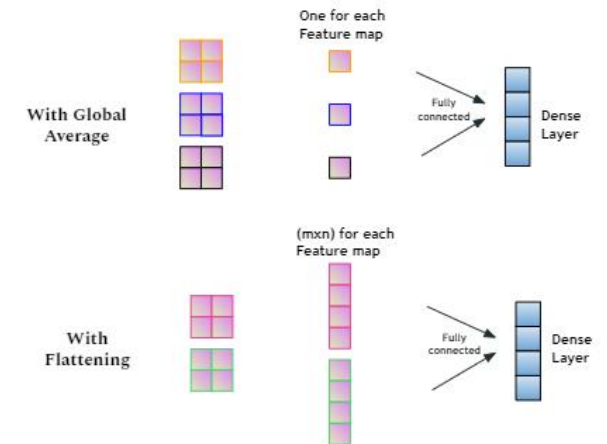
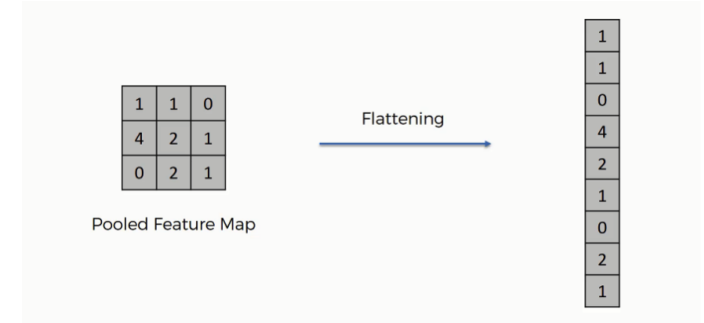






# Altri tipi di layer: flatten layer

- **Layer di attivazione** → vedi *la funzione di attivazione*
- **Flatten layer**
  - Tradizionalmente utilizzato per connettere il feature extractor con il classifier delle CNN
  - «Srotola» il volume di input, attenzione alla dimensionalità!
    - $12 \times 12 \times 64 \rightarrow 12 \times 12 \times 64 \rightarrow 9216$
- È possibile anche utilizzare altre tecniche per «srotolare» il volume di input





# CNN: Architetture

- Quella vista è un'architettura di base, ne esistono molte altre:
  - Migliorare le prestazioni
  - Alleggerire il carico computazionale
  - Task specifici
- Nelle nostre esercitazioni, utilizzeremo alcune architetture molto note in letteratura:

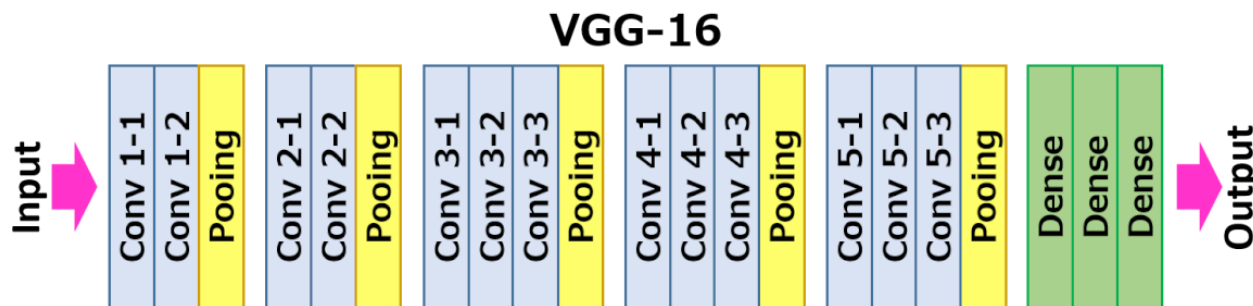
- VGG

- ResNet

- AlexNet

- ...

- Tutte queste possono essere usato **pre-trained** (già addestrate).

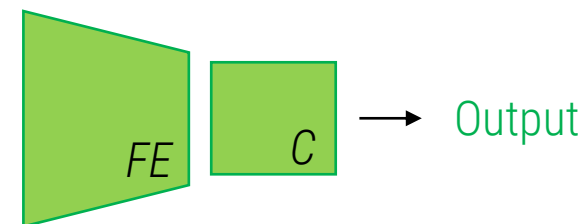




# CNN: Tipologie di training

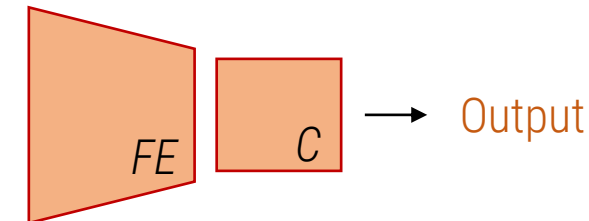
## 1. From scratch

- La rete è addestrata partendo da una configurazione random dei pesi
- Sono richiesti molti dati di training
- Procedura di training lunga e complessa



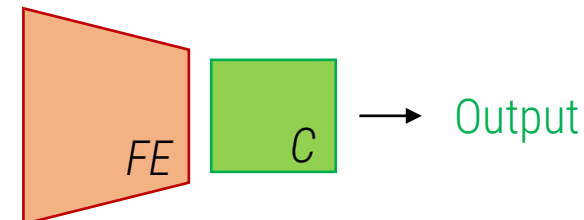
## 2. Pre-trained

- Viene utilizzata una rete già addestrata
- L'output corrisponde a quello originale
- No training, no bisogno di dati



## 3. Fine-tuned

- Viene utilizzata una rete già addestrata
- Vengono “congelati” i pesi del feature extractor
- Cambio solo i pesi del classifier (per esempio, per cambiare il numero delle classi).







# CNN: Tipologie di Reti

## Modelli feedforward « discriminativi » per la classificazione (o regressione) supervisionati

- **FC-DNN: Fully Connected DNN** (MLP con almeno due livelli hidden)
- **CNN**: Convolutional Neural Network (o ConvNet)

## Modelli ricorrenti con memoria e attenzione (utilizzati per sequenze)

- **RNN**
- **Recurrent Neural Network**
- **LSTM**
- **Long Short Term Memory**
- **Transformers**

**Addestramento non supervisionato**: modelli addestrati a ricostruire l'input originale prendendo come input una versione a più bassa dimensionalità ( utilizzati per denoising, anomaly detection)

- **Autoencoders**

## Modelli « generativi » per generare dataset sintetici (data augmentation) style transfer, art applications

- **GAN** Generative Adversarial Networks
- **VAE** Variational Autoencoders

## Reinforcement learning (per apprendere comportamenti)

- **Deep Q Learning**



# Training: ingredienti necessari

## 1) Big Data

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.

## 2) 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**.

