

LA RECOGNITION DI FIGURE ANOMALE

SILVIA PARMEGGIANI, EDMONDO CERINI, MATTEO VENTURI, TOMMASO GIURIATO

A CURA DI:

SOMMARIO

1. Triangolo di Kanizsa



2. Leggi della Gestalt



3. Anatomia dell'Occhio e percorso della Luce ed Immagini



4. Convolutional Neural Networks



5. Algoritmi principali di Recognition:



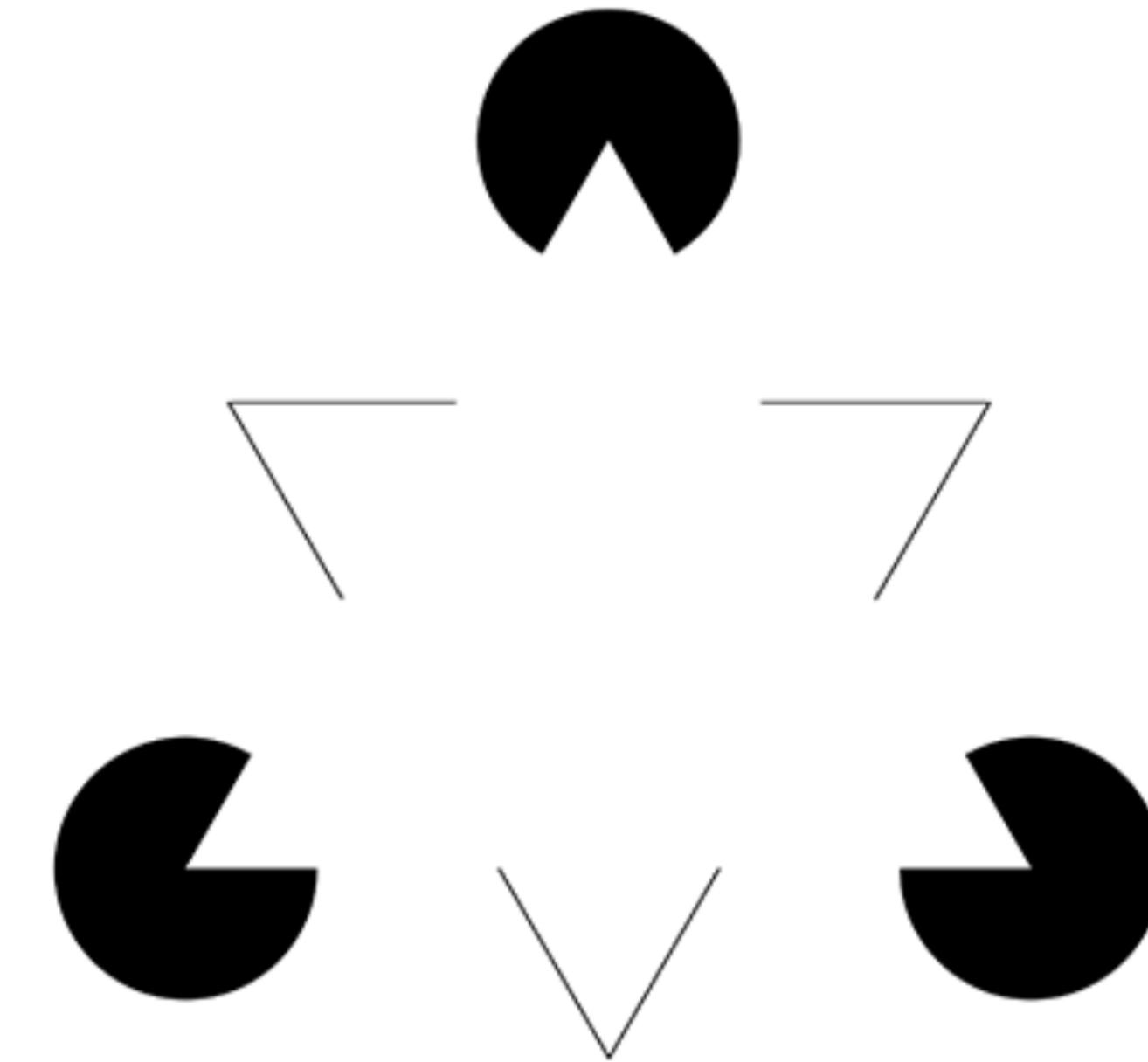
- Canny Edge Detection
- Hough Line Transform

1.

Il Triangolo di Kanizsa



4



Kanizsa, G. (1955)

"Margini quasi-percettivi in campi con stimolazione omogenea", Rivista di Psicologia, 49(1), 7-30



Esperimento sui macachi

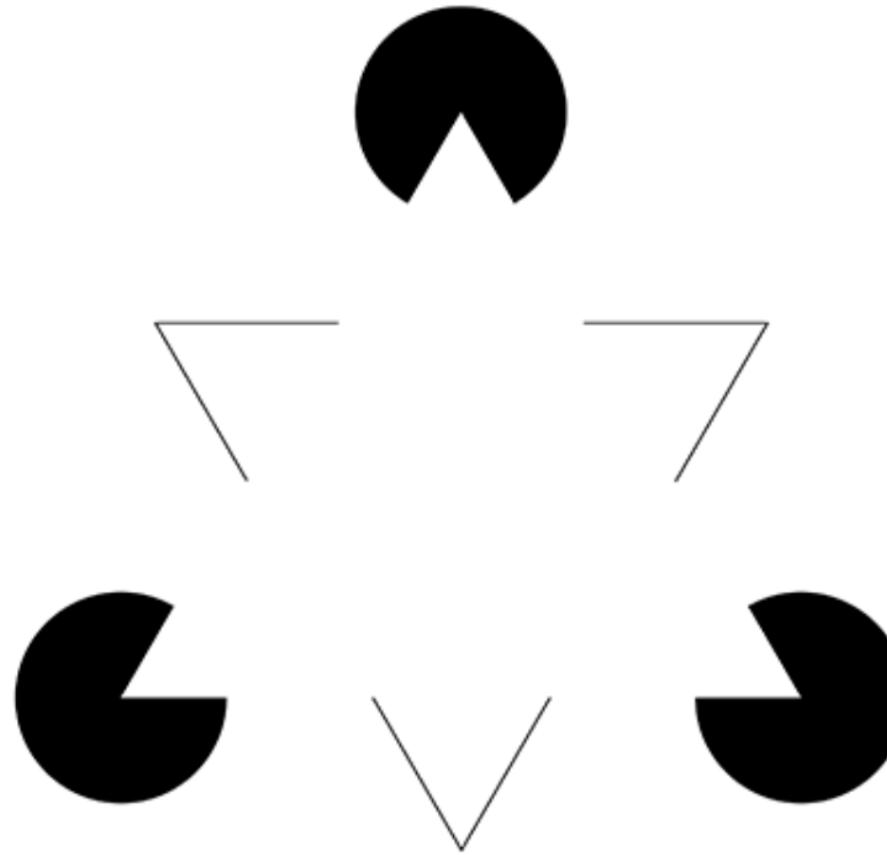
I neuroscienziati hanno compiuto vari esperimenti utilizzando il triangolo di kanizsa per capire in che modo il nostro cervello osservava le cosiddette figure anomale e venga ingannato da esse.

Hanno mostrato l'illusione a dei macachi e hanno notato come i neuroni della zone V2 e V1 della regione della corteccia visiva si attivano quando gli inducenti sono ben posizionati in modo da creare l'illusione.



6

Esperimento sui macachi





7

Perchè vediamo un triangolo?

Il cervello si comporta da detective e risponde agli stimoli e segnali dell'ambiente e facendo le sue migliori ipotesi su come si adattano insieme, in questo caso si giunge ad una conclusione errata.

In poche parole vediamo ciò che non esiste davvero.

2.

Leggi della Gestalt

La teoria della Gestalt

Secondo la Gestalt o teoria della forma, le nostre esperienze sono strutturate: la percezione non si presenta come somma degli elementi sensoriali, ma ci basiamo su una forma complessiva dello stimolo visivo.

La Gestalt si basa sull'osservazione che il "tutto" è qualcosa di più della "somma delle parti" e propone il concetto di "Gestalt" (che significa "intero" o "struttura") quale principale unità d'analisi nell'ambito della percezione (visiva e non).

Figura-Sfondo

Lo sfondo ha una grande importanza infatti esso non scompare mai completamente dalla visione continuando a influenzare la percezione della figura. Figura e sfondo non possono mai essere letti contemporaneamente. E' una condizione necessaria per la comprensione di ina qualsiasi immagine.

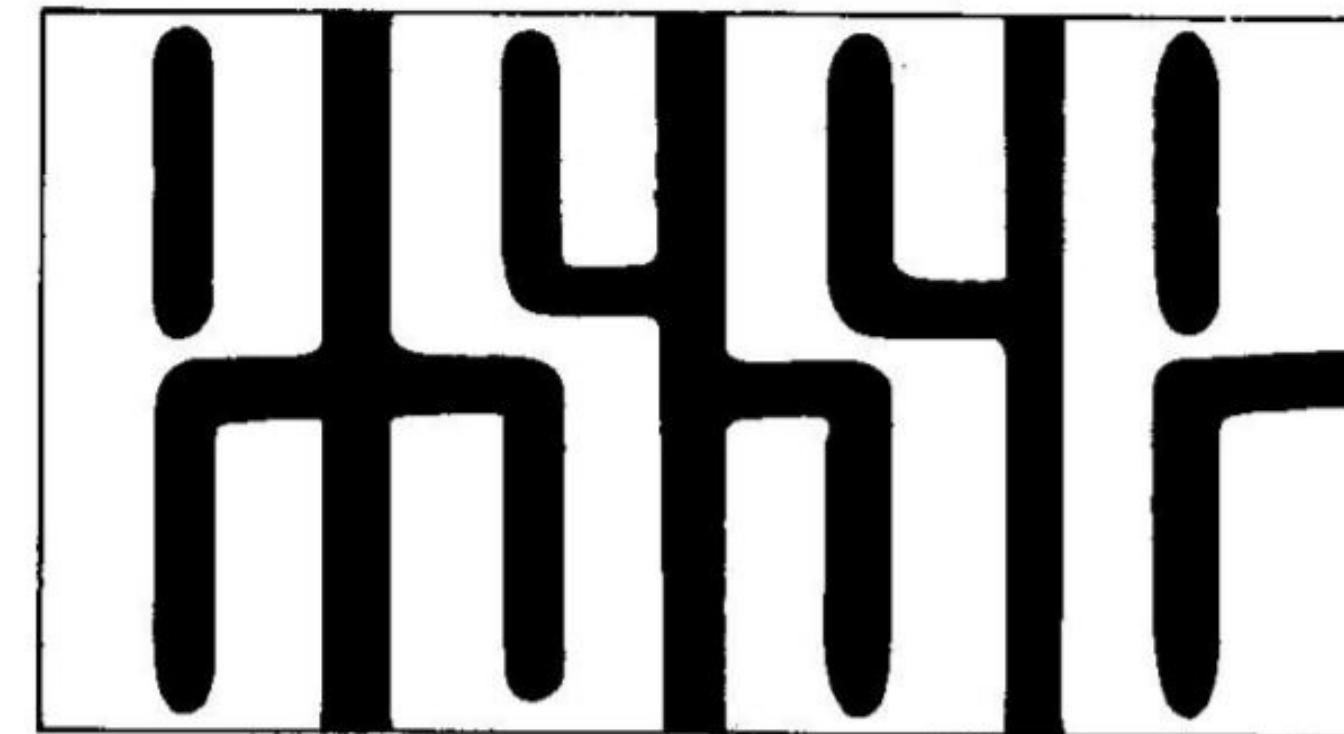
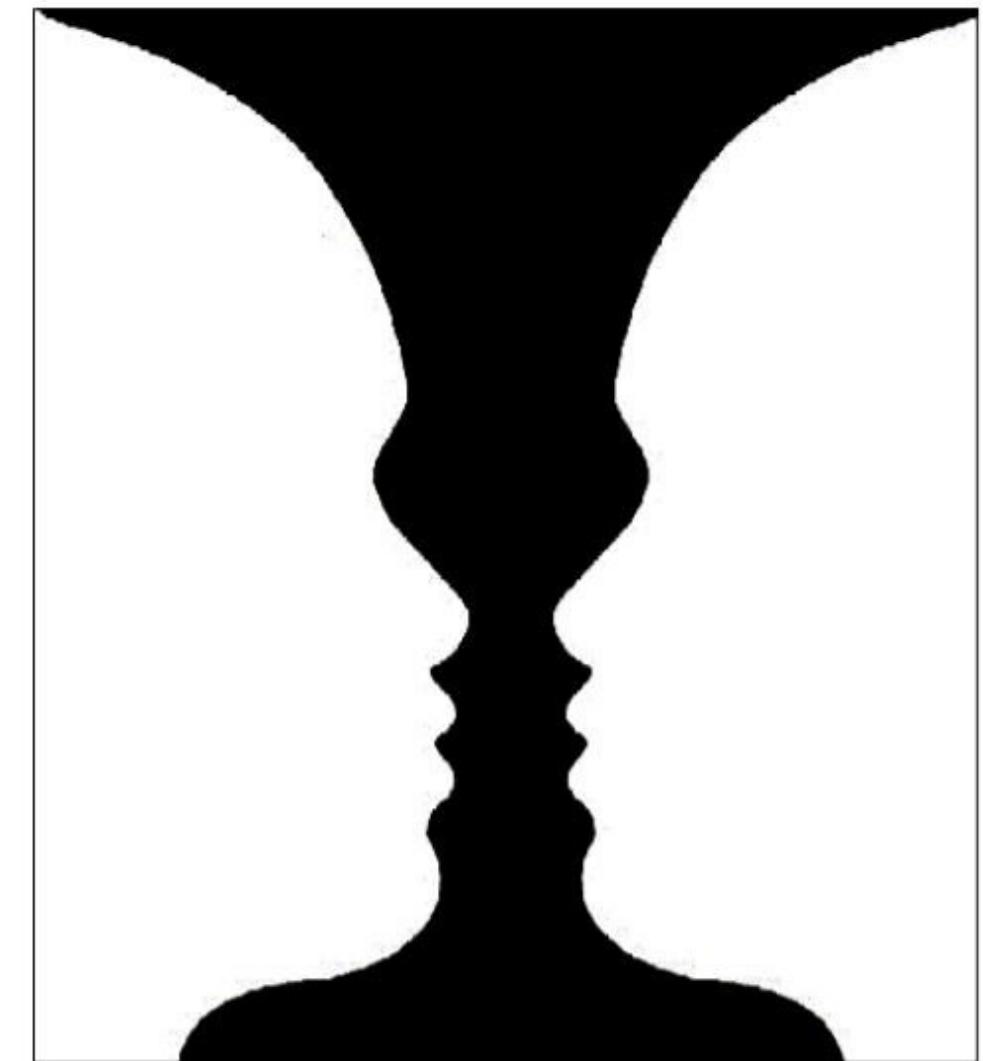


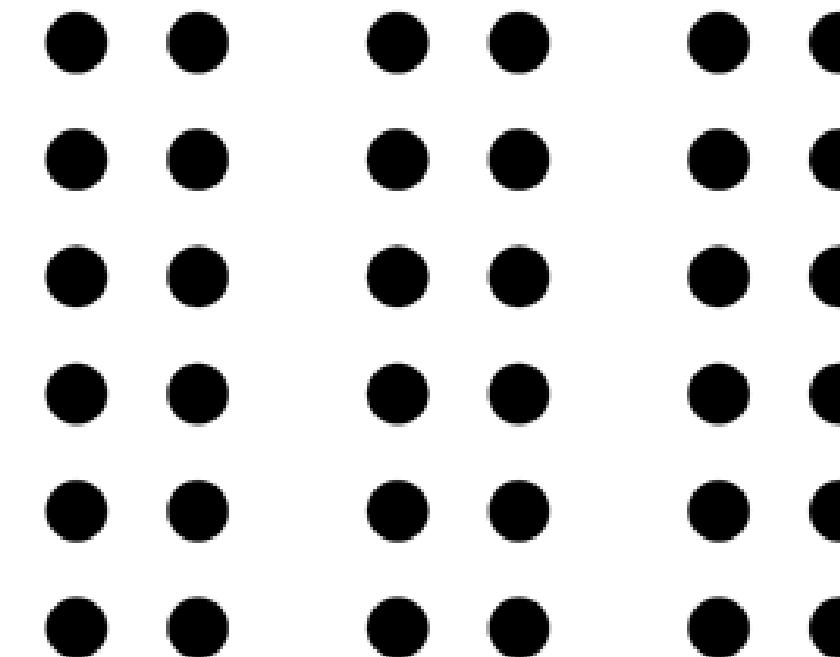
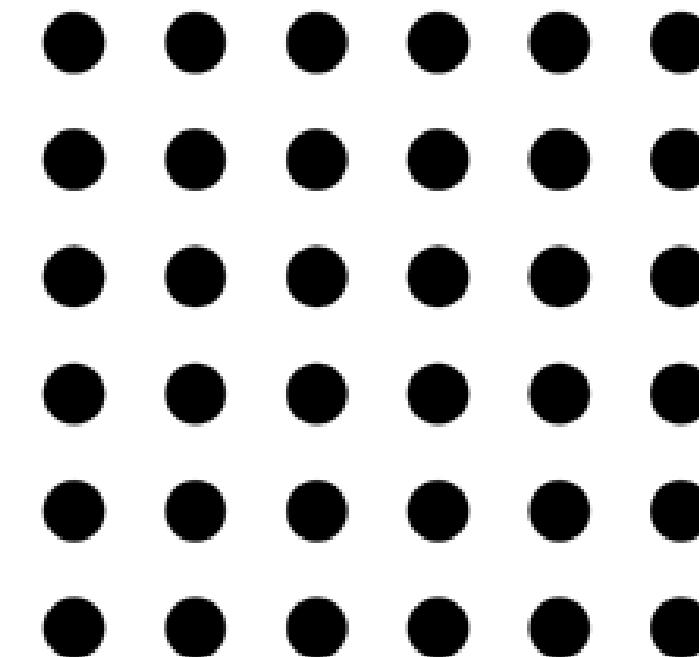
Figure ambigue

Sono immagini che raffigurano dei volti, delle facce, dei ritratti, i cui tratti essenziali possono diventare a loro volta degli oggetti, delle forme, altri volti, ecc. A seconda di come il nostro cervello focalizza le immagini si avranno diverse interpretazioni.



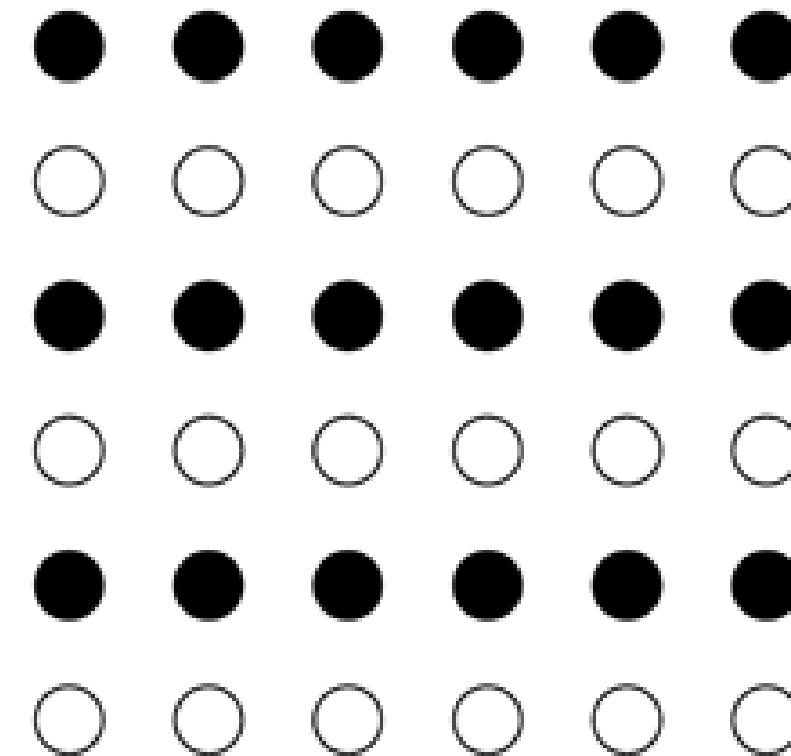
Legge della vicinanza

Gli elementi che stanno “vicini” tendono ad essere percepiti come insieme unitario.



Legge della somiglianza

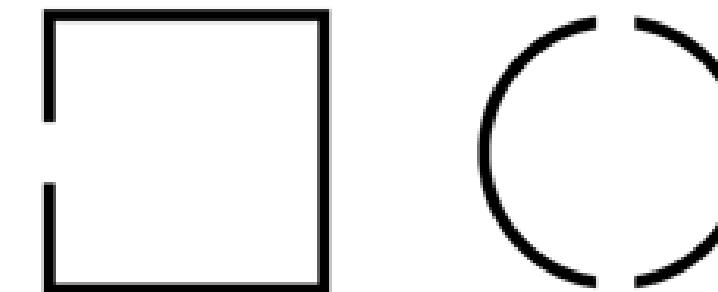
Gli elementi di un gruppo simili (per forma, colore o altro parametro) vengono percepiti come unità.



Legge della chiusura o completamento

Gli elementi di un insieme che tendono a chiudersi in forme riconoscibili vengono percepiti come figura. Il cervello riesce a ricostruire le parti mancanti della figura grazie alla memoria che ha di quel particolare oggetto.

CHIUSURA

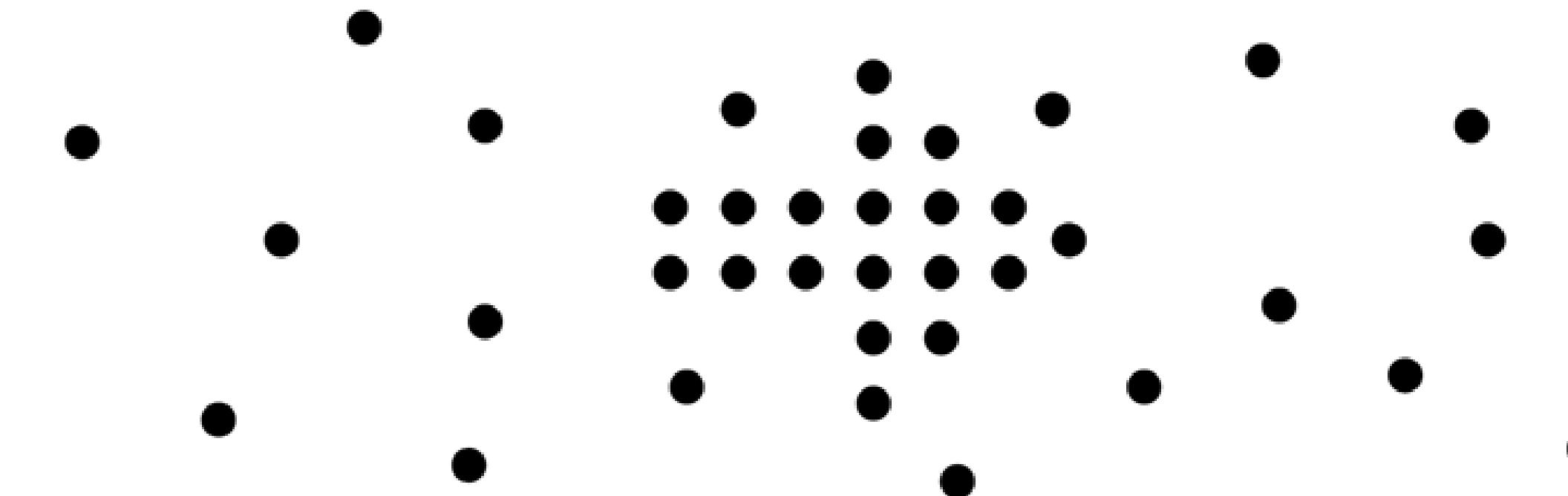


COMPLETAMENTO

GRAFICA

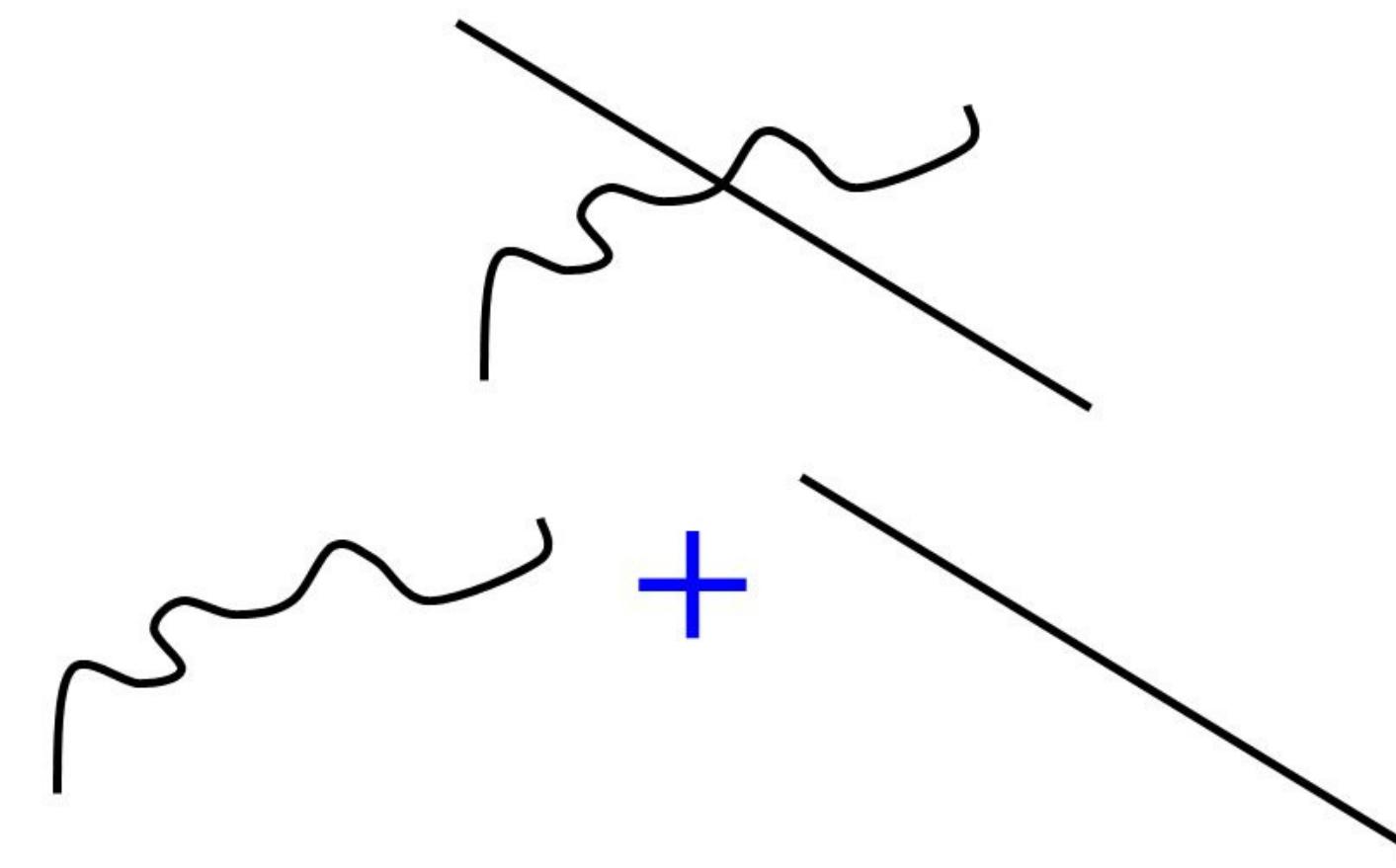
Legge del destino comune

Gli elementi di un gruppo che condividono le stesse direzioni di movimento, di ritmo, di orientamento, sono percepiti come insieme unitario.



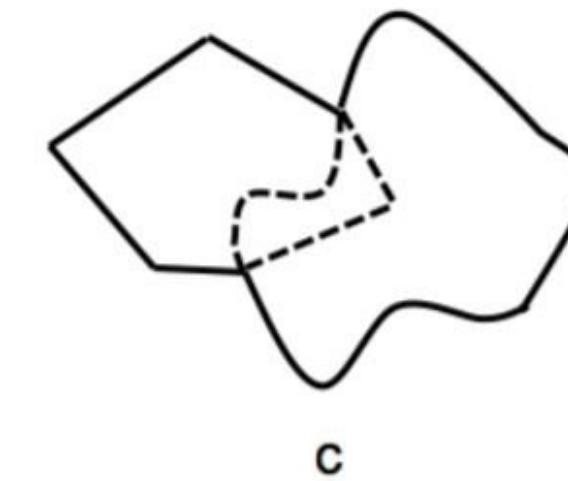
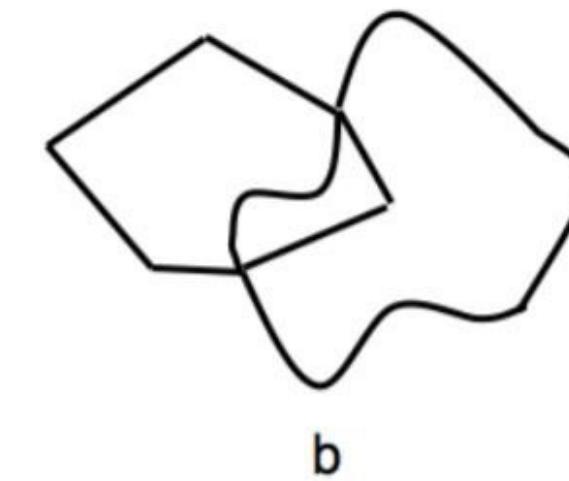
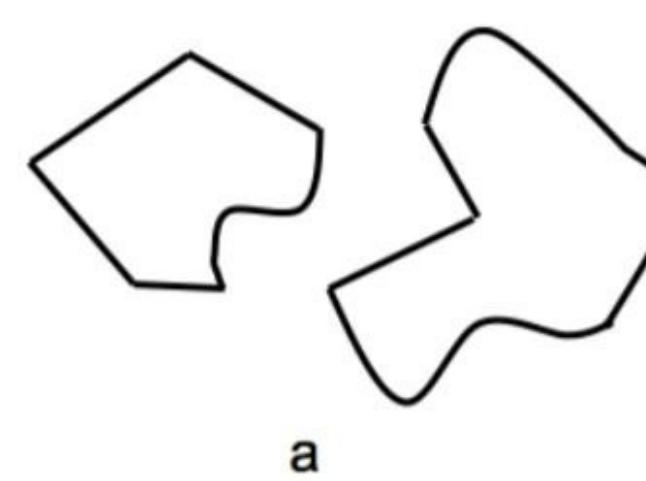
Legge della continuità di direzione

Gli elementi di un insieme che si susseguono in una continuazione regolare e logica sono percepiti come figura.



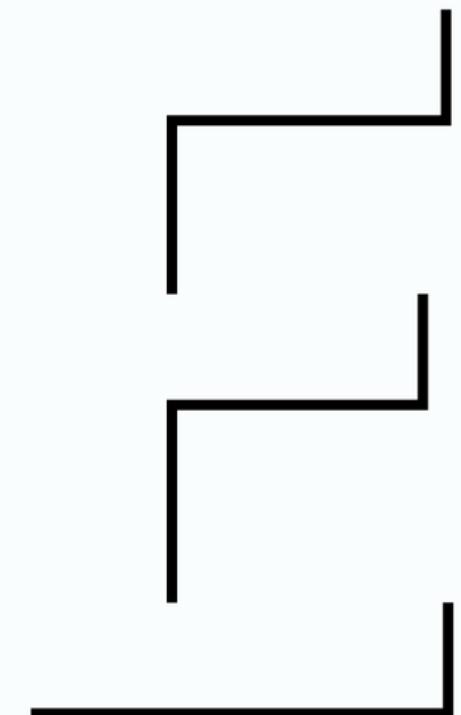
Legge della pregnanza

Gli elementi che producono forme più semplici, simmetriche, regolari e unitarie tendono ad essere percepiti unitariamente.



Legge dell'esperienza passata

Gli elementi di un insieme che riescono a far rivivere le nostre esperienze percettive di un dato oggetto, tendono ad essere raggruppati e a formare una figura.



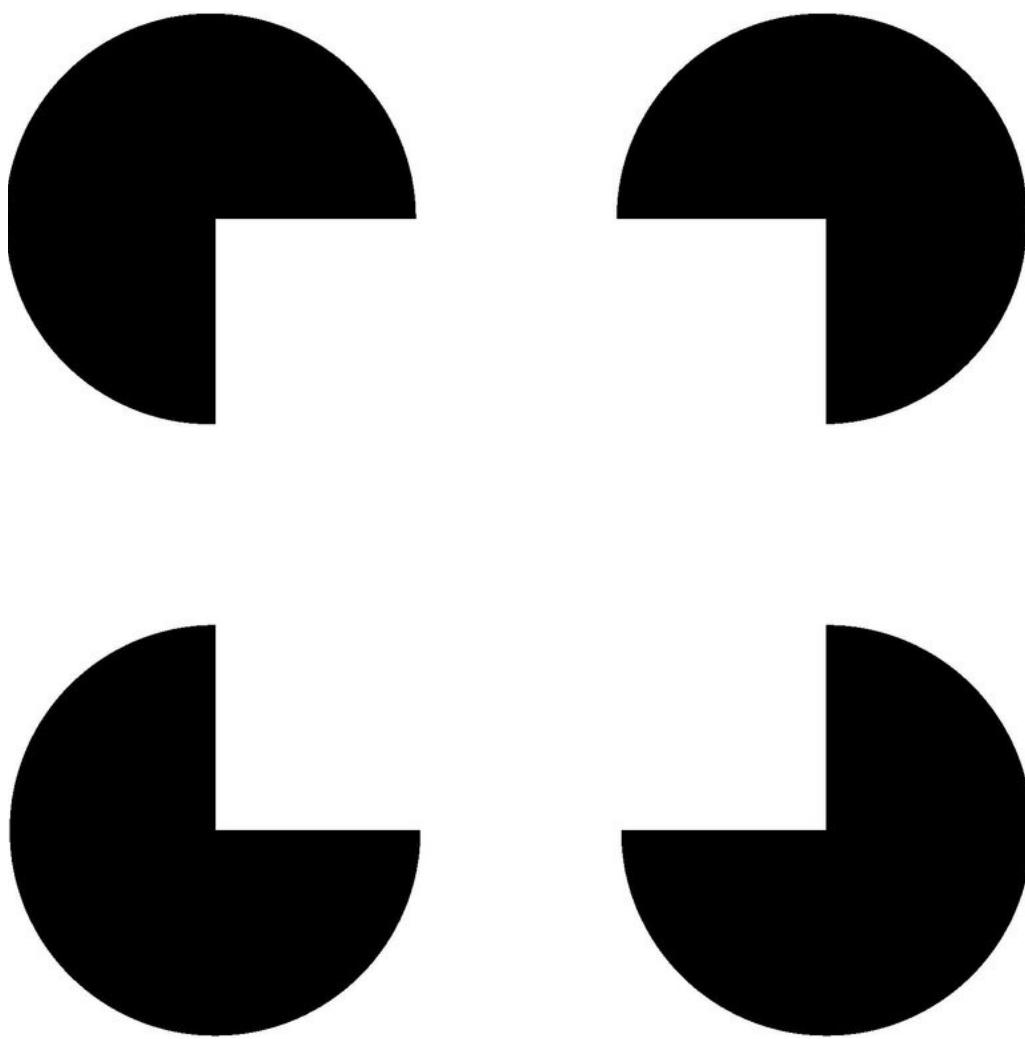
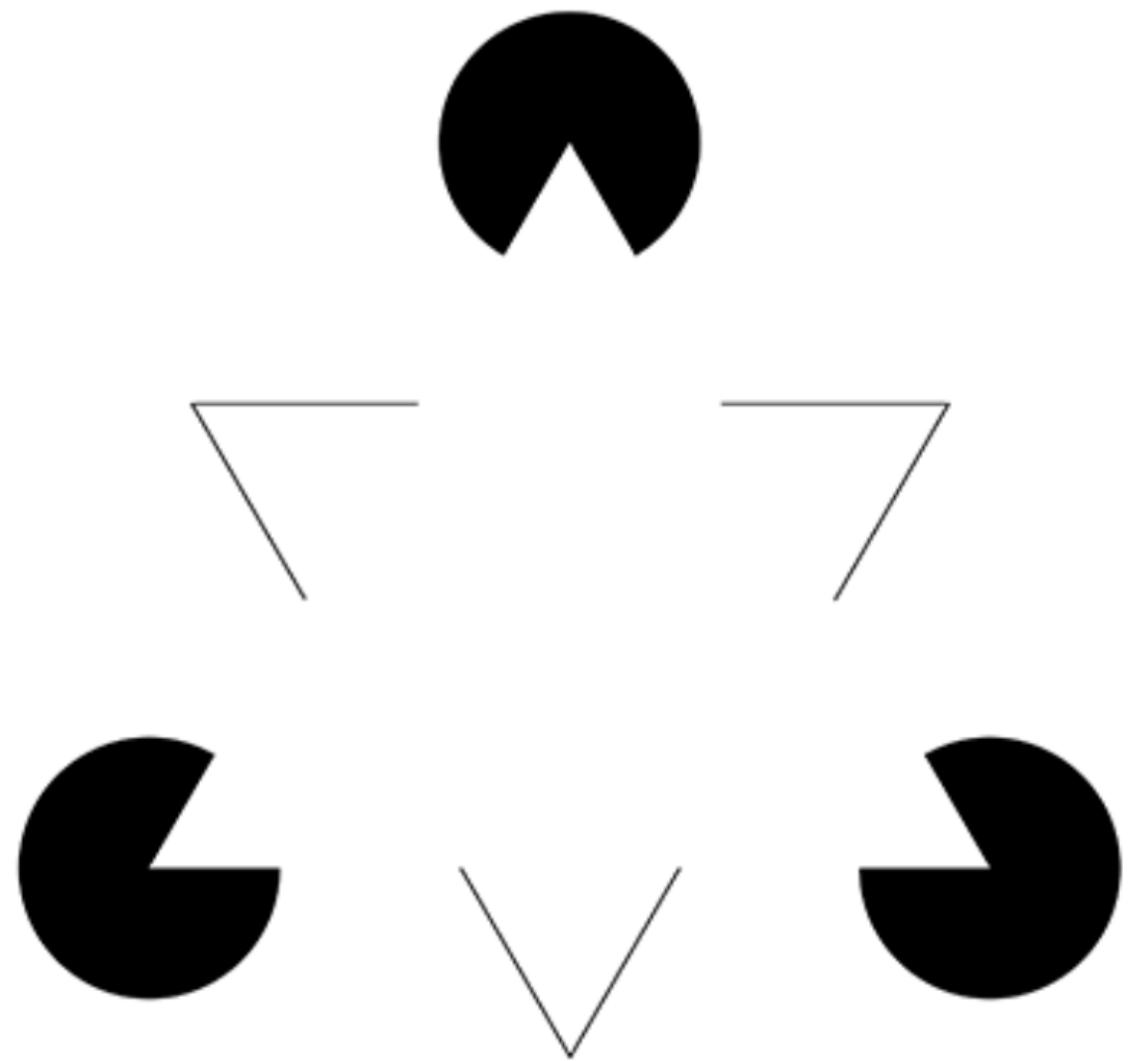
3. Anatomia dell'Occhio e Percorso della Luce ed Immagini





20

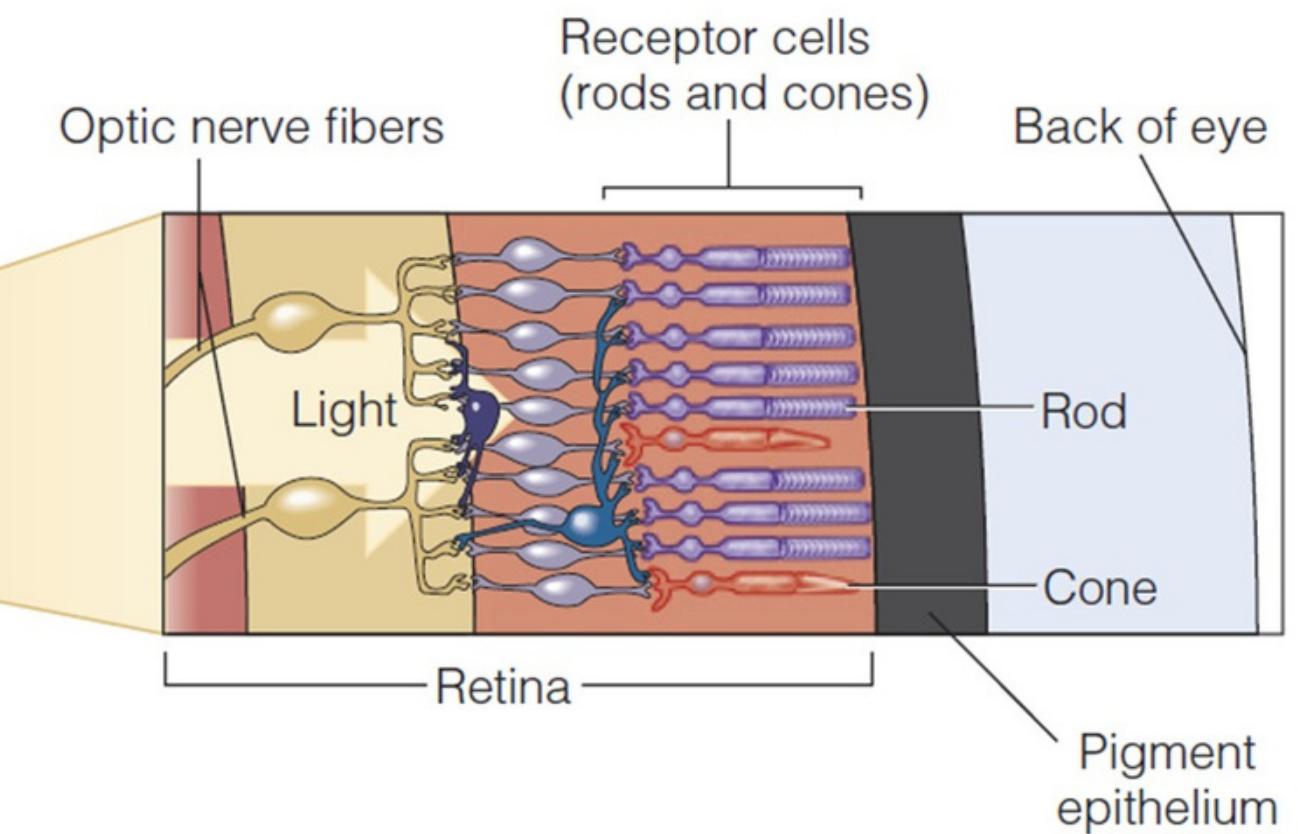
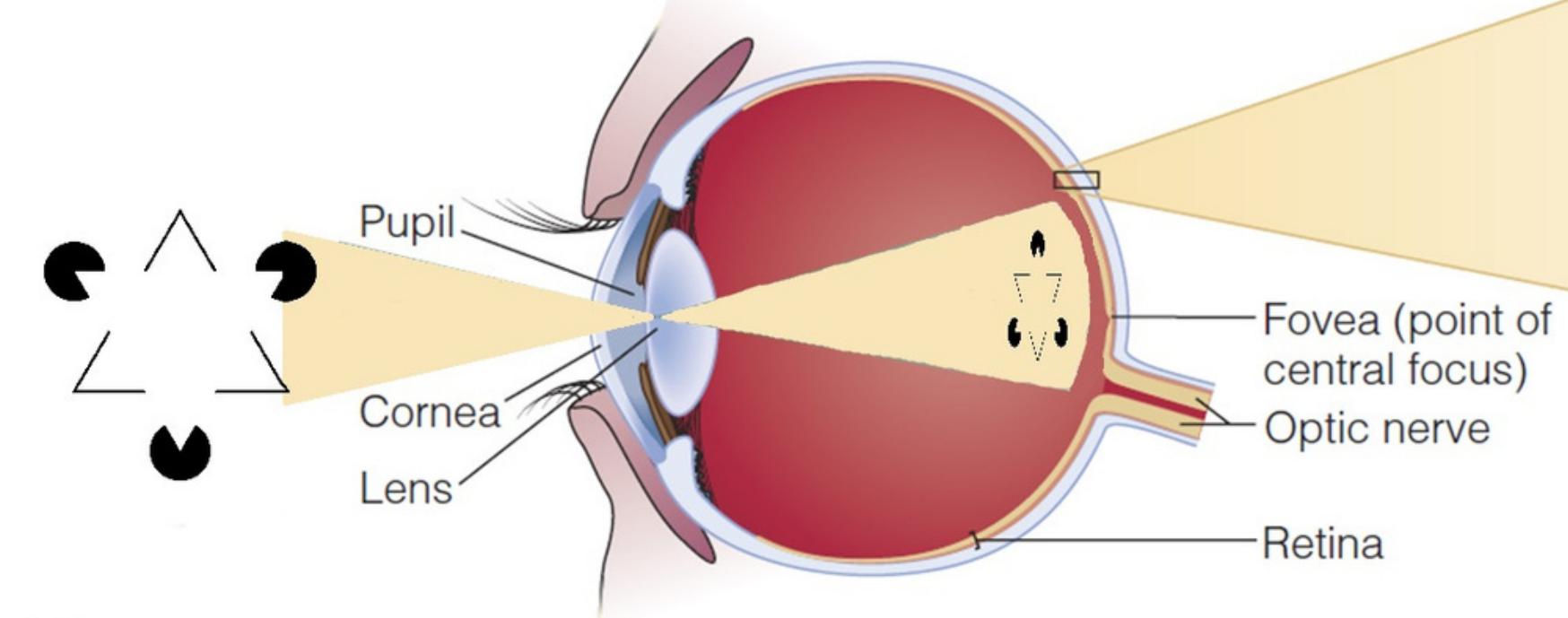
Le illusioni di Kanizsa





21

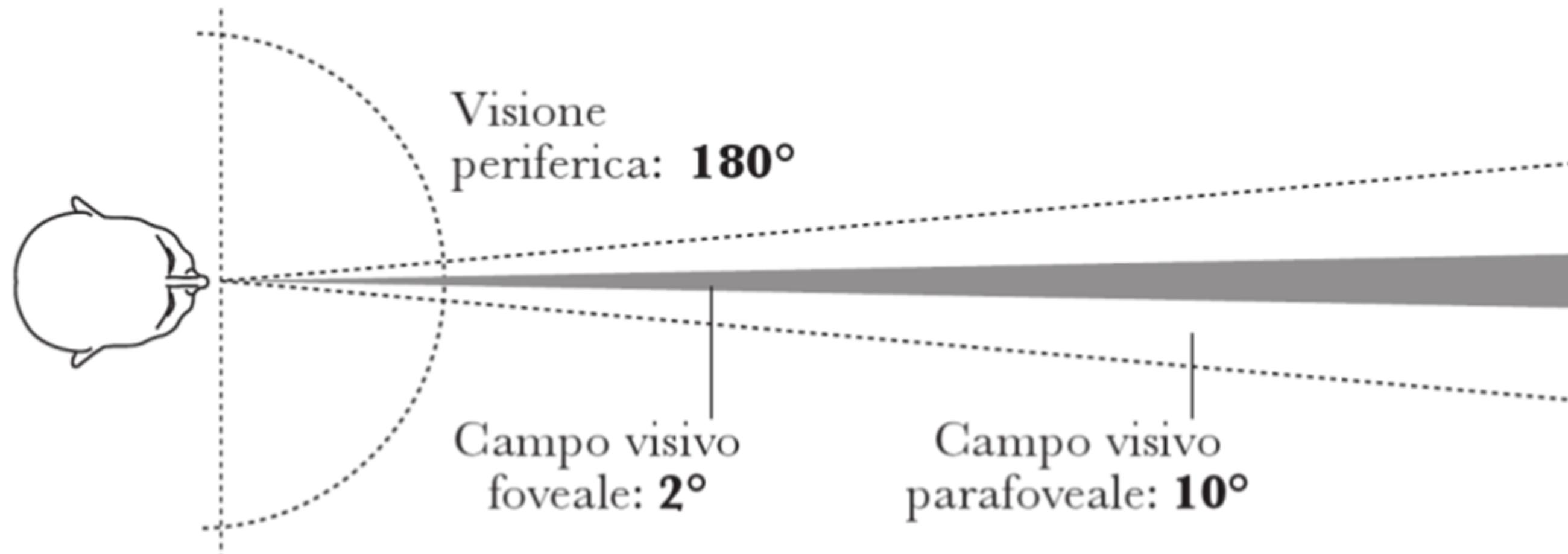
Anatomia dell'Occhio





22

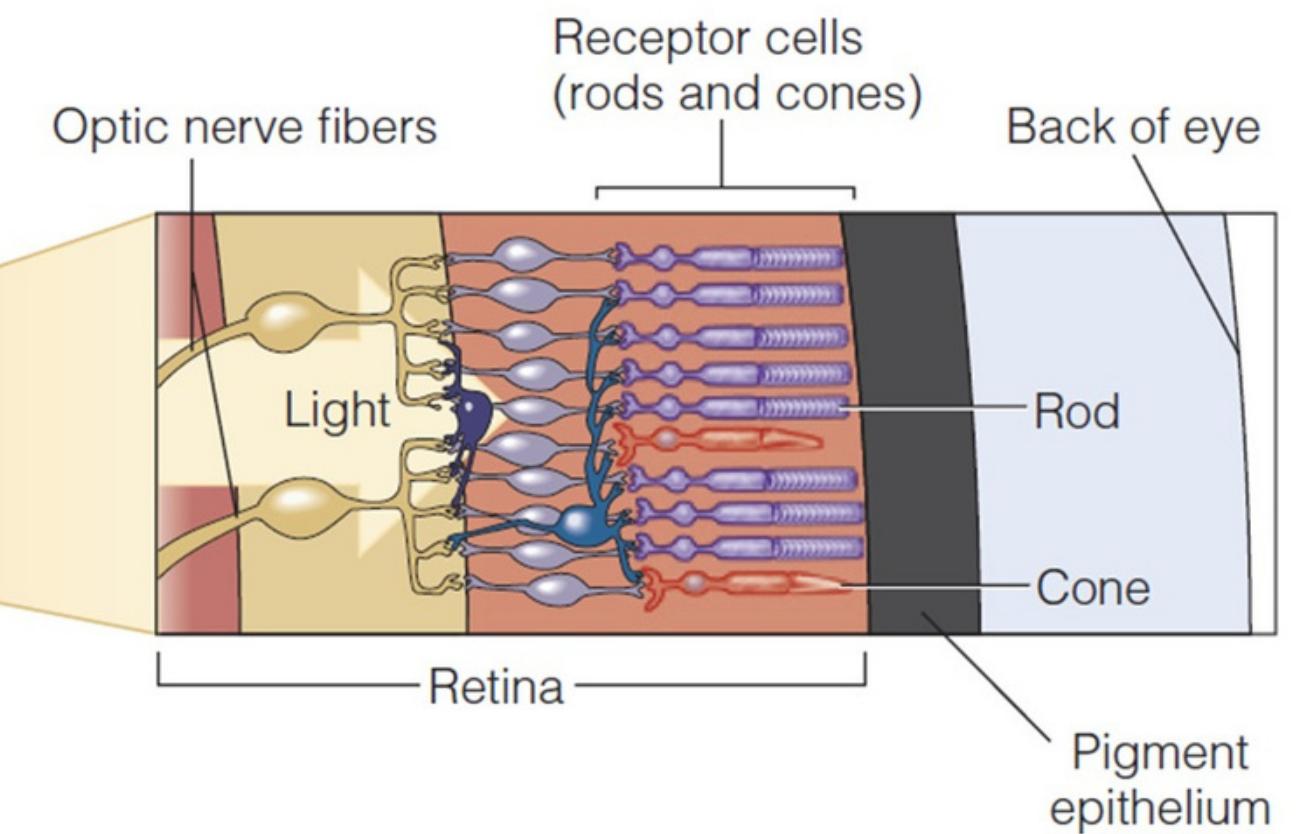
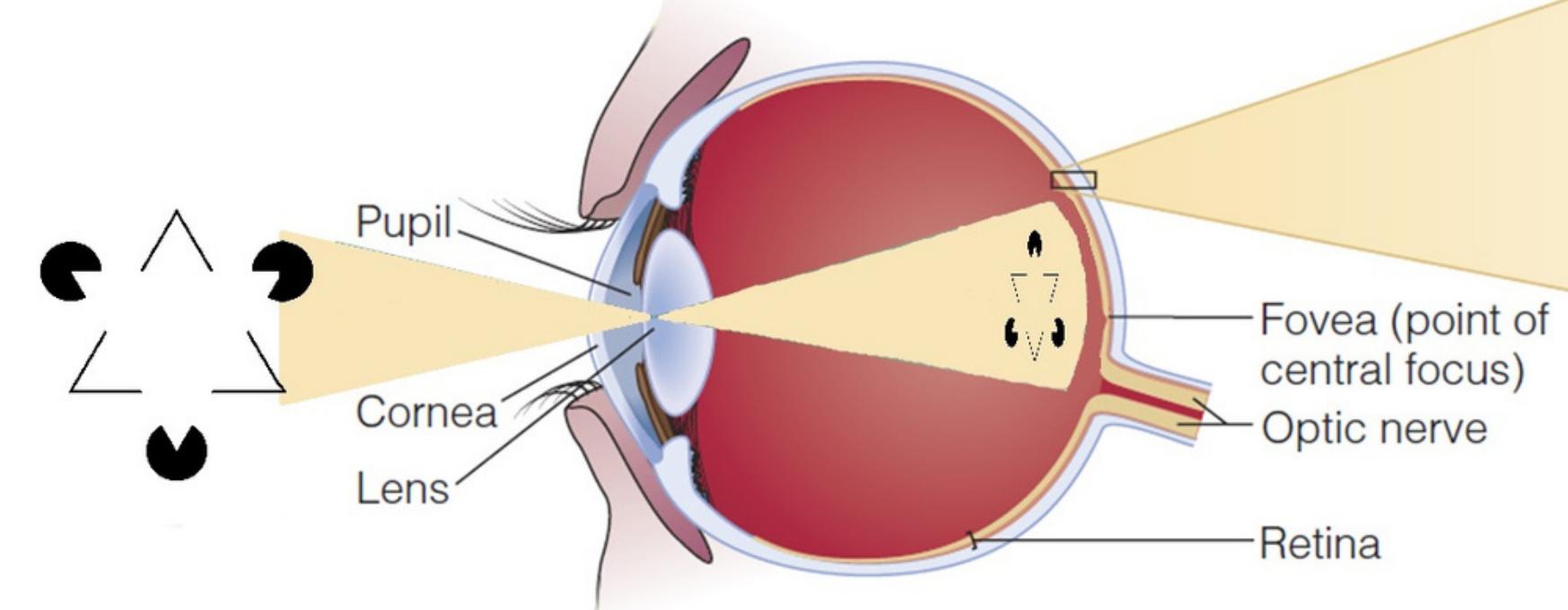
Visione Foveale





23

Anatomia dell'occhio

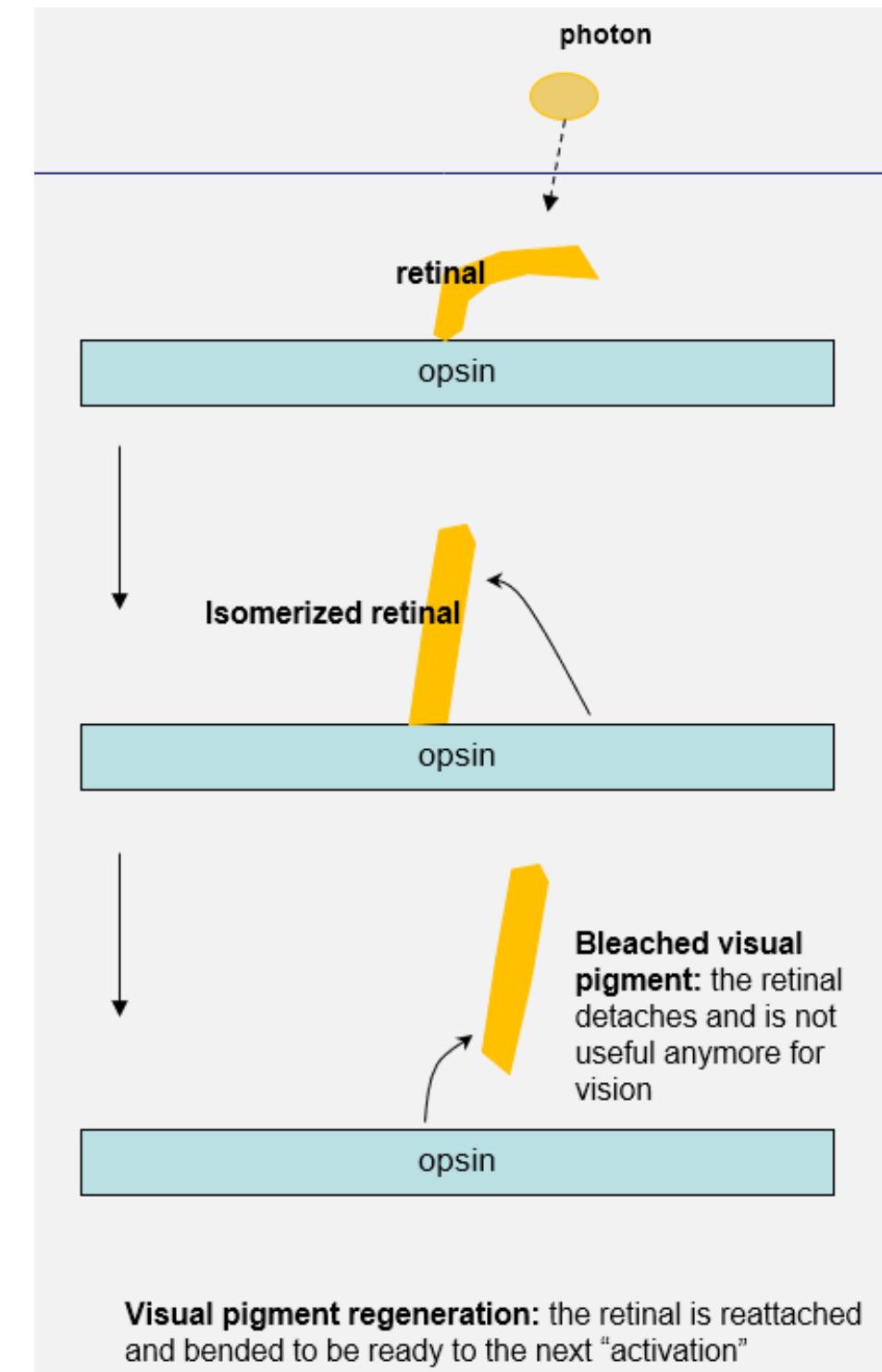




24

Trasduzione Visiva

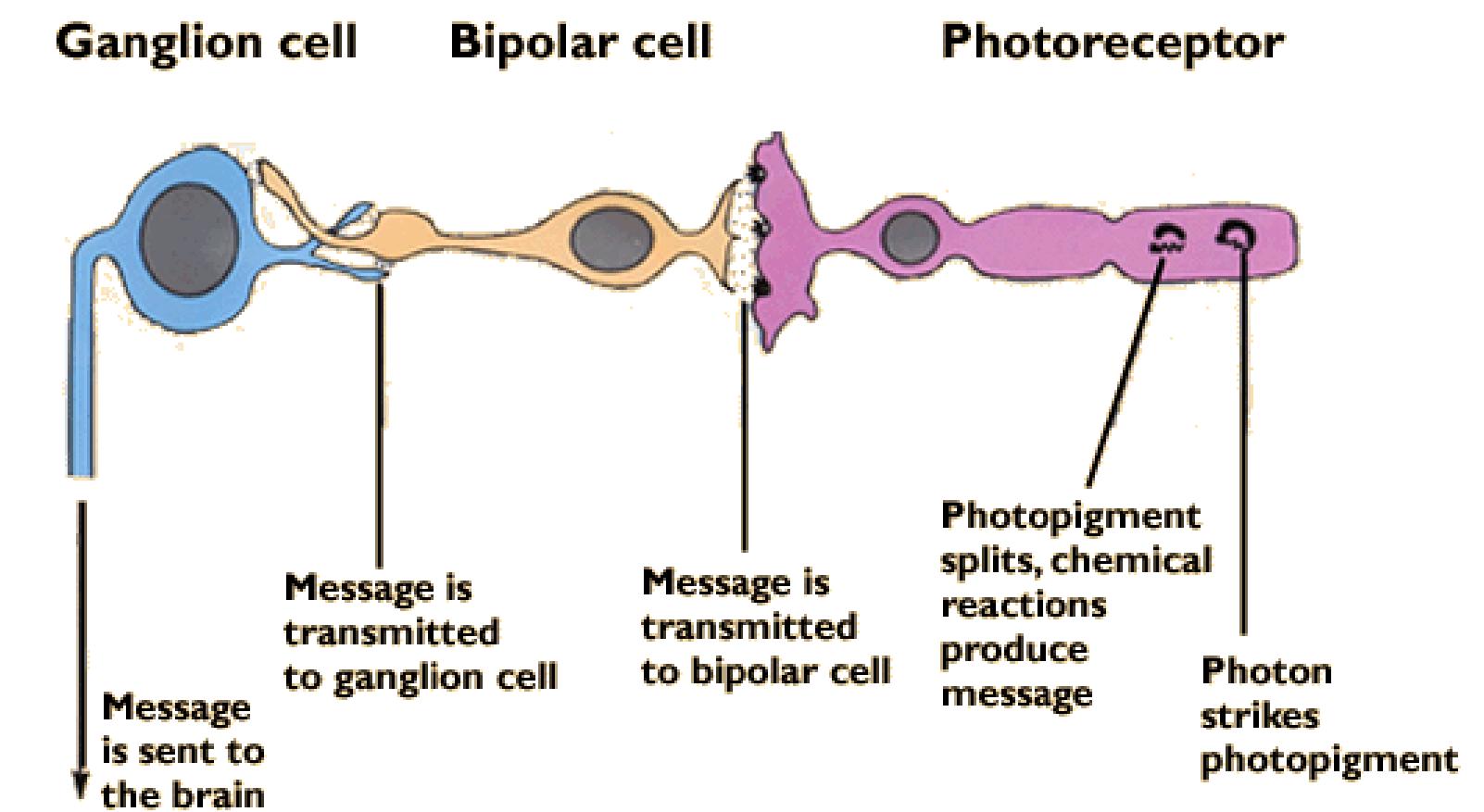
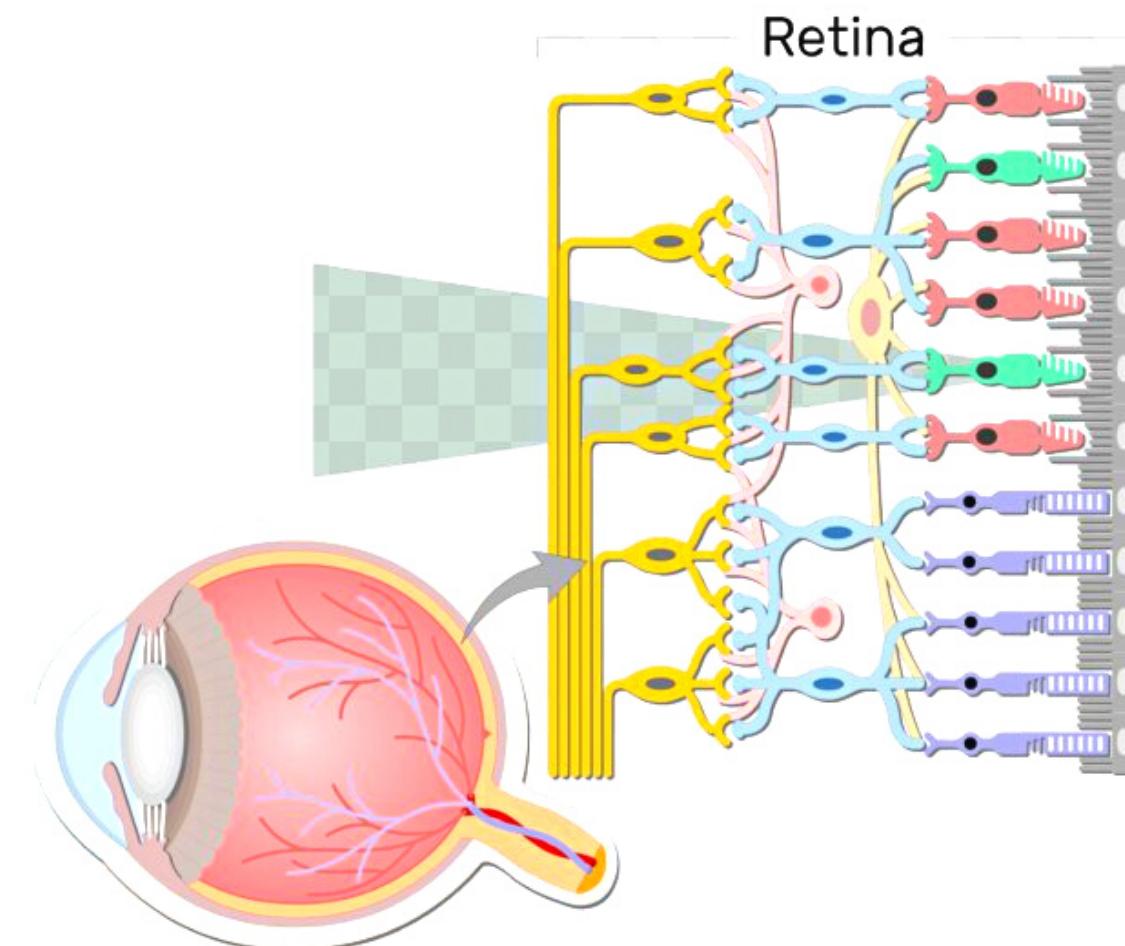
I ricettori processano lo stimolo (luce) trasformandola in elettricità. Questo output viene inviato al nervo ottico che lo porta al cervello, viaggiando in un network di neuroni. Ma come avviene questa operazione di trasformazione della luce (onde e fotoni)?





25

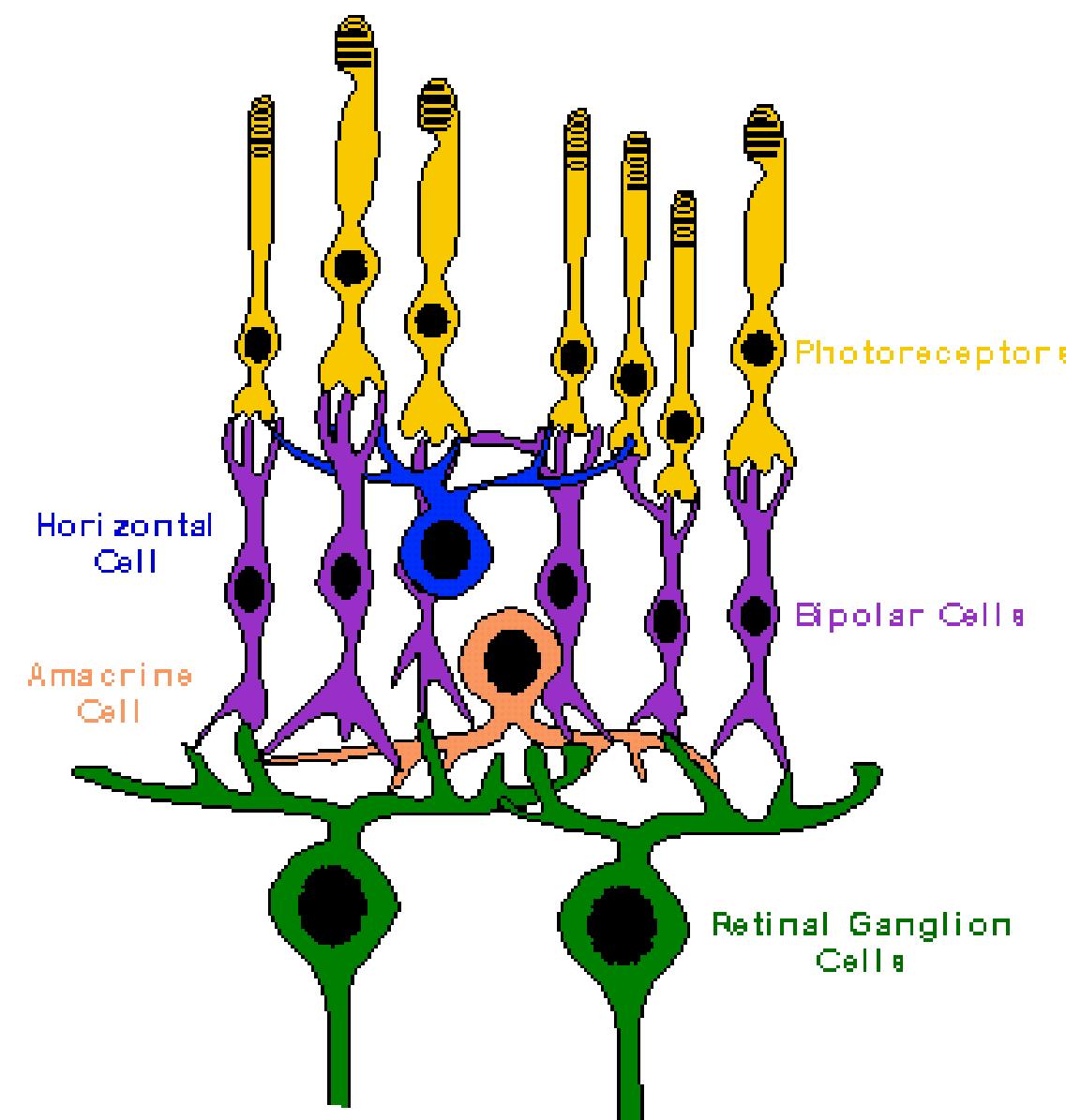
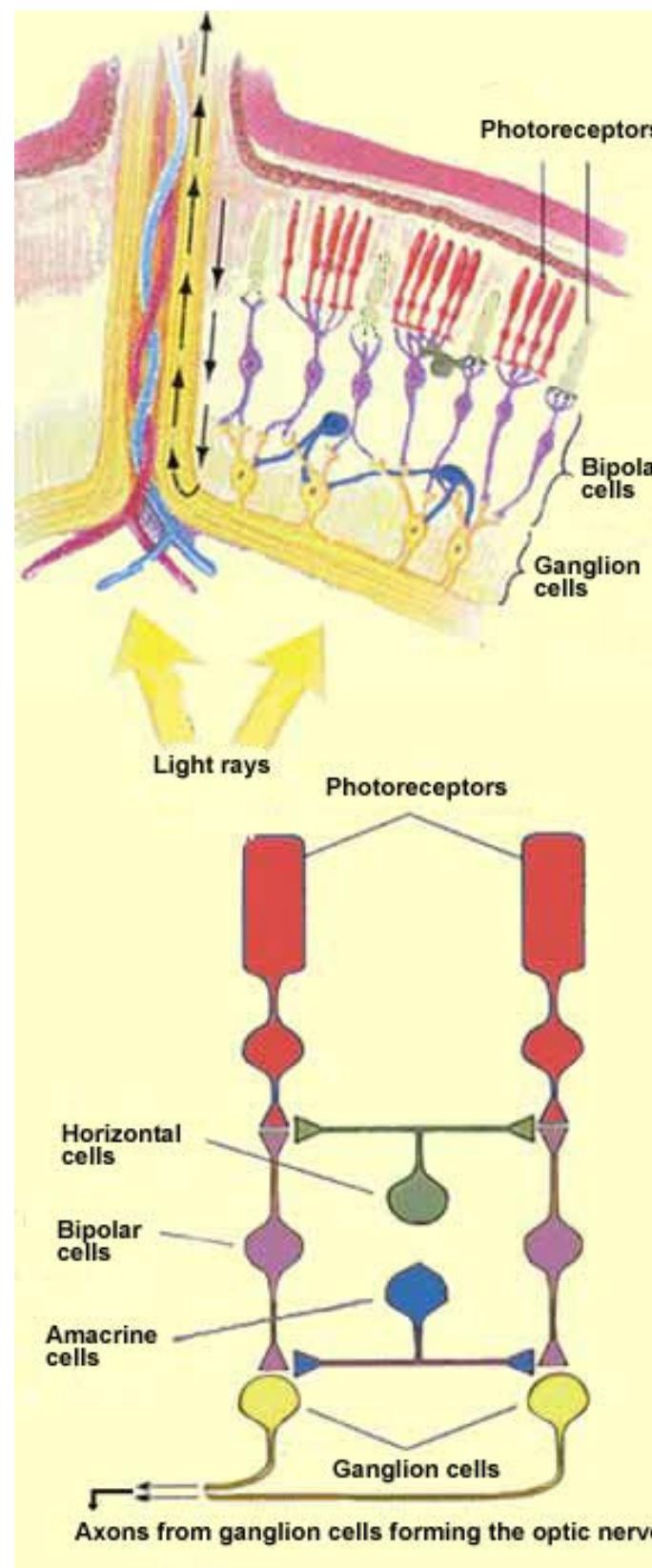
I Layers della Retina





26

I due Layers Intermedi (cellule orizzontali e amacrini)

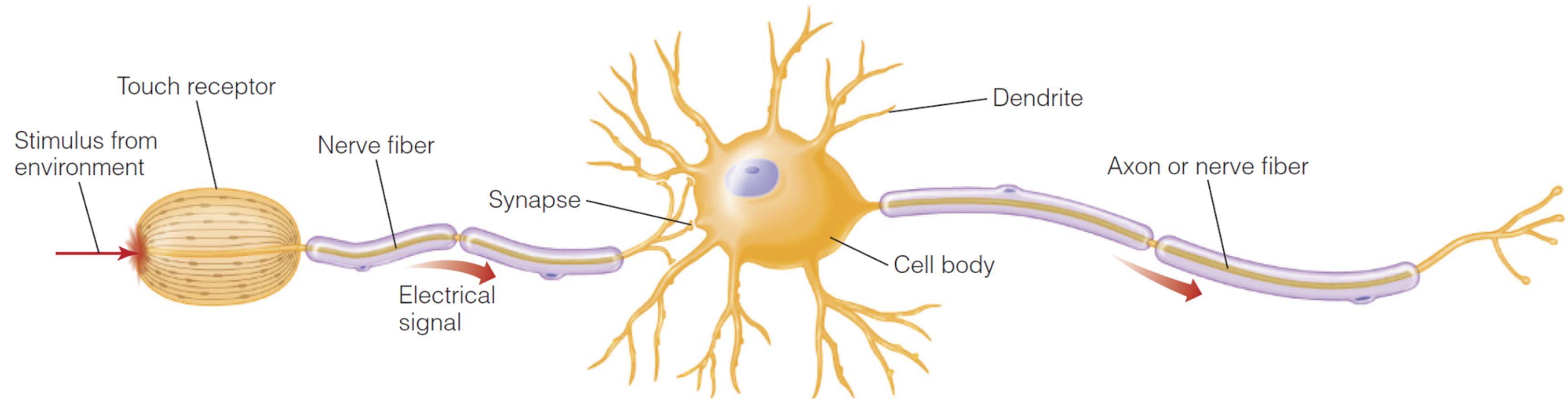


I tre layers della Retina sono separati da due layers intermedi, il cui scopo è fare connessioni fra i neuroni nello stesso layer: le **Cellule Orizzontali**, le quali ricevono informazioni dai fotorecettori e le trasmettono a un numero di neuroni bipolari circondanti, e le **Cellule Amacrini**, le quali ricevono input da cellule bipolari e fanno la stessa cosa alle cellule gangliari)



27

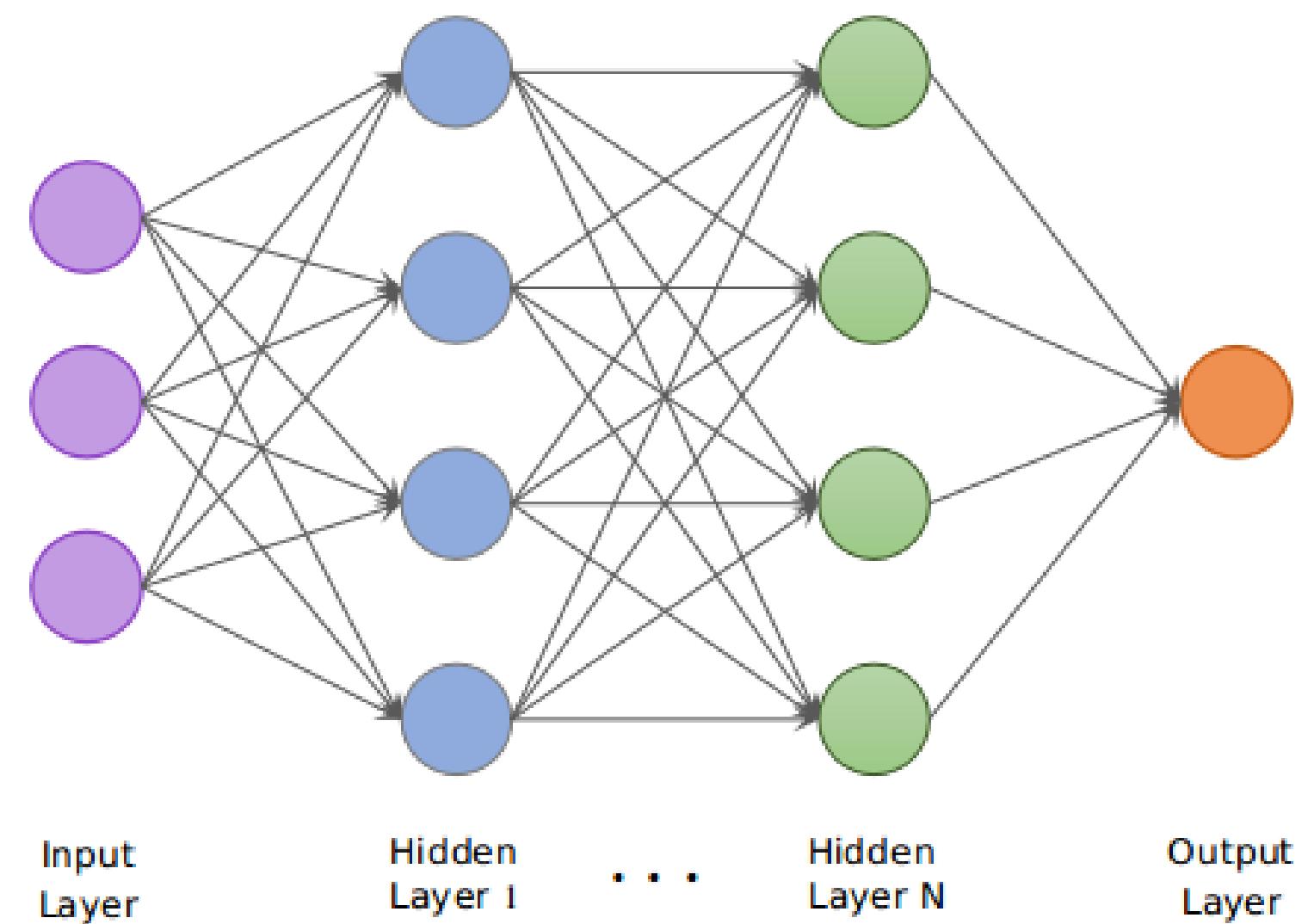
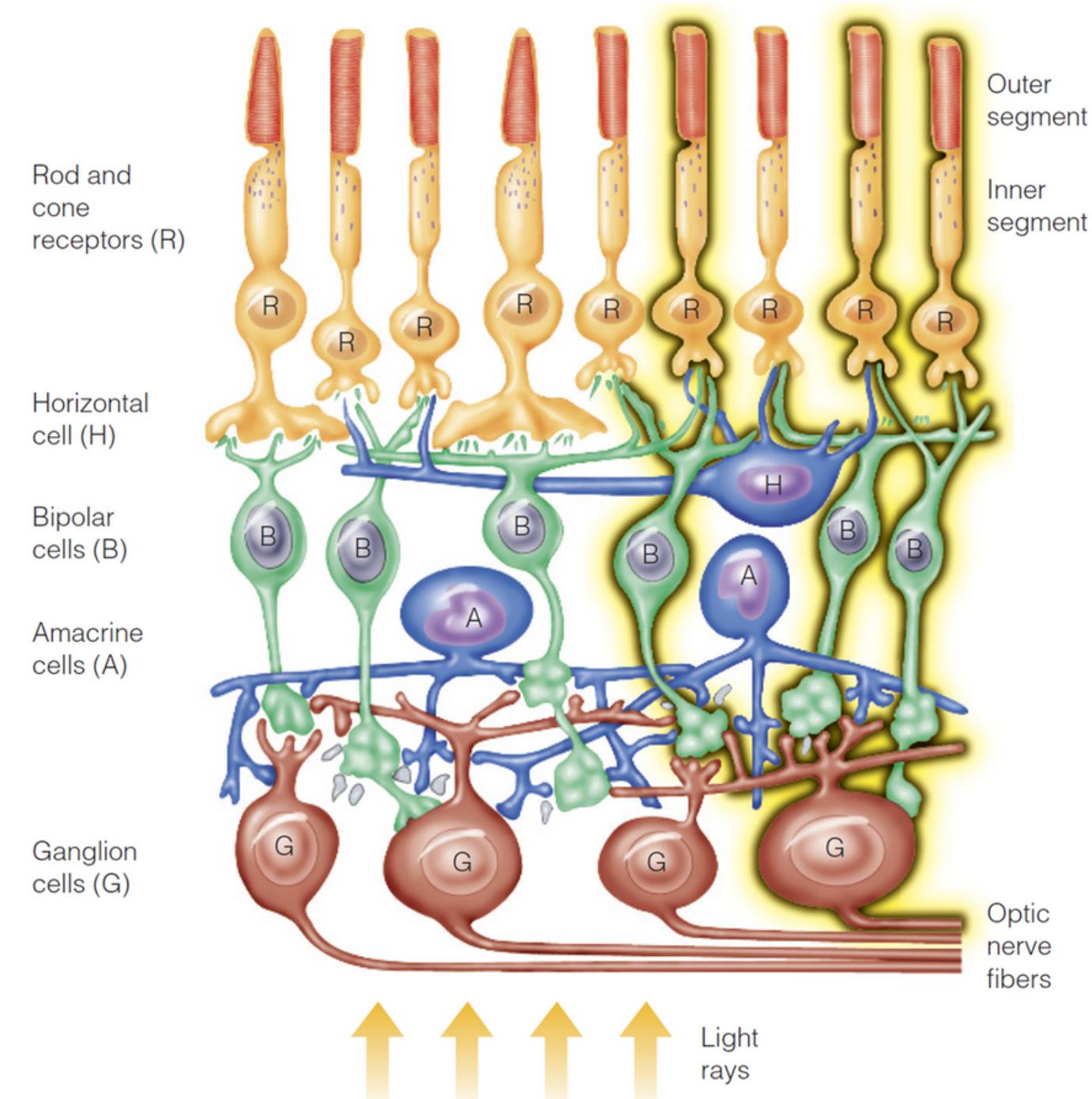
Come sono connessi i Neuroni? Assoni e Dendriti





28

I Layers di Neuroni nella nostra Retina e le CNN



4.

Convolutional Neural Network



30

Cosa fa il nostro programma:

Il programma che abbiamo implementato tenta di emulare ciò che effettua il cervello durante la percezione di una figura anomala.

Esso si compone di 2 parti principali:

- La CNN per emulare la funzione di "memoria" che viene attivata nel cervello
- Canny Edge Detector + Hough Line Transform per ricostruire i bordi dell'illusione



Perchè abbiamo bisogno di una CNN?

La CNN ci serve principalmente per:

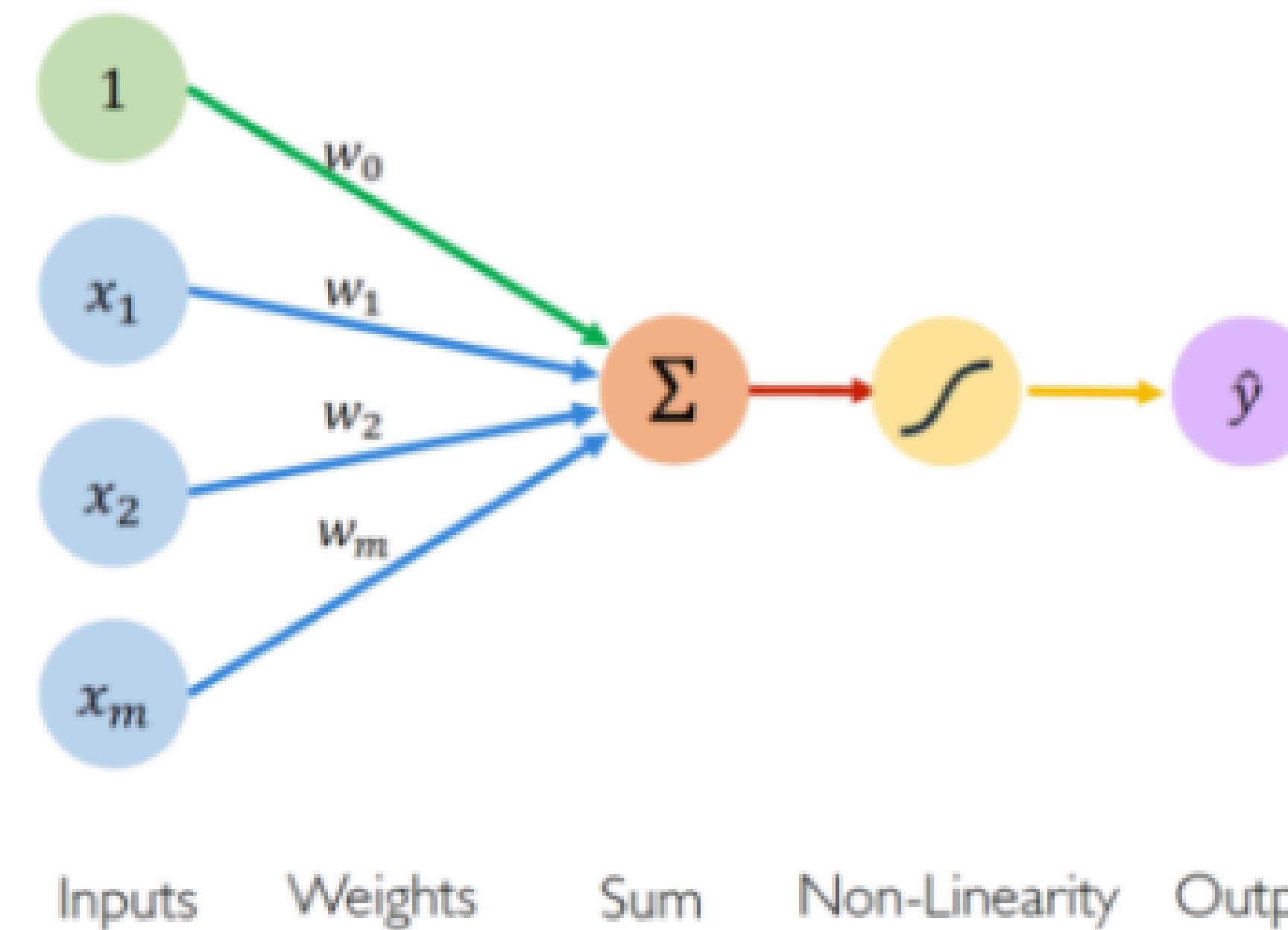
- Risolvere il problema dell'Object Recognition: non è risolvibile con un semplice matching di immagini perchè il template è casuale ogni volta
- Effettuare un training su dataset specifico per aumentare sensibilmente la precisione della predizione da parte del programma.



32

Il percettrone

Si ispira alla sua controparte biologica ossia il neurone

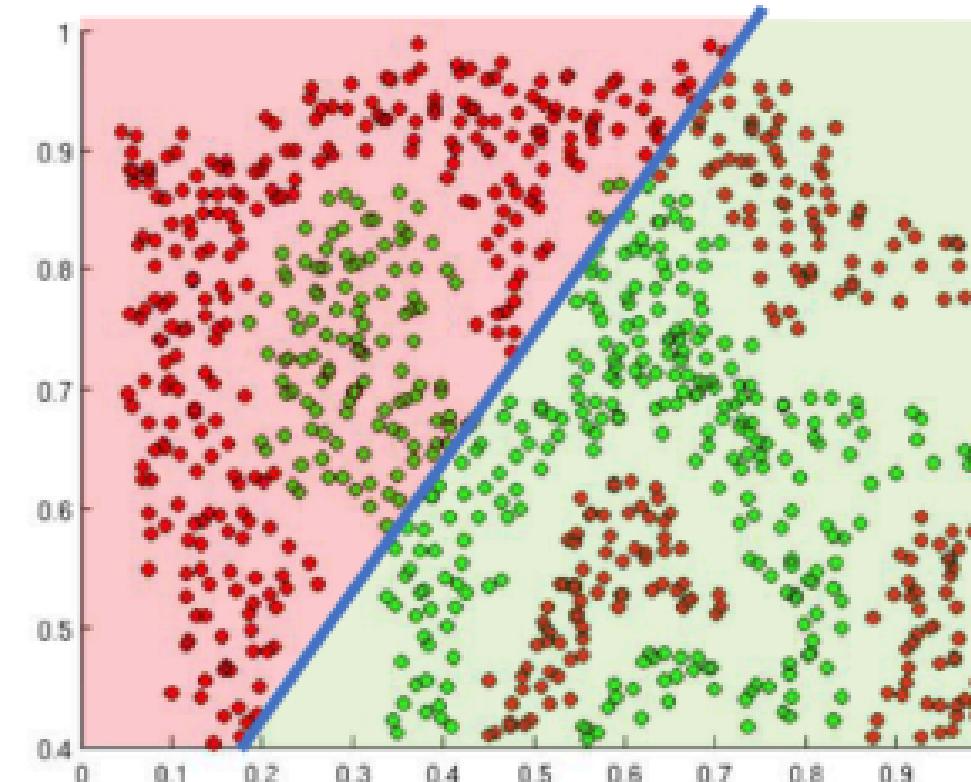




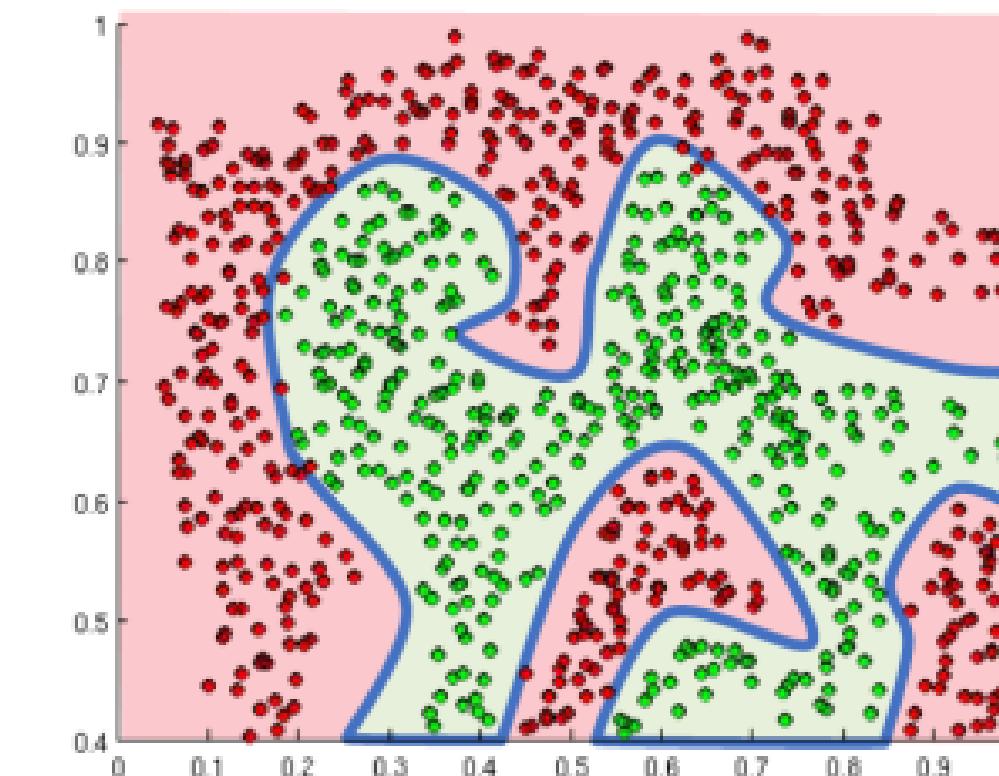
33

L'importanza delle funzioni non lineari

Introducono la non-linearità nella rete neurale permettendo la simulazione di decisioni complesse da parte dei percetroni



Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

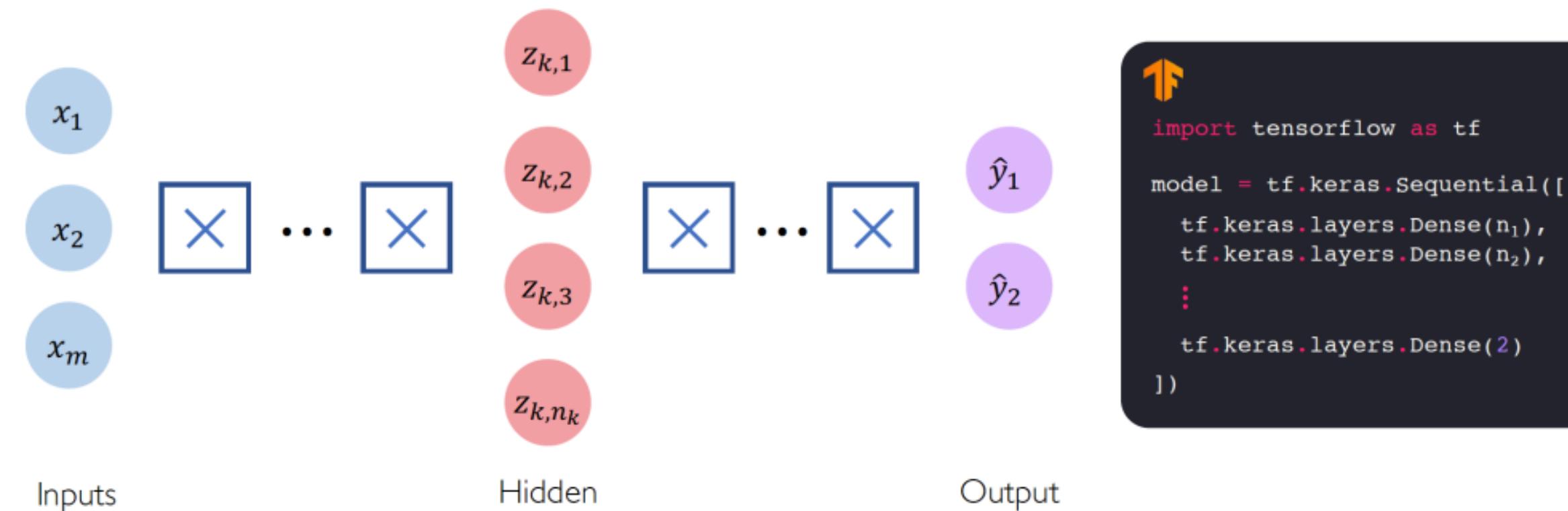


34

La Neural Network

Viene costruita con vari strati di percettroni collegati tra di loro.

Ogni X rappresenta le connessioni di un Layer denso ossia ogni percettore del layer precedente è connesso ai percettroni del layer successivo.





Le pratiche di training

Per effettuare un training del modello abbiamo bisogno di:

- **Training Set**: set di immagini nell'ordine di centinaia o (meglio) migliaia, divise per classi.

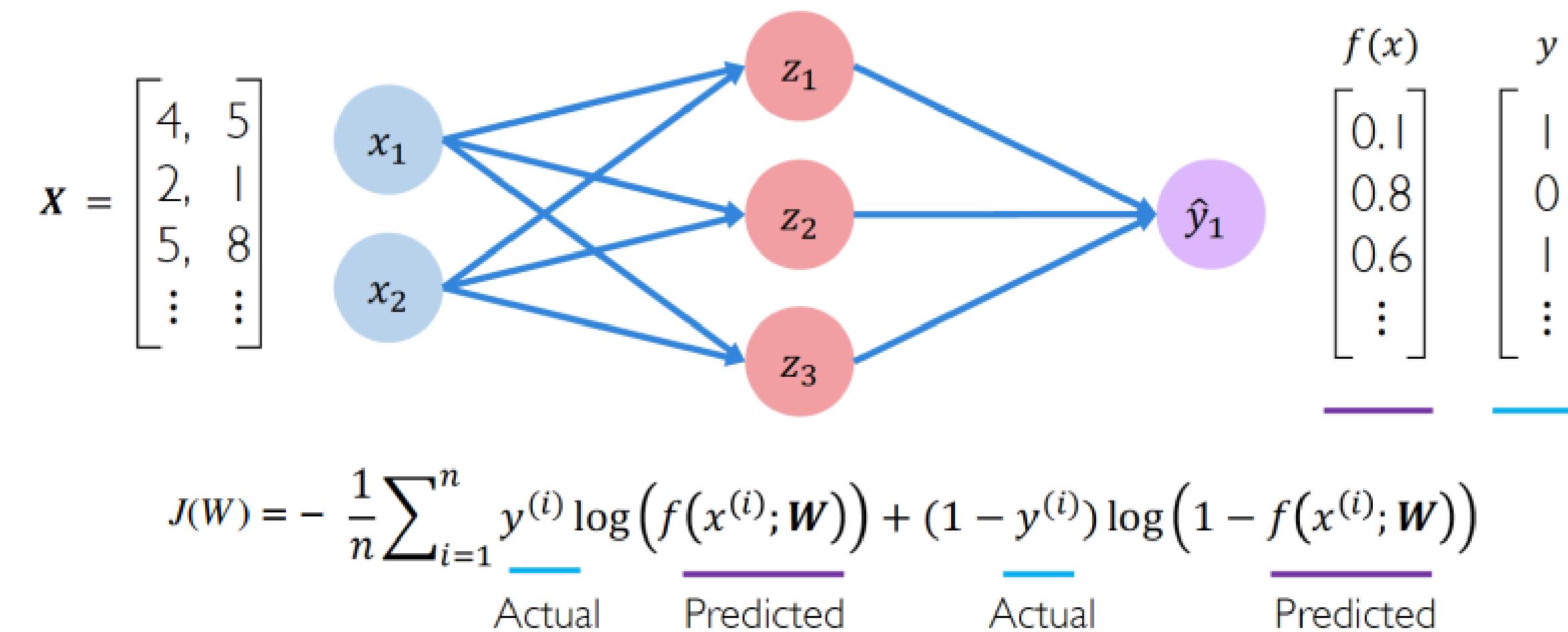
$$T = \{(\mathbf{x}^{(i)}, y^{(i)})\}, \quad i = 1, \dots, N, \quad \mathbf{x}^{(i)} : \text{features}; y^{(i)} : \text{labels}$$

- **Parametro di LOSS**: funzione che descrive i pesi delle connessioni nella rete neurale e ci permette di quantificare le predizioni sbagliate
- **Gradiente di Discesa**: algoritmo che permette di correggere i pesi della rete neurale in seguito ad una predizione



Le pratiche di training

Nel nostro caso specifico ci serviamo della Cross Entropy Loss , una funzione che può essere usata con modelli che danno in output una probabilità tra 0 e 1



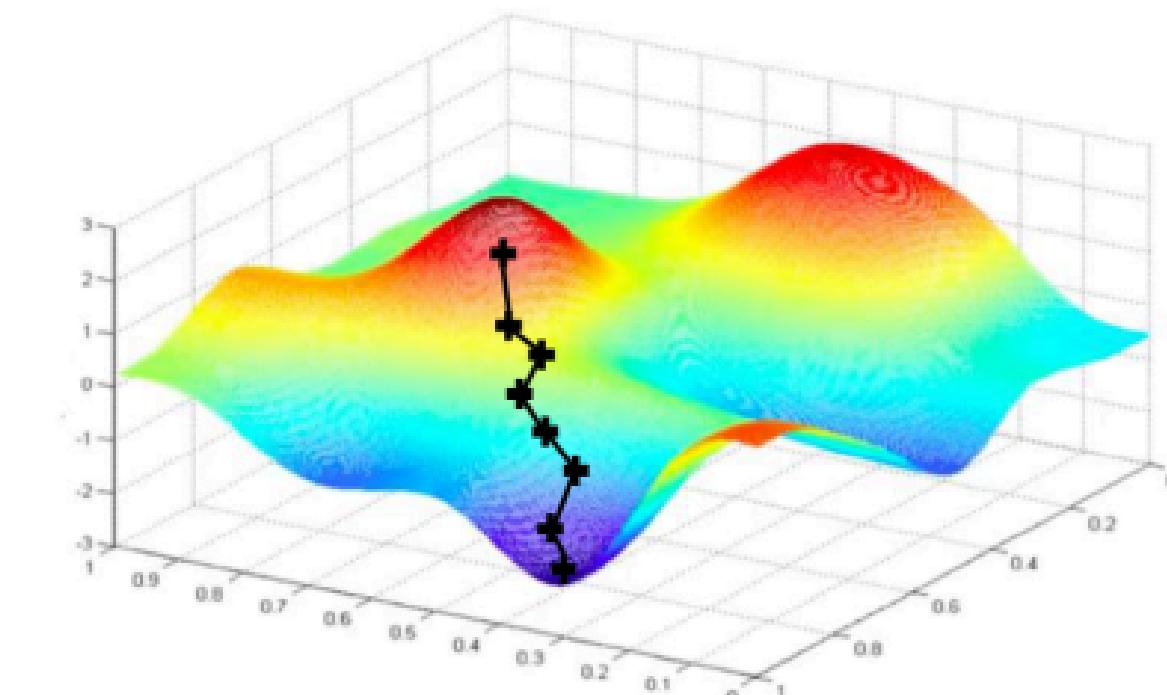


Le pratiche di training

Per ottimizzare la funzione di LOSS ci serviamo del gradiente di discesa

Algorithm

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
4. Update weights, $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$
5. Return weights

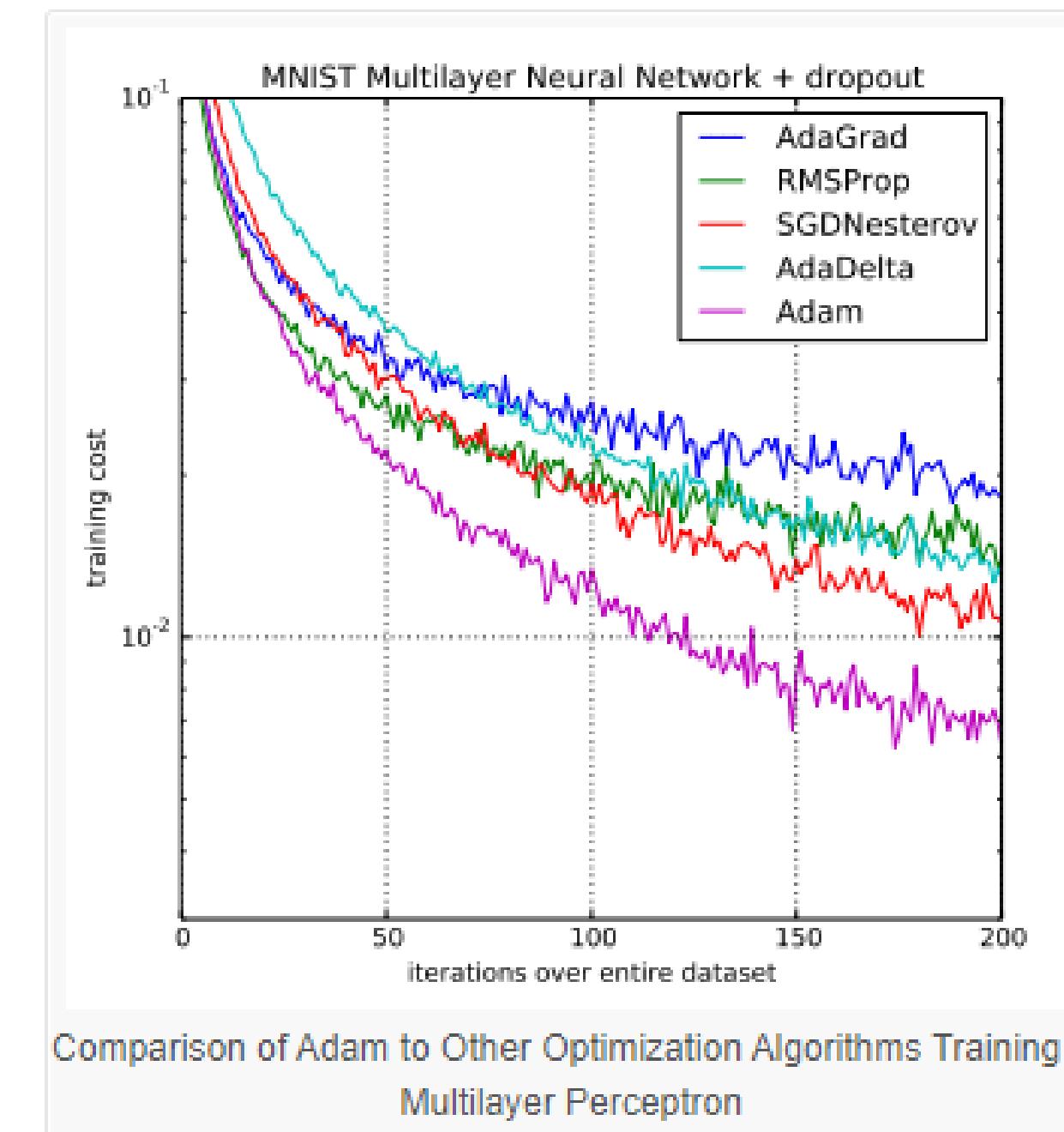




38

Le pratiche di training

Problema: il gradiente di discesa da solo non basta, in quanto è difficile da ottimizzare. Dobbiamo settare un Learning Rate adatto → [Adam](#)





Le pratiche di training

Batch vs mini-batch

Il dataset non viene passato tutto insieme come un'unica batch ma viene suddiviso in mini-batch più piccole in quanto permettono:

- Maggiore accuratezza per la stima del gradiente (tramite una convergenza dei pesi più “regolare” e learning rates maggiori)
- Un training più veloce dato che la computazione delle mini-batch viene parallelizzata dalla GPU



Ci basta?

Quasi, la rete appena descritta è una **Feed Forward Neural Network**, funziona bene per immagini semplici ma inizia a performare male nel caso di immagini complesse.

Ci serve una **CNN**, per ottenerla aggiungiamo:

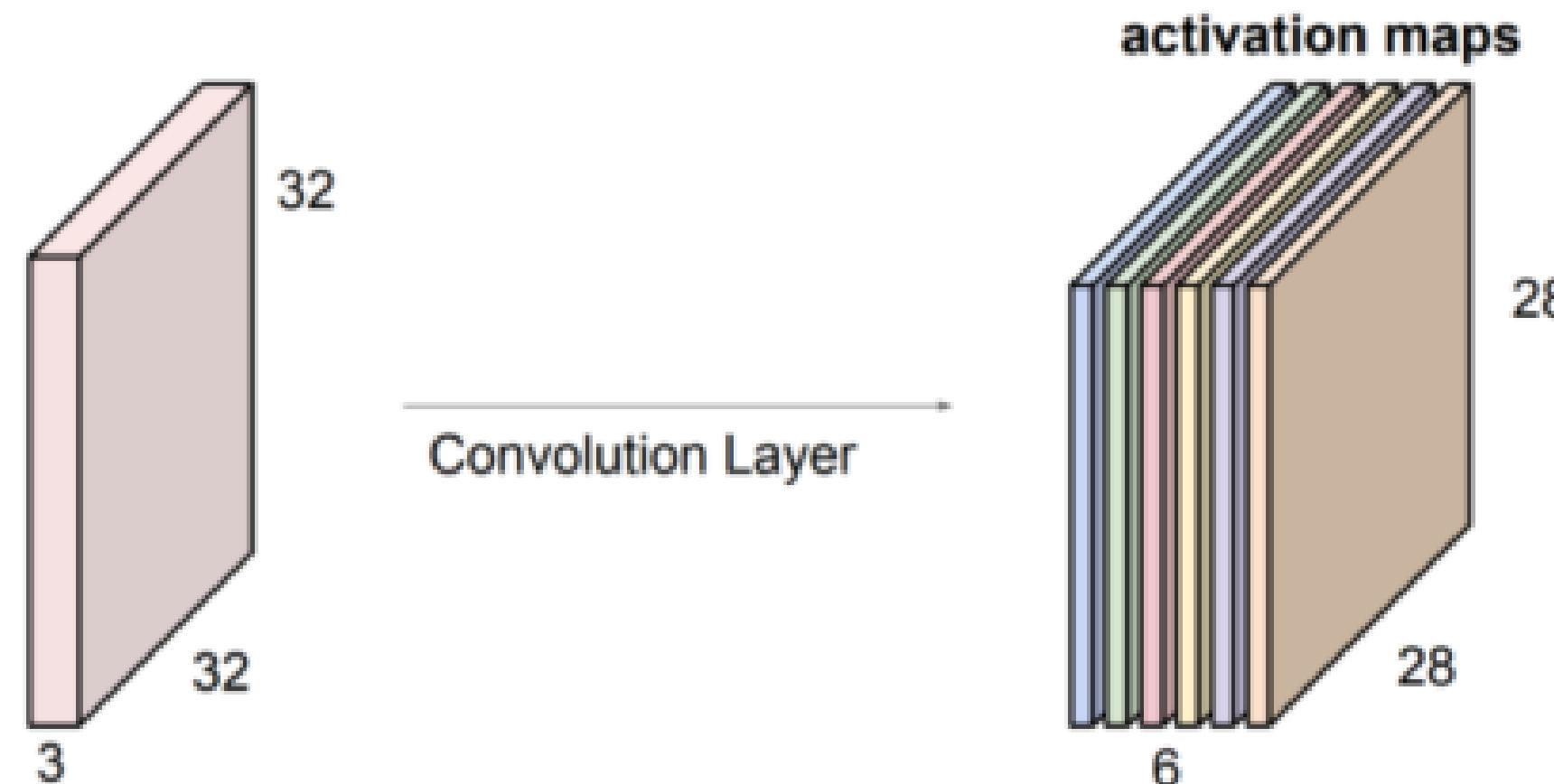
- **Convolutional layer**: ha come obiettivo l'estrazione di features d'alto livello (Bordi, colori, spigoli, etc.) e permette alla rete una miglior comprensione dell'immagine
- **Operazione di Pooling**: riduce la dimensionalità delle features estratte permettendo così una maggiore efficienza a livello computazionale e di evidenziare feature dominanti che sono spazio-invarianti



41

Convolutional layer

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



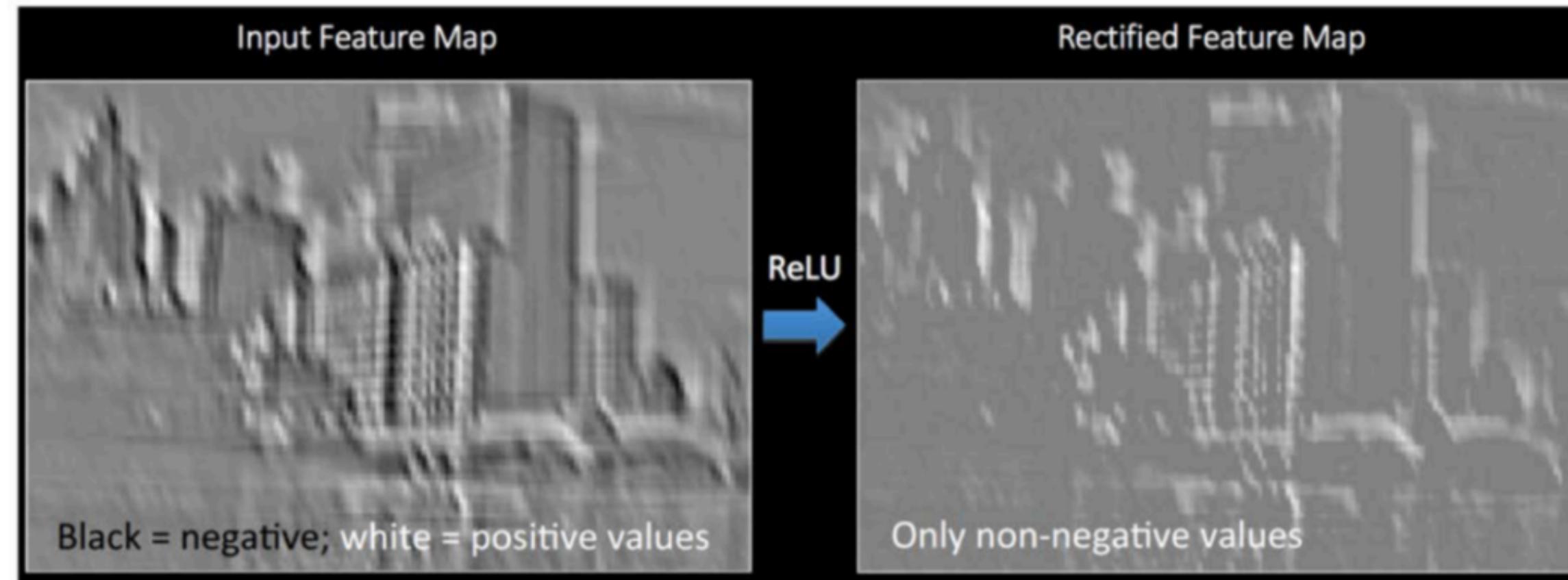
We stack these up to get a “new image” of size 28x28x6!



42

Conv Output

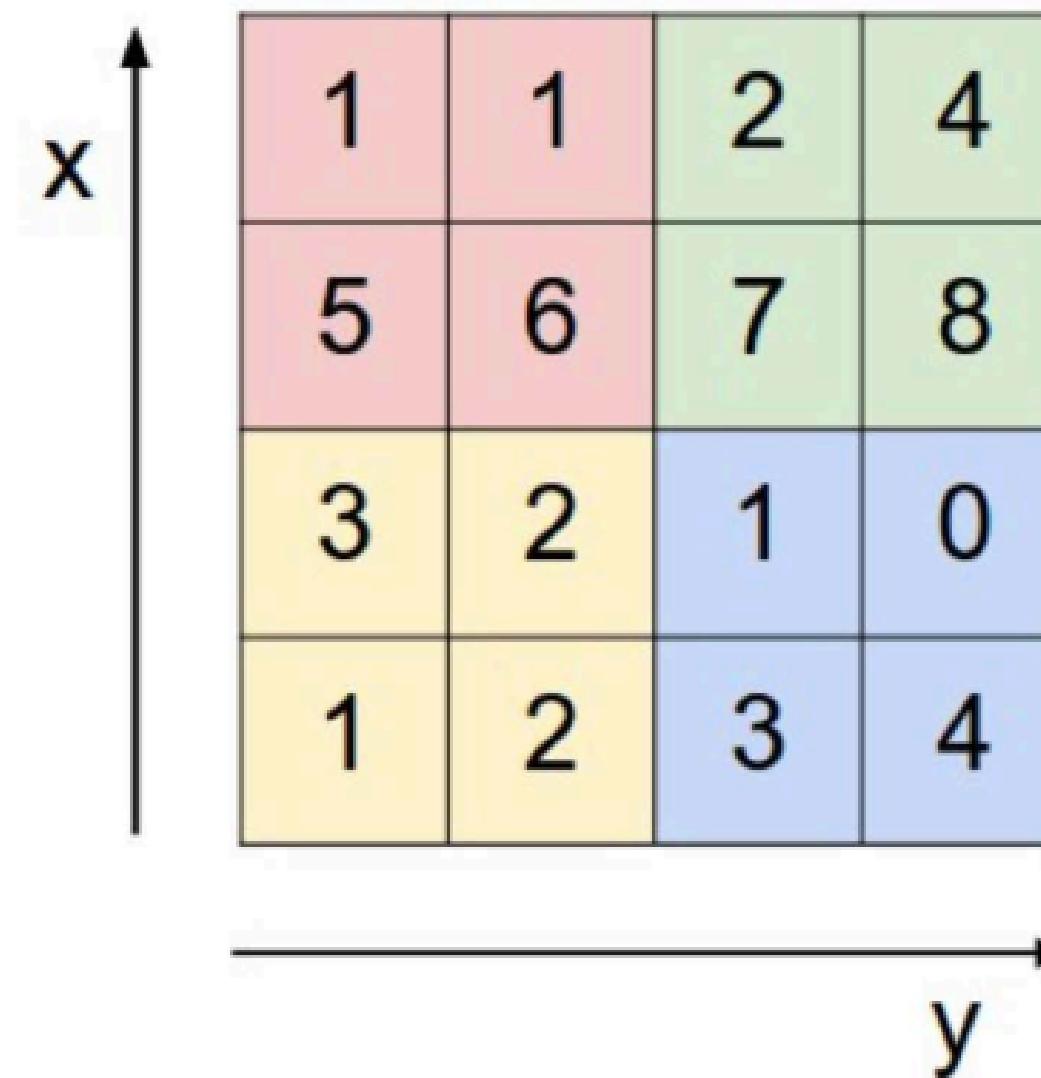
- Apply after every convolution operation (i.e., after convolutional layers)
- ReLU: pixel-by-pixel operation that replaces all negative values by zero. **Non-linear operation!**





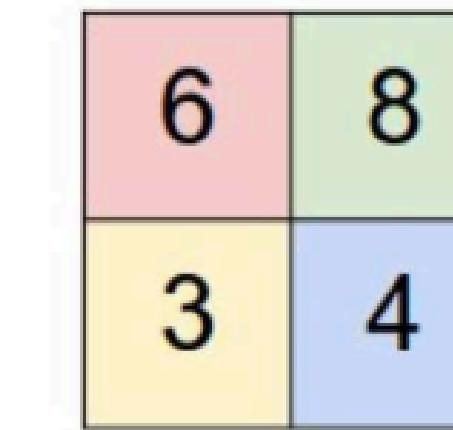
43

Pooling



max pool with 2x2 filters
and stride 2

```
tf.keras.layers.MaxPool2D(  
    pool_size=(2,2),  
    strides=2  
)
```

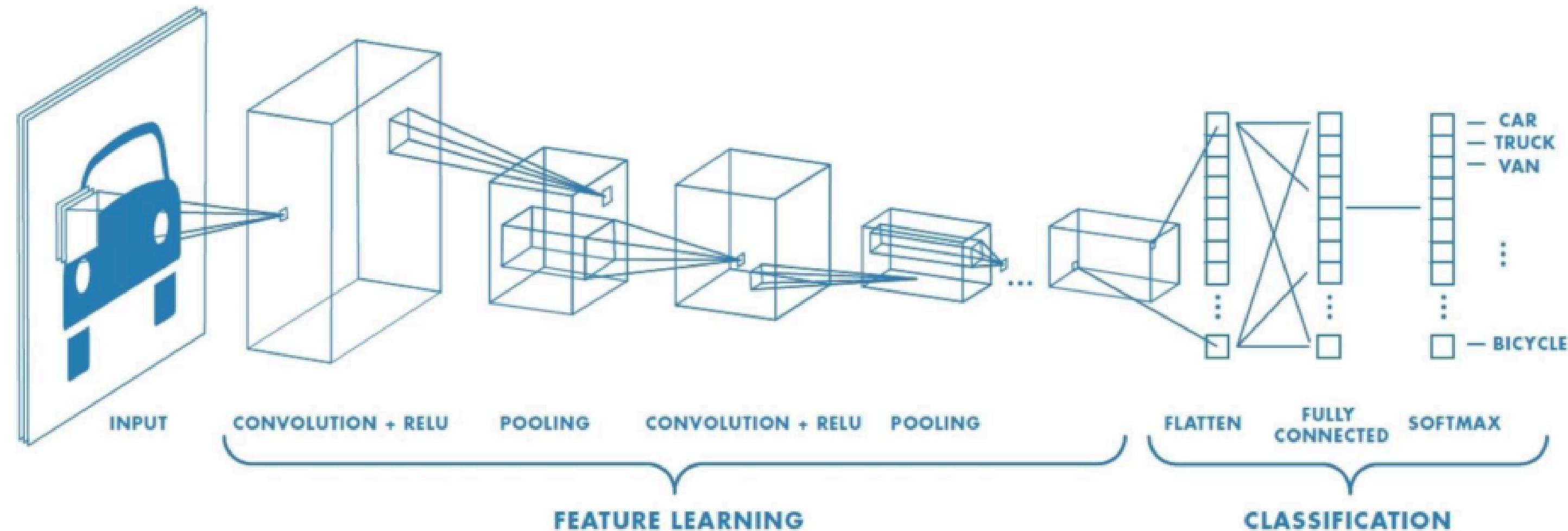


- 1) Reduced dimensionality
- 2) Spatial invariance



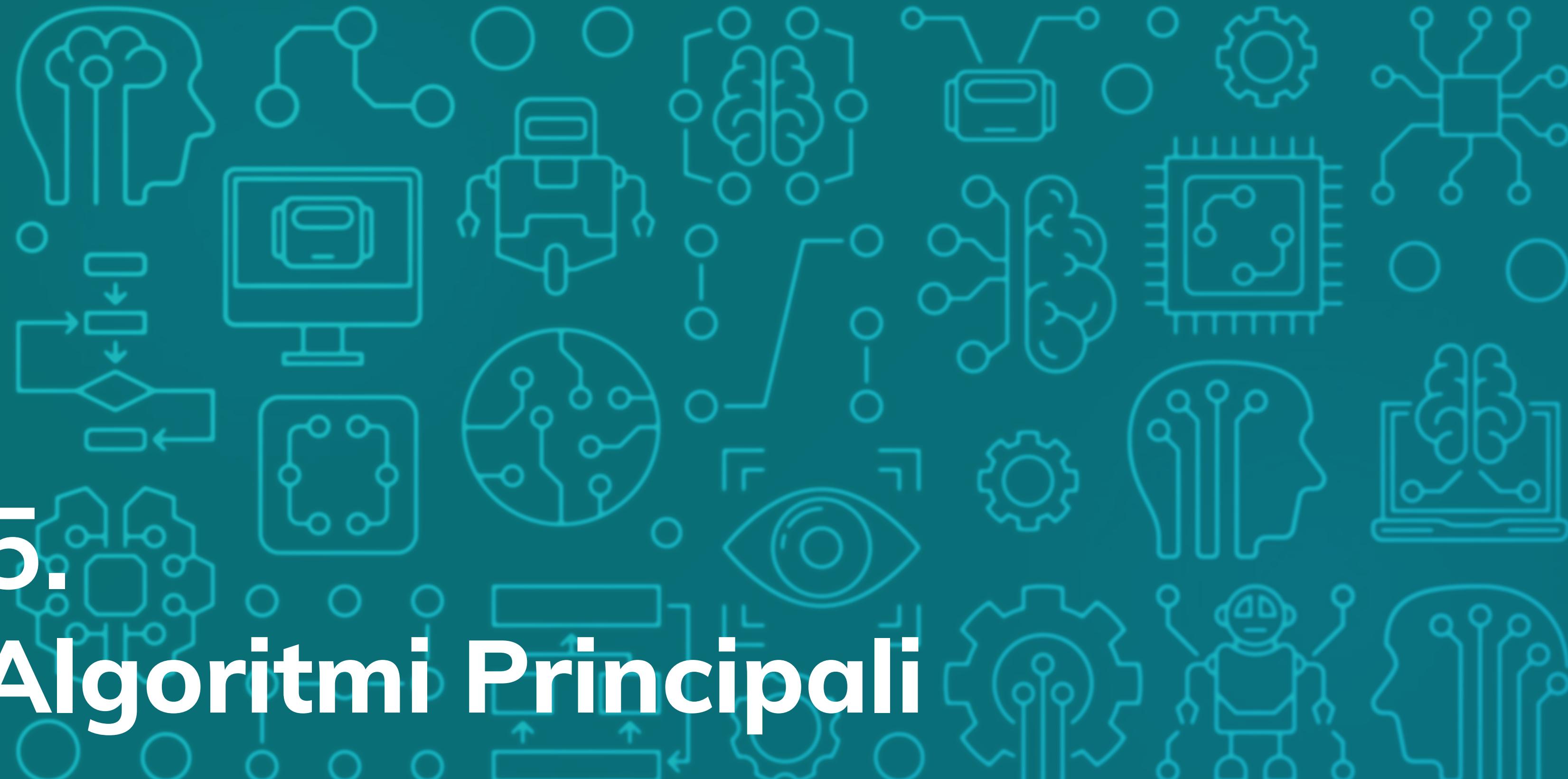
44

Struttura finale



5.

Algoritmi Principali





46

Canny Edge Detection

Algoritmo utile a mostrare i bordi di qualsiasi oggetto presente in un'immagine.

input: Immagine gray scale (o binaria)

E' composto da 5 passaggi che sono:

- Riduzione del rumore
- Calcolo del gradiente
- Non-maximum suppression
- Double Threshold
- Edge Tracking by Hysteresi



Original image on the left — Processed image on the right



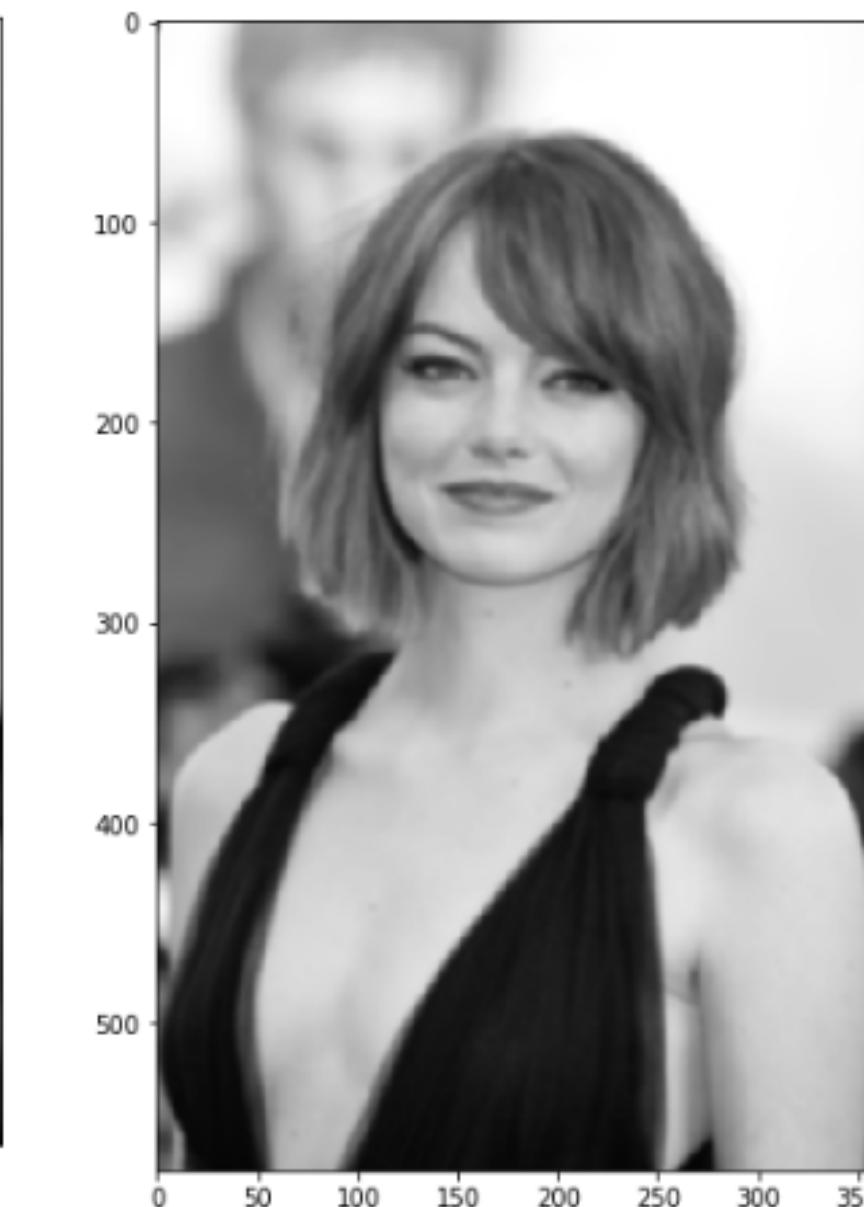
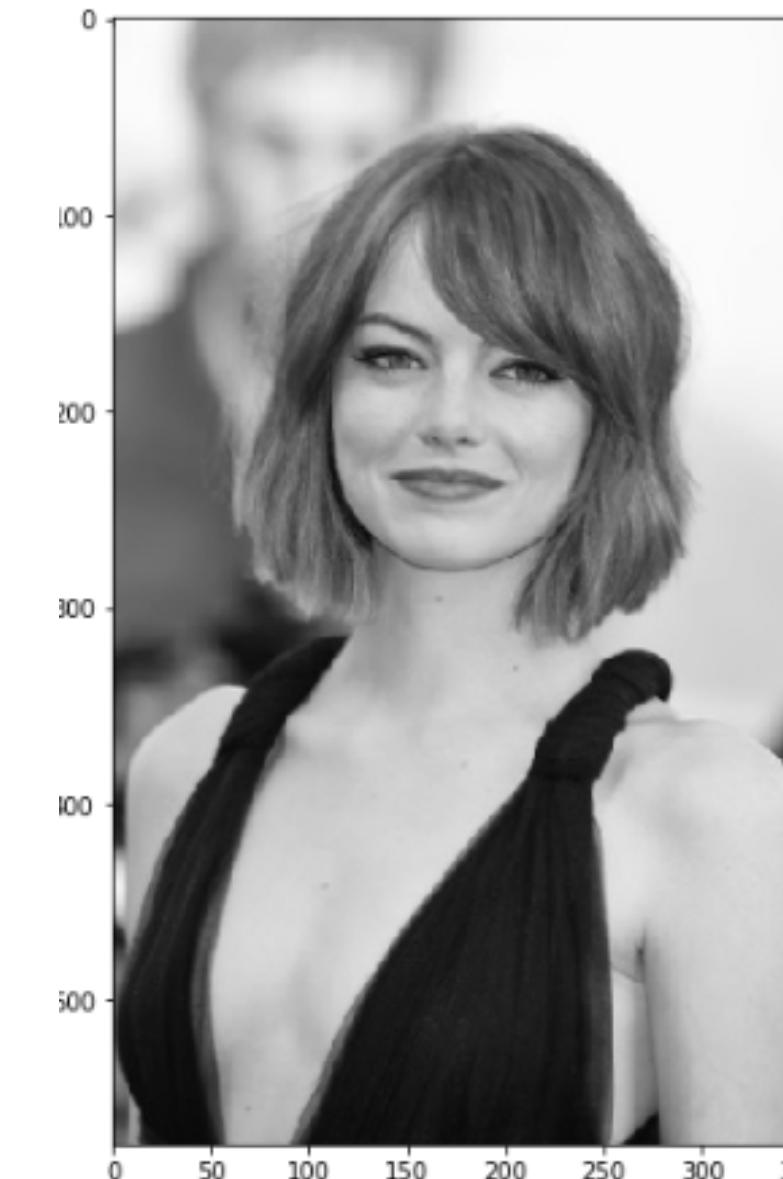
47

Riduzione del rumore

Dato che la matematica involta nel rilevamento dei bordi è molto sensibile al rumore, si applica una Sfocatura Gaussiana per ridurre il rumore.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

Gaussian filter kernel equation





48

Calcolo del Gradiente

Rileva l'intensità del bordo e la rispettiva direzione, calcolando il gradiente dell'immagine.

I bordi corrispondono con un cambio della densità dei pixel

Il modo più semplice per rilevarli è applicare un filtro (Sobel) che evidenzia questo cambio in entrambe le direzioni : orizzontale (x) e verticale (y)

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

Sobel filters for both direction (horizontal and vertical)

$$|G| = \sqrt{I_x^2 + I_y^2},$$

$$\theta(x, y) = \arctan\left(\frac{I_y}{I_x}\right)$$

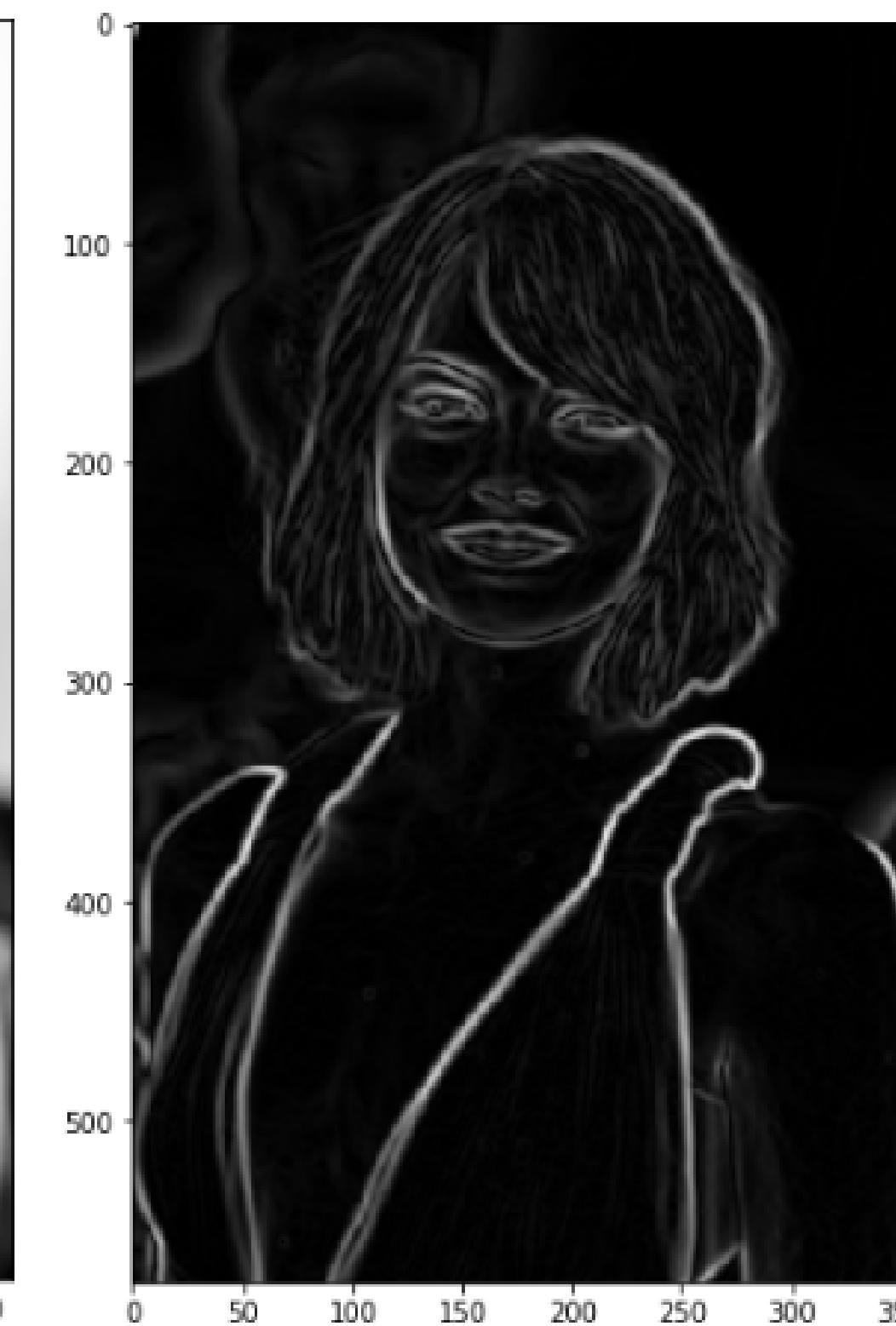
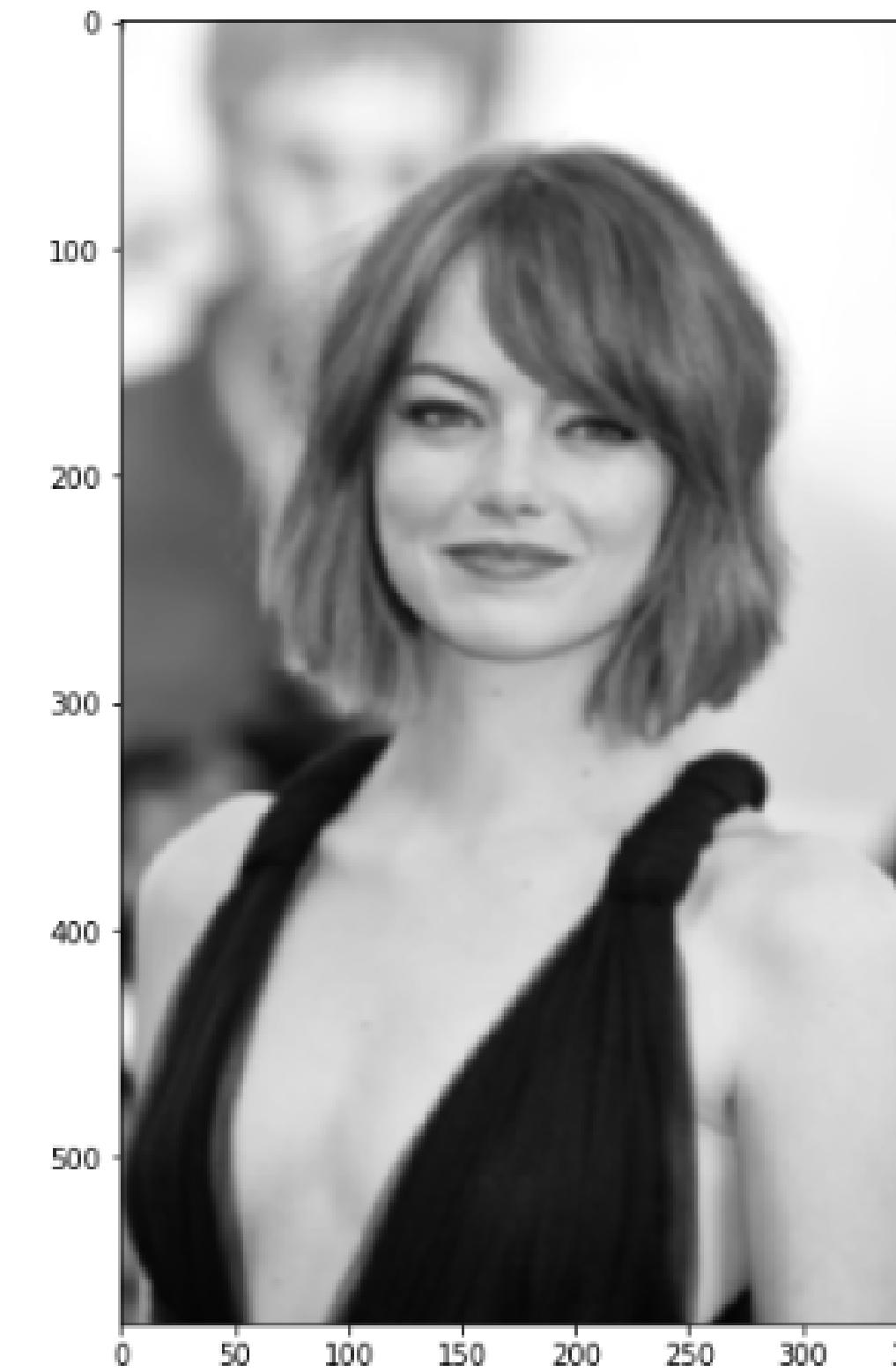
Gradient intensity and Edge direction

Il risultato di questa operazione, è quasi quello aspettato, ma ha due problemi: l'intensità e lo spessore dei bordi non sono uniformi.



49

Calcolo del Gradiente



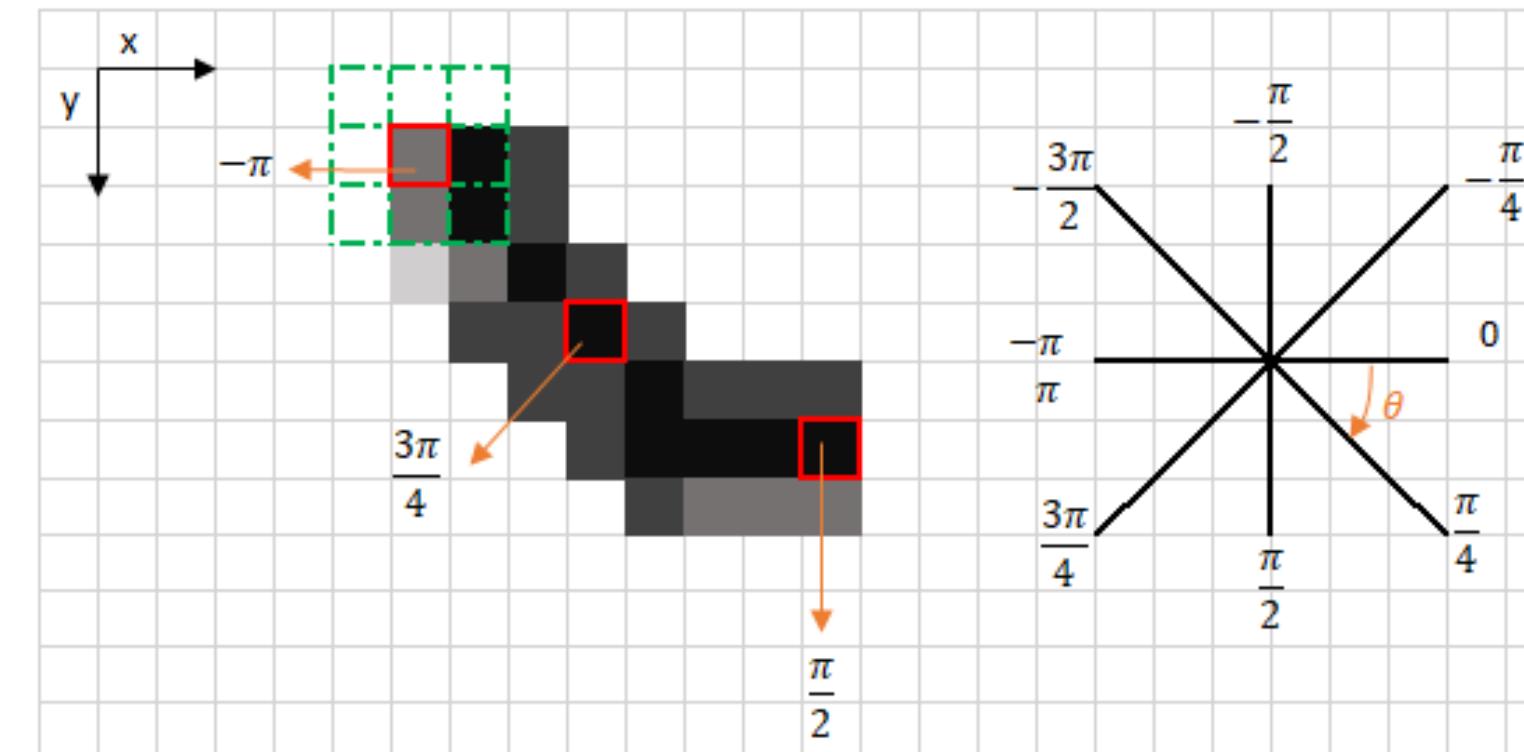


50

Non-Maximum Suppression

Idealmente l'immagine finale dovrebbe avere bordi fini.

L'algoritmo cicla su ogni punto della matrice di intensità (del gradiente) e trova i pixel che hanno il valore massimo nella direzione del bordo.



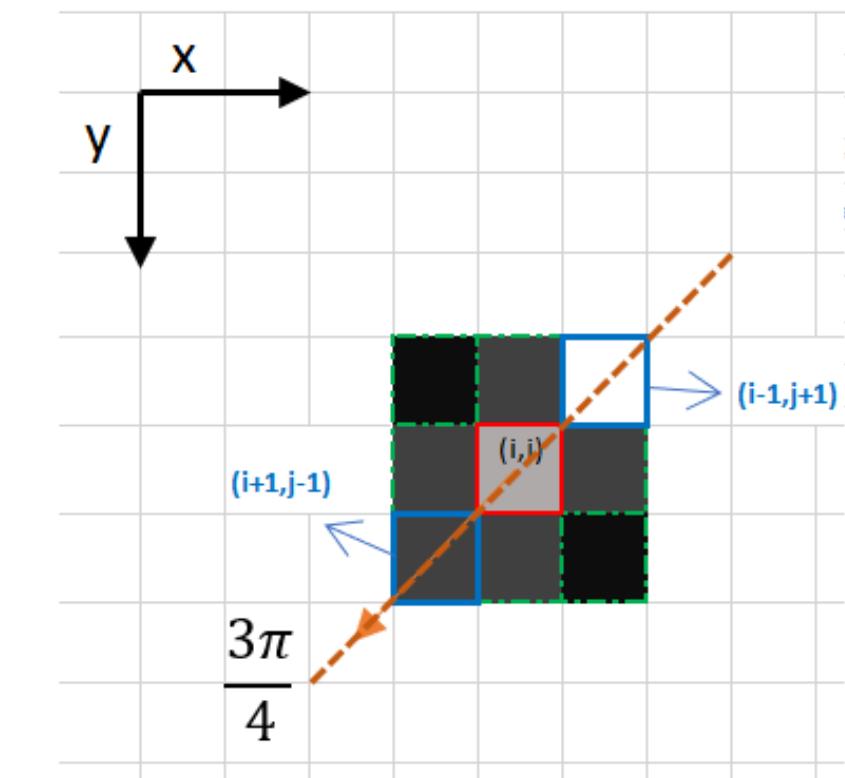


51

Non-Maximum Suppression

In questo caso il pixel che verrà mantenuto nella matrice finale è il pixel $(i-1,j+1)$ perchè nella direzione è il pixel di intensità maggiore.

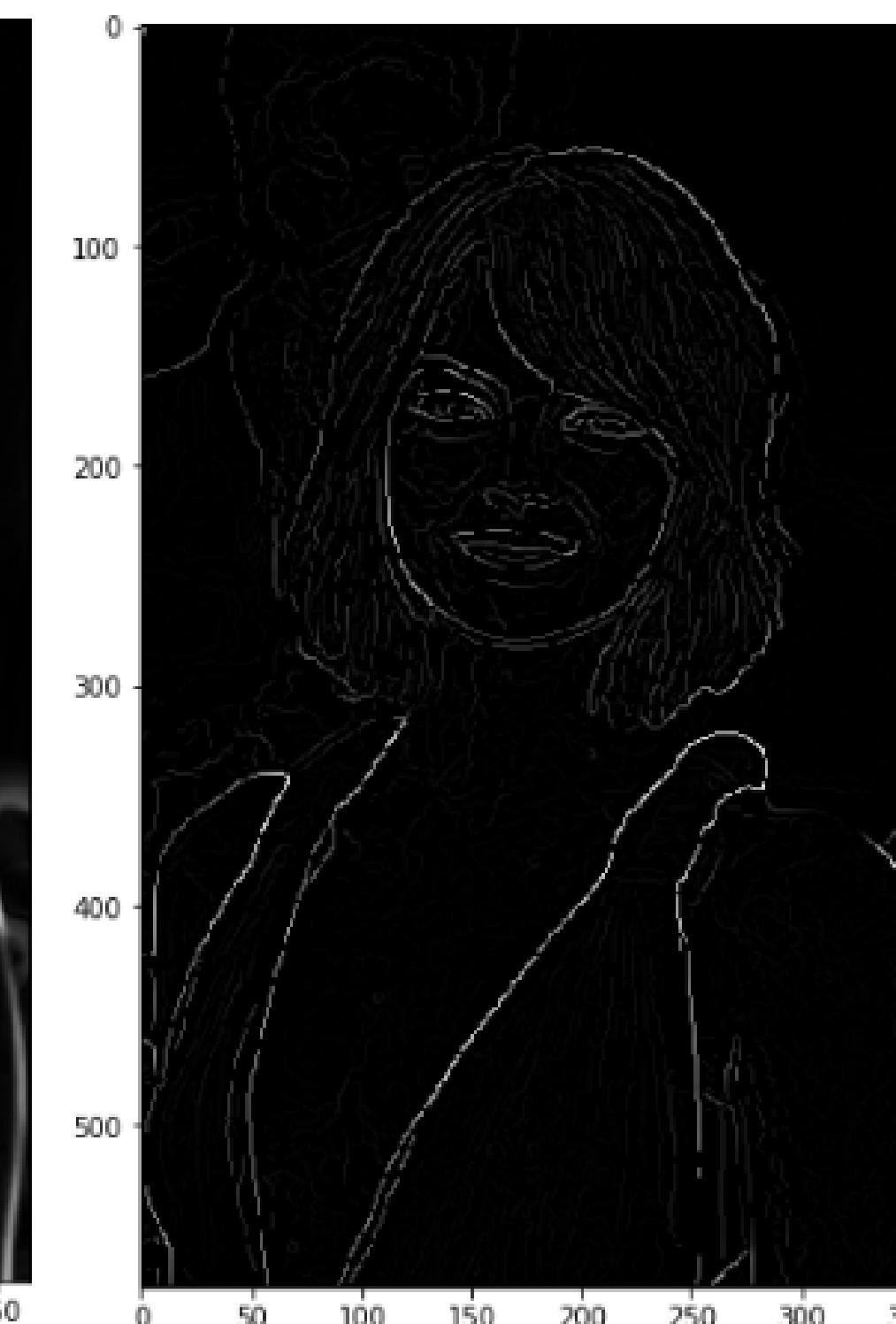
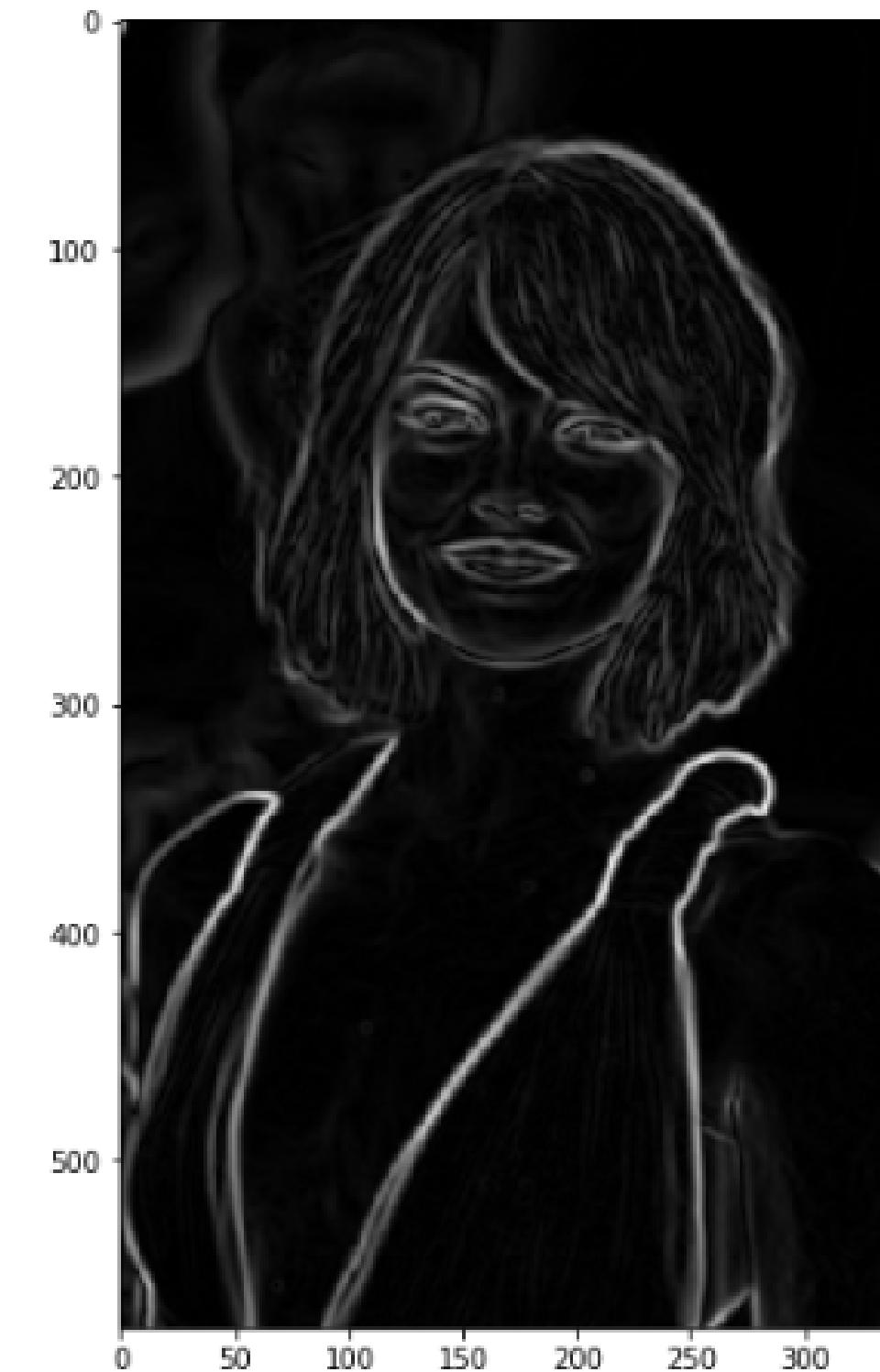
Il risultato di questo passo è la stessa immagine, ma con bordi più fini, rimane il problema della variazione dell'intensità.





52

Non-Maximum Suppression





Double Threshold

L'obiettivo di questo passo è identificare 3 tipi di pixel:

- **Forti**: alta intensità, siamo certi che appartiene al bordo finale
- **Deboli** : intensità abbastanza alta per essere rilevanti, ma non abbastanza per essere forti
- **Non rilevanti**

Utilizzo due soglie:

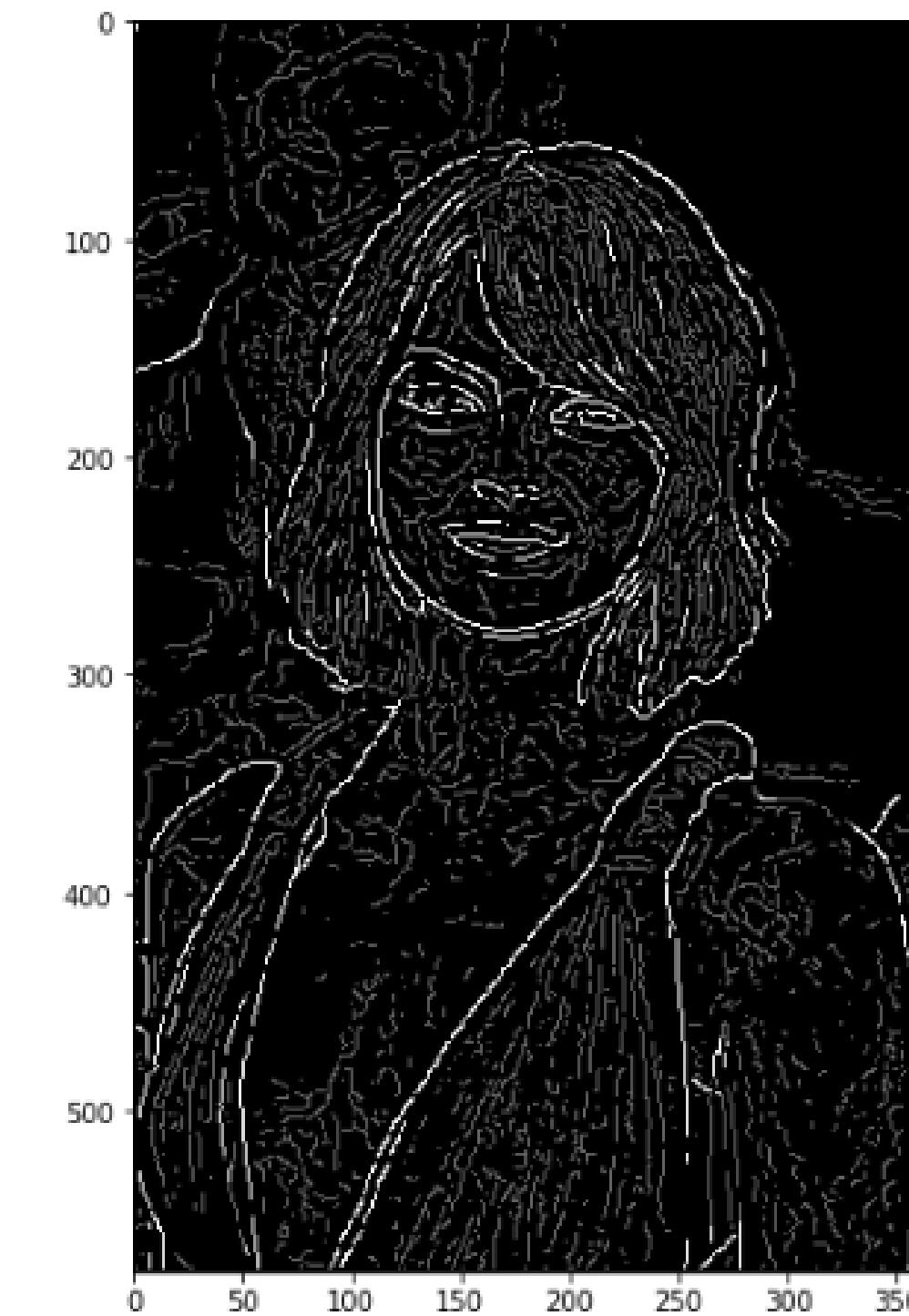
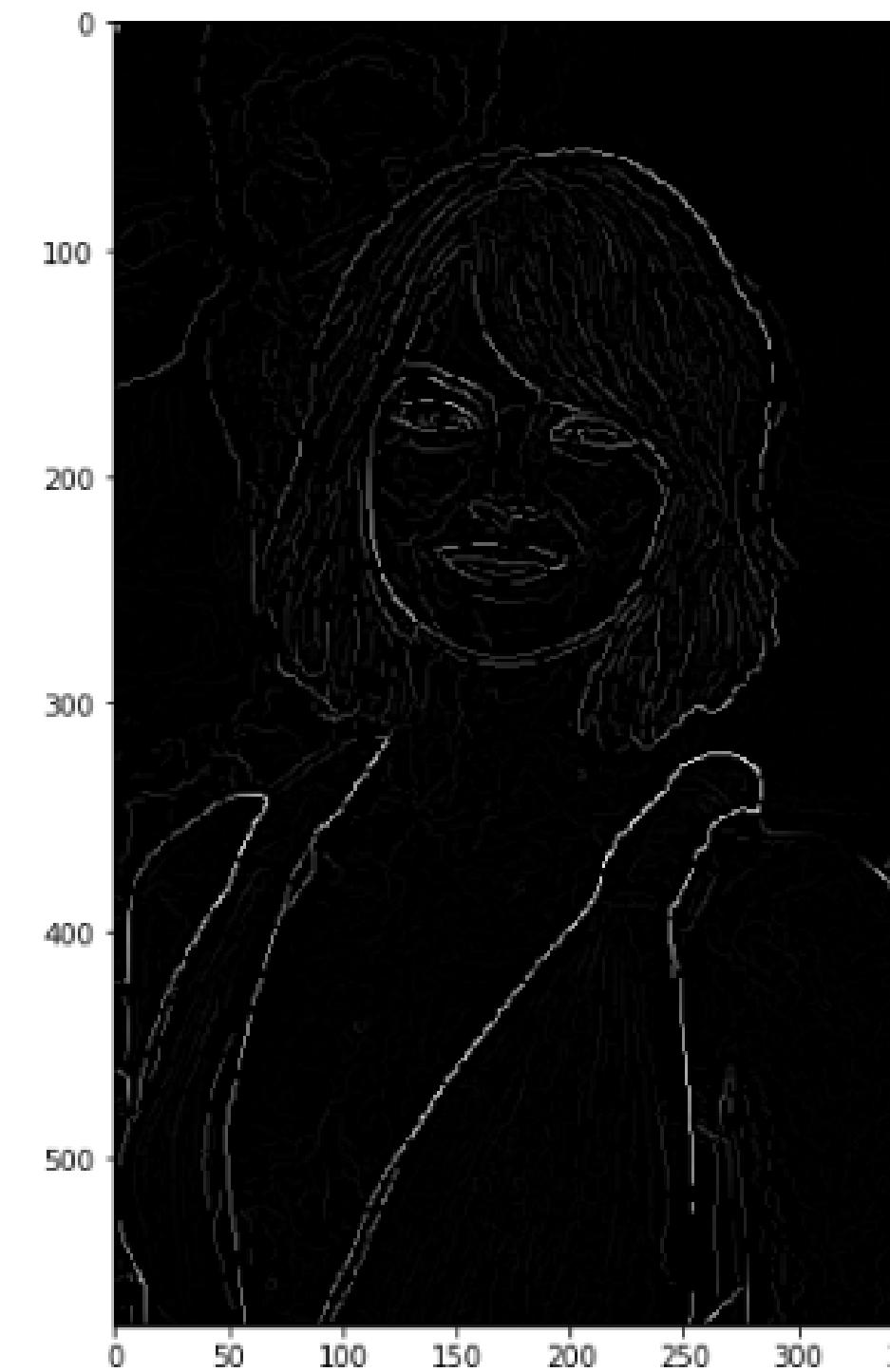
- alta per identificare i pixel forti
- bassa per identificare i non rilevanti

I pixel deboli cadono tra queste due soglie



54

Double Threshold



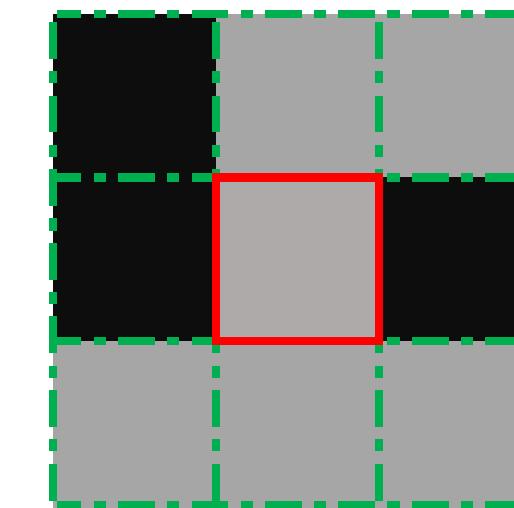


55

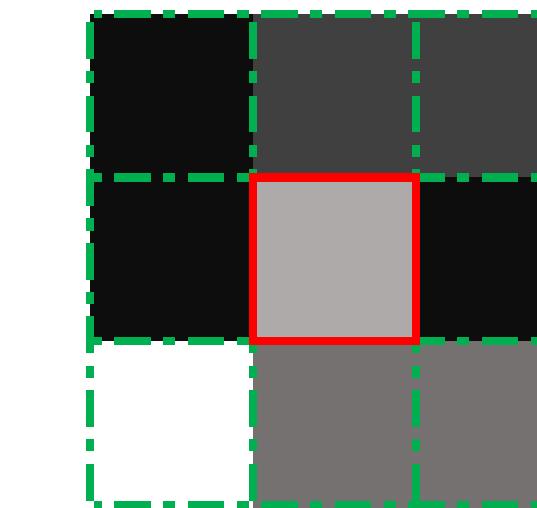
Edge Tracking by Hysteresis

L'obiettivo è quello di smistare i pixel deboli.

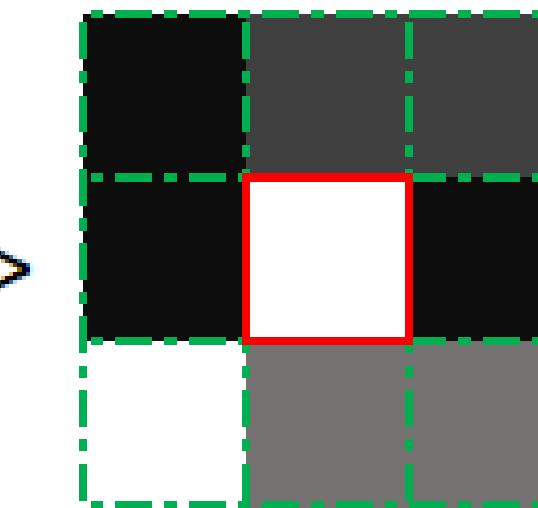
Trasforma i pixel deboli in forti solo se è circondato da almeno un pixel forte.



No strong pixels around



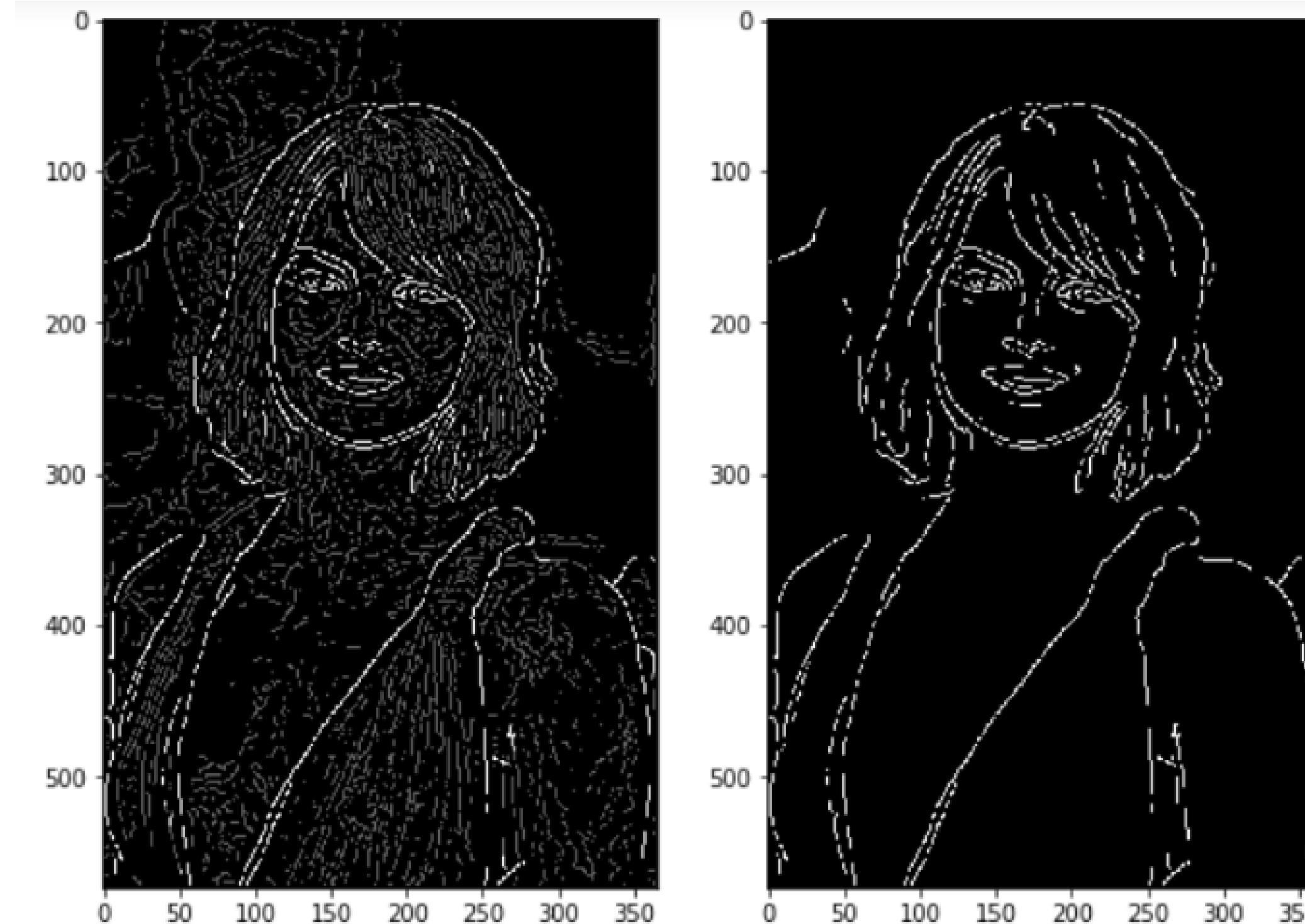
One strong pixel around





56

Edge Tracking by Hysteresi



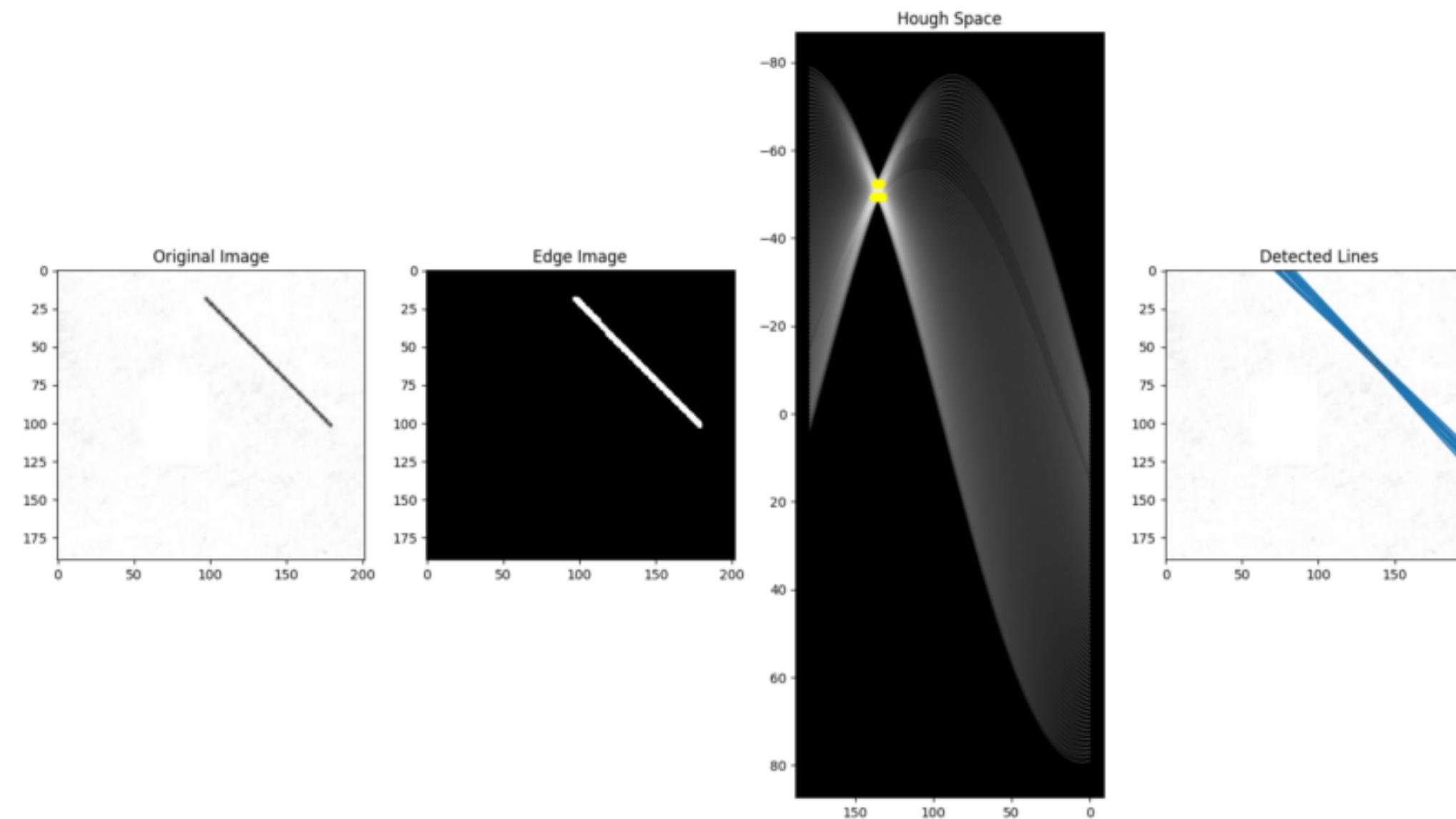


57

Hough Lines Transform

E' utilizzata per rilevare delle linee in un'immagine

Necessita come input una Edge image(output del Canny edge detection alg.)





58

Rappresentazione di una retta

Al posto di utilizzare il sistema di coordinate cartesiane (m, b),
è utilizzato il sistema di coordinate polari (r [rho], Θ [theta])

$$r = x \cos \theta + y \sin \theta \quad y = \left(-\frac{\cos \theta}{\sin \theta} \right) x + \left(\frac{r}{\sin \theta} \right)$$

Per ogni punto (x0,y0) esiste un fascio di rette che passa per quel punto:

$$r_\theta = x_0 \cdot \cos \theta + y_0 \cdot \sin \theta$$

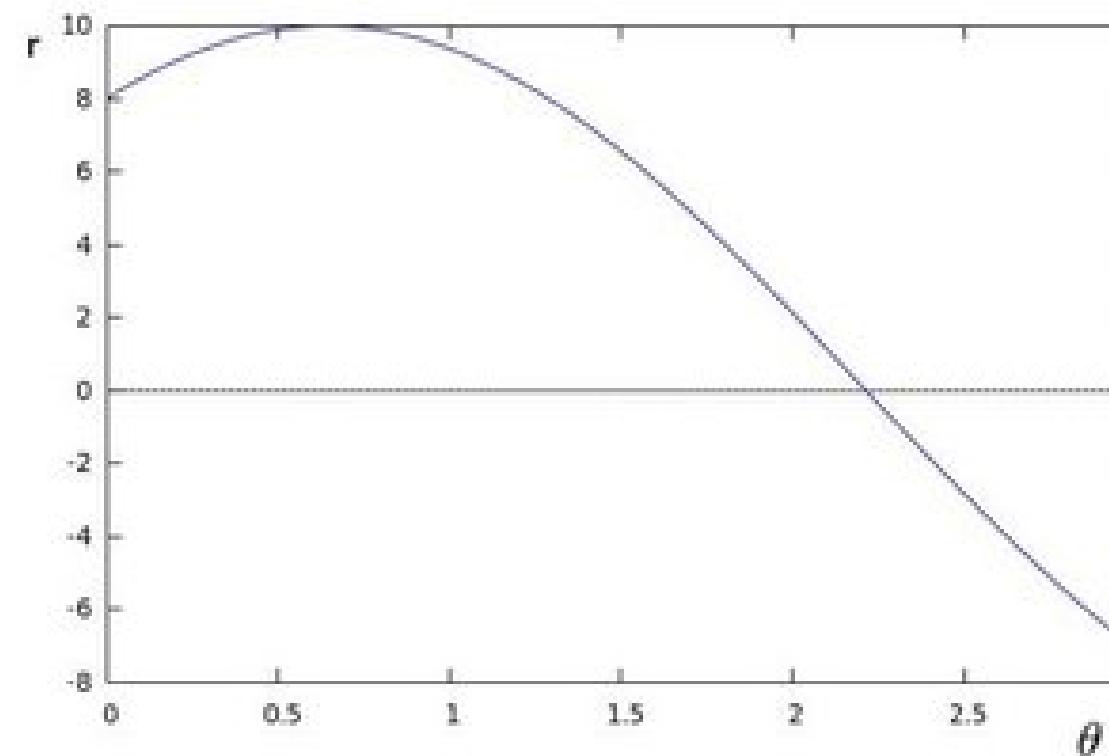


59

Hough Space

Se si plotta in un piano 2D ($x : \Theta$, $y : r$) questa famiglia di rette formano una sinusode.

[considero solo $r > 0$ e $0 < \Theta < 2\pi$]

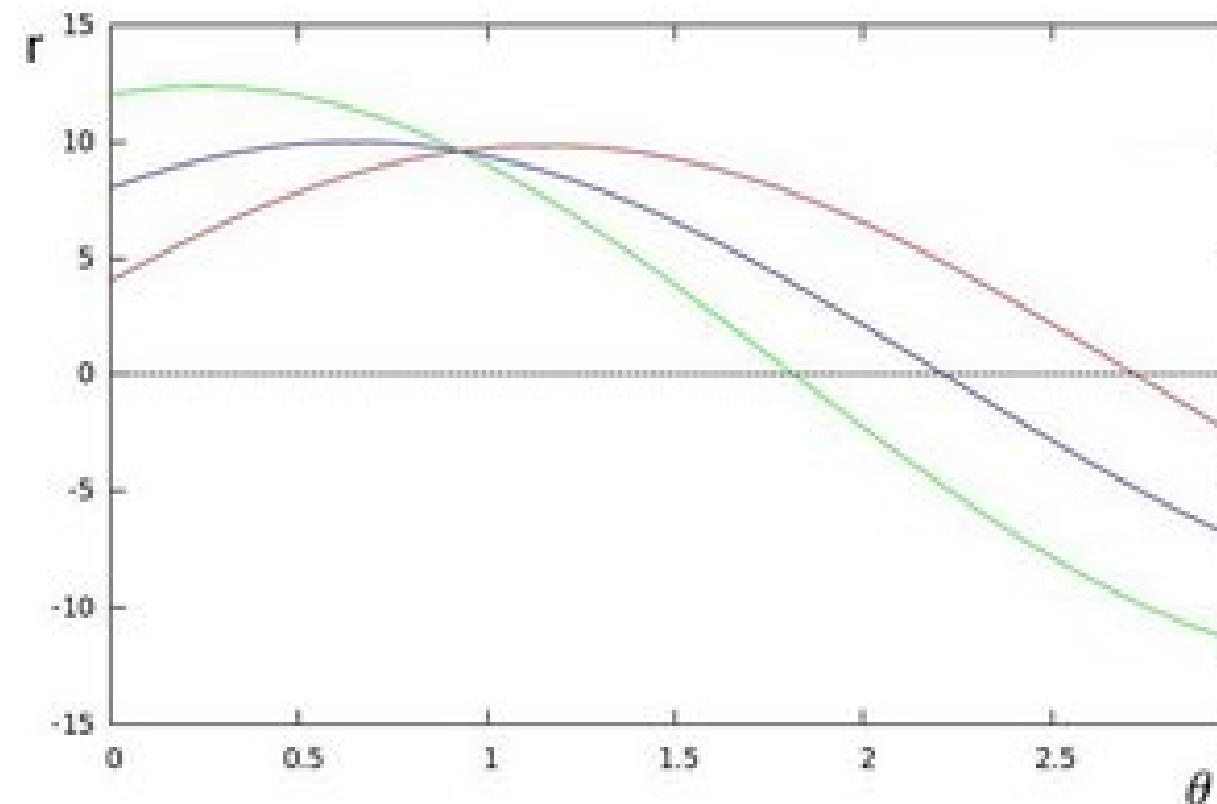




60

Hough Space

Posso fare questa operazione per ogni punto di mio interesse, se le curve di due (o più) punti diversi si intersecano in un punto del piano di Hough, questo vuol dire che i punti appartengono alla stessa linea, le coordinate di quella intersezione equivalgono ai parametri (r, Θ) della retta in cui i punti risiedono.





61

Threshold

Una linea può essere rilevata trovando il numero di intersezioni tra le curve. Il numero di curve che si intersecano equivale al numero di punti che appartengono alla linea rappresentata, quindi si può indicare una soglia (threshold) di numero di intersezioni minime per definire una linea.



Hough Line Transform in OpenCV

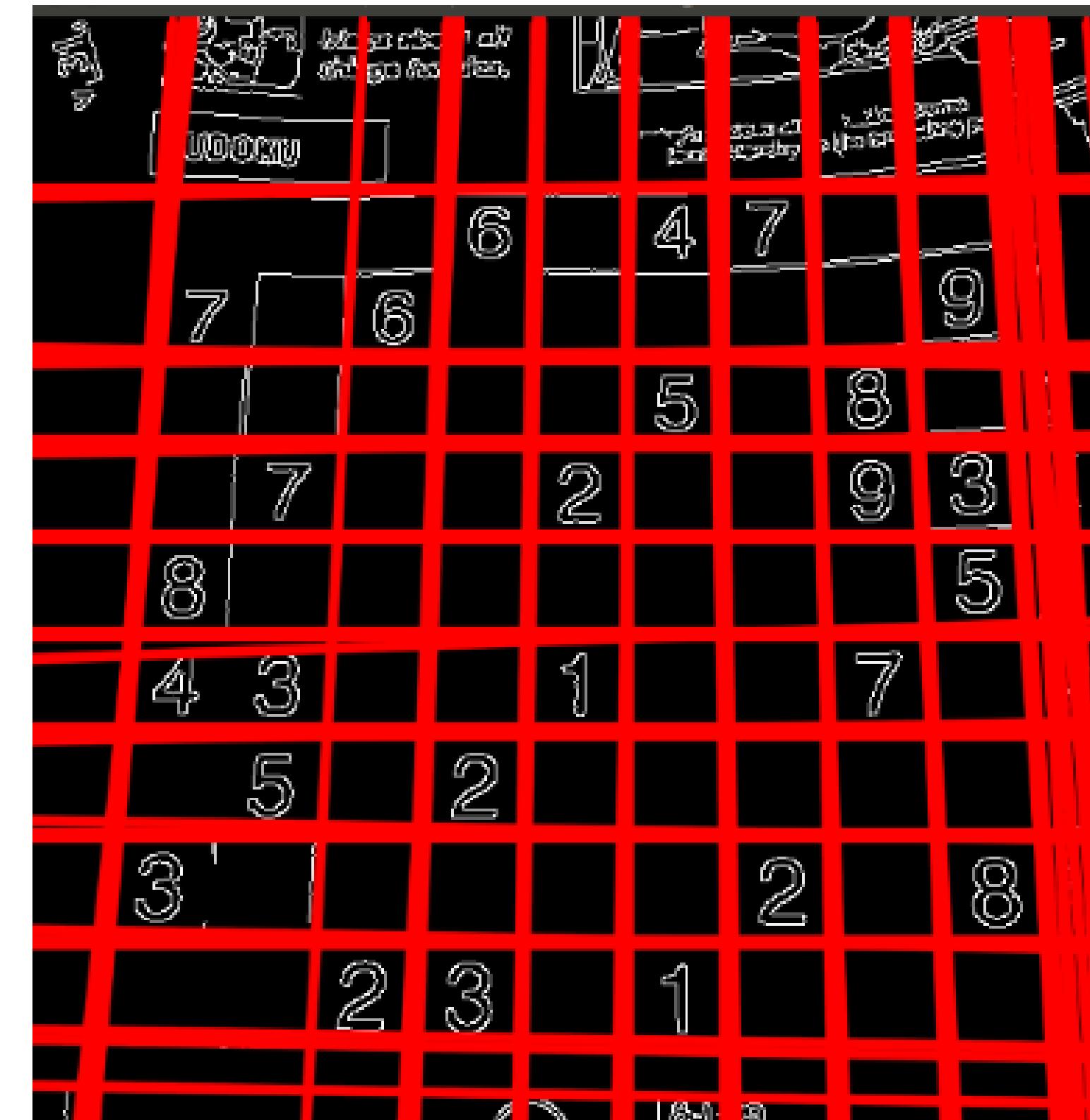
Due tipi:

- **Standard**: HoughLines()
 - Restituisce una lista di rette definite dalle coordinate polari
 - Permette di definire il threshold
- **Probabilistica**: HoughLinesP()
 - Implementazione più efficiente
 - Ritorna una lista di coppie di punti, che sono gli estremi delle linee rilevate
 - Oltre al threshold, permette di definire
 - il numero minimo di punti per definire una linea
 - la lunghezza massima di stacco tra due punti



63

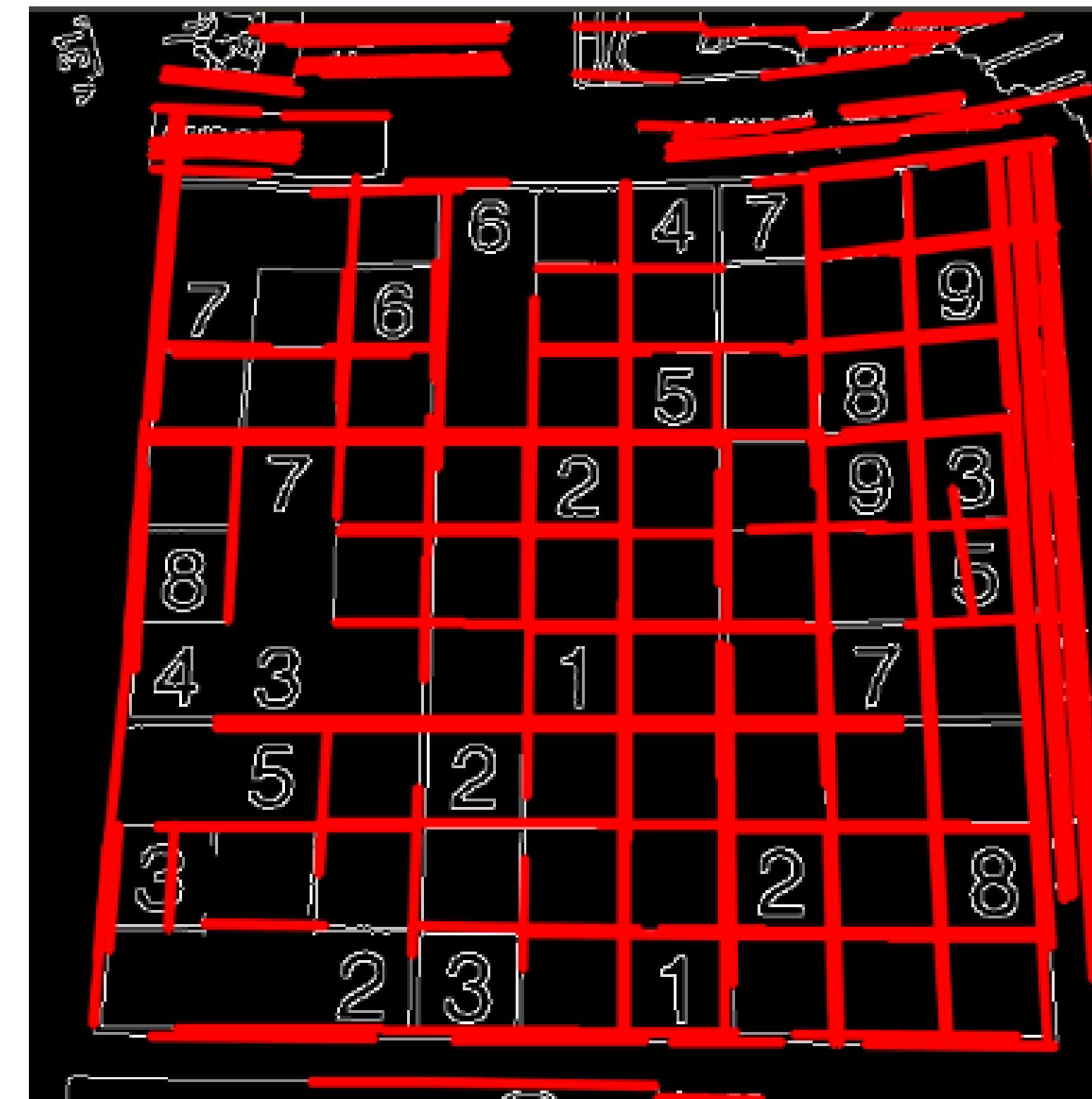
Esempio standard





64

Esempio probabilistico



FINE