



**Implementazione di algoritmi di Computer Vision tramite approccio classico e Deep Learning. Organizzazione di un progetto di Data science.**

[Github repo](#)



## **Sezione 1**

# **Introduzione alla computer Vision**

# Introduzione alla computer vision



## Cosa è la Computer Vision

**La computer vision è un campo dell'informatica che si occupa di insegnare ai computer a interpretare e comprendere le immagini e i video digitali.**

### Evoluzione

Dagli algoritmi tradizionali ai moderni approcci basati su Deep Learning

# Introduzione alla computer vision



## Applicazioni

- **Industria e Automazione:**
  - Controllo di qualità nelle linee di produzione.
  - Veicoli autonomi.
- **Sanità:**
  - Diagnosi tramite analisi di immagini mediche (radiografie, TAC).
- **Retail:**
  - Analisi del comportamento dei clienti.
  - Sistemi di checkout automatico.
- **Sicurezza:**
  - Riconoscimento facciale.
  - Sorveglianza intelligente.
- **Intrattenimento:**
  - Realtà aumentata (AR) e virtuale (VR).

## Task principali risolti dalla Computer Vision

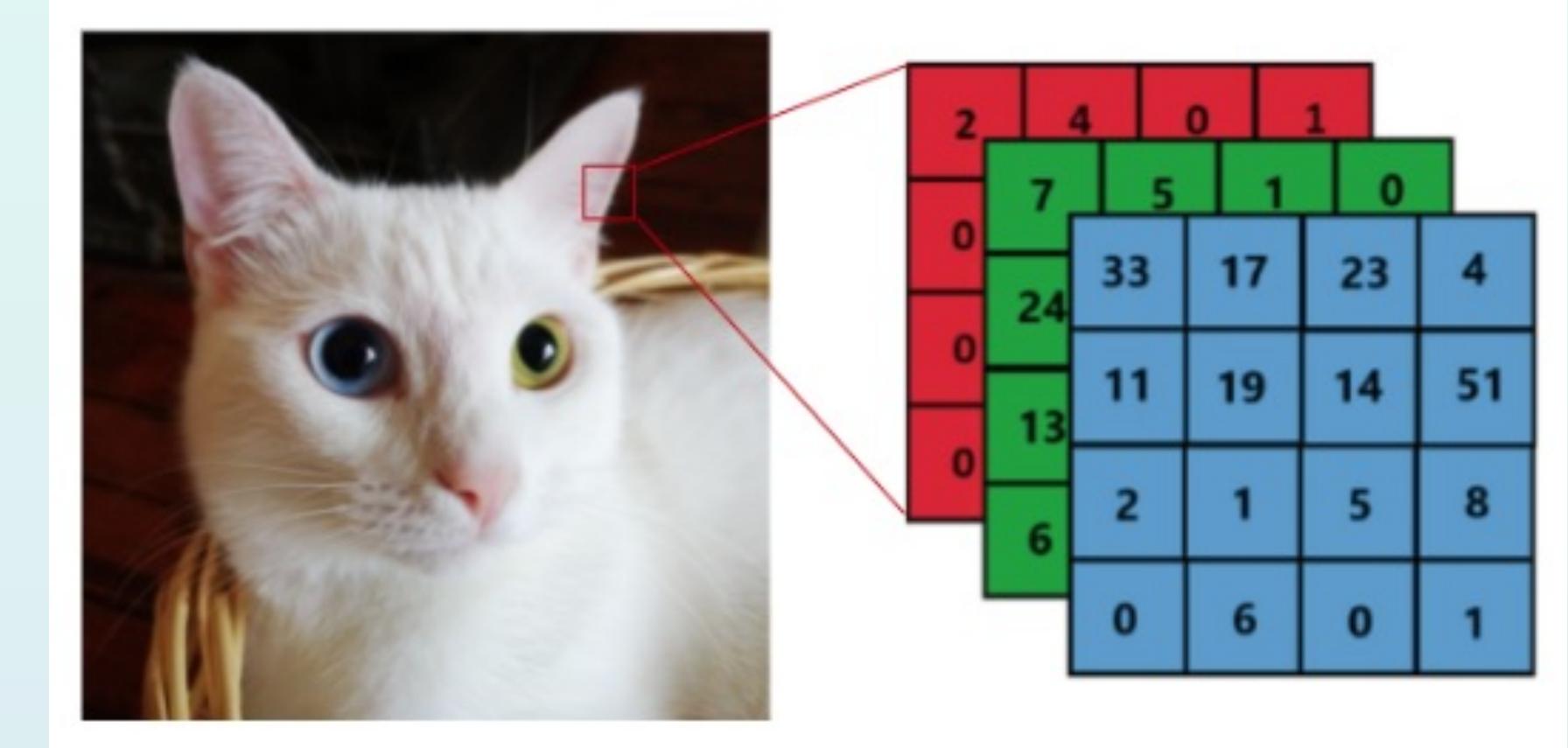
- **Image Classification:** Classificare un'immagine in una o più categorie predefinite.
- **Object Detection:** Identificare e localizzare oggetti specifici all'interno di un'immagine.
- **Object Segmentation:** Separare ogni oggetto in un'immagine con contorni precisi (es. segmentazione semantica o istanza).
- **Image Captioning:** Generare descrizioni testuali a partire da un'immagine.
- **Optical Character Recognition (OCR):** Riconoscere e digitalizzare testo presente in immagini o documenti.

# Introduzione alla computer vision

## Come sono rappresentate le immagini

**Le immagini sono rappresentate da matrici di numeri**

- Tipicamente tramite matrice di dimensione [3, H, W]
- 3 matrici 2D, una per ogni colore (RGB)
- I numeri rappresentano le intensità di ogni colore
- Le immagini in bianco e nero sono matrici [H, W]

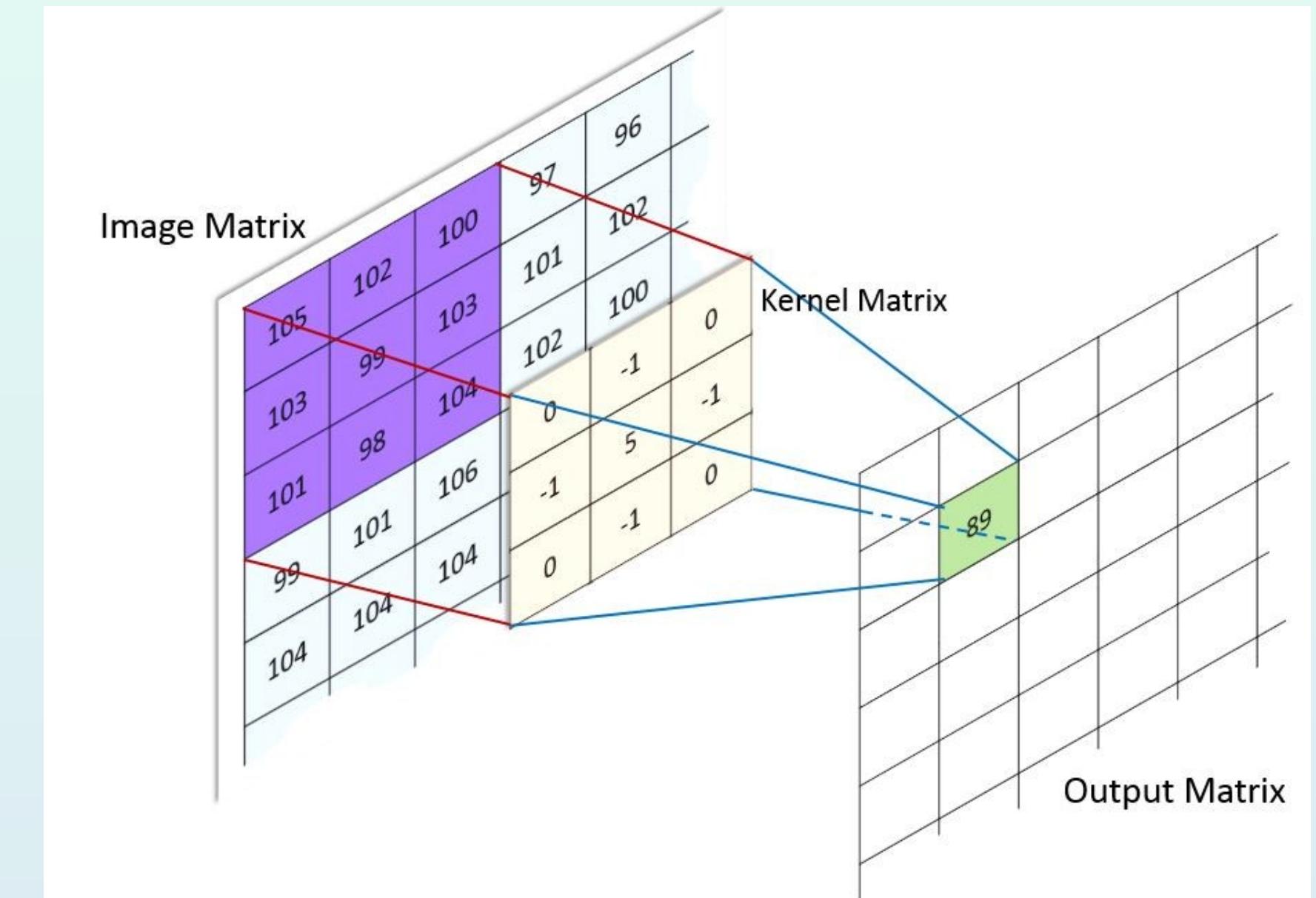


# Introduzione alla computer vision

## Image processing attraverso Kernel convoluzionali (1)

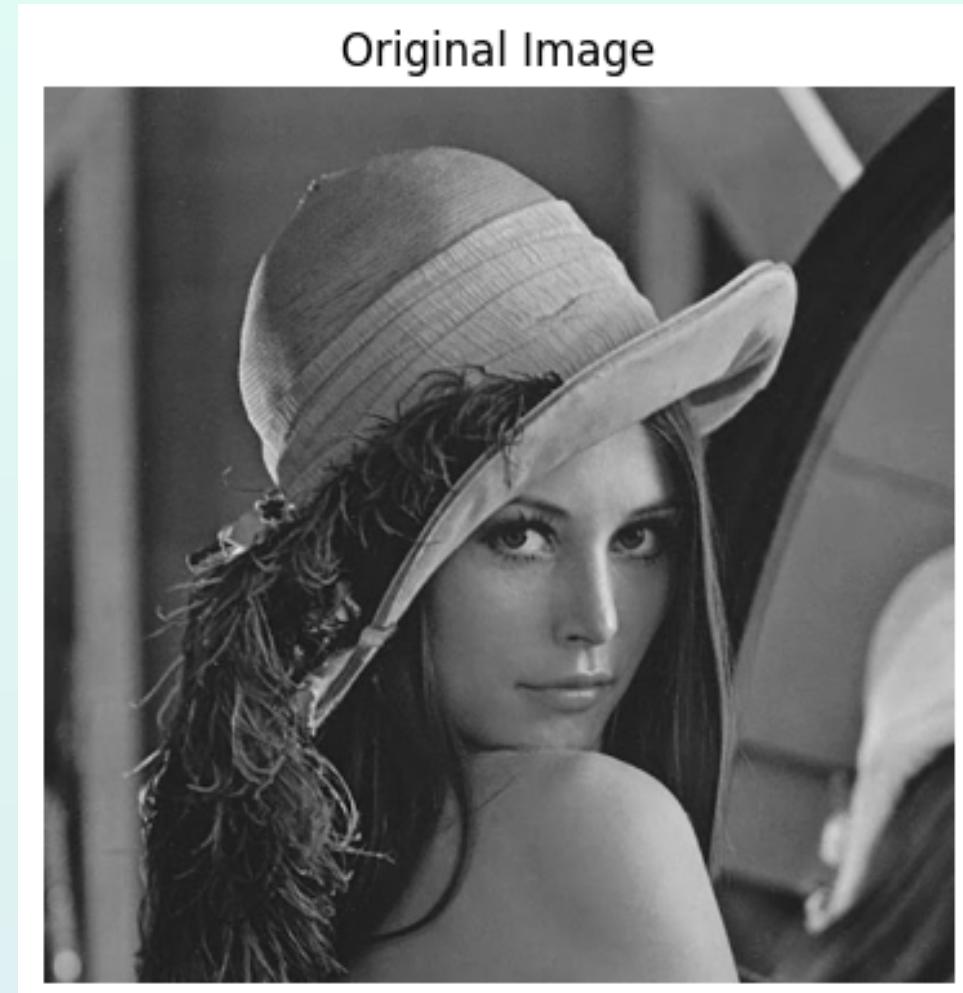
### Matrice di convoluzione (kernel)

- Piccola matrice usata per processare un immagine
- Operazione di convoluzione tra il kernel e l'immagine
- Ogni pixel in output, è una funzione dei pixel adiacenti dell'immagine di input



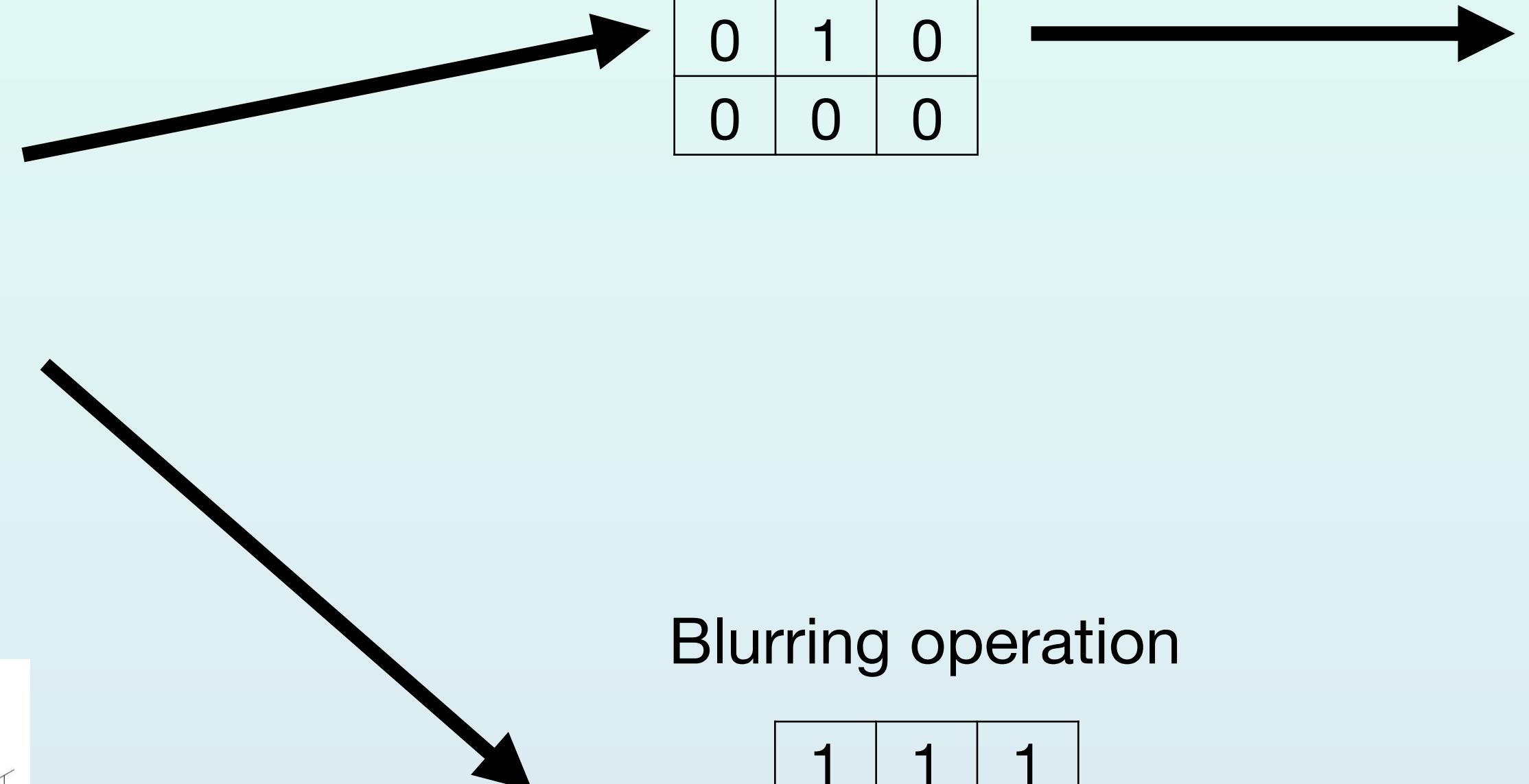
# Introduzione alla computer vision

## Image processing attraverso Kernel convoluzionali



Identity operation

0	0	0
0	1	0
0	0	0

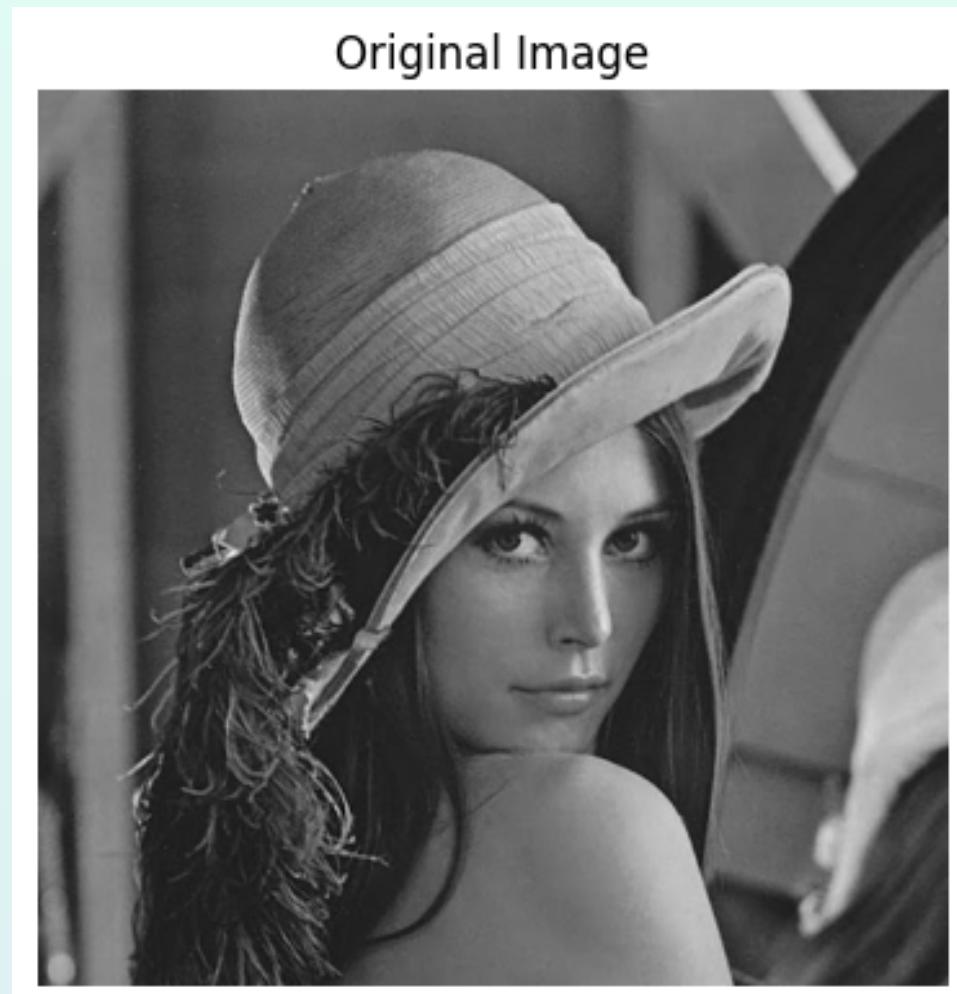


Blurring operation

$$1/9 * \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

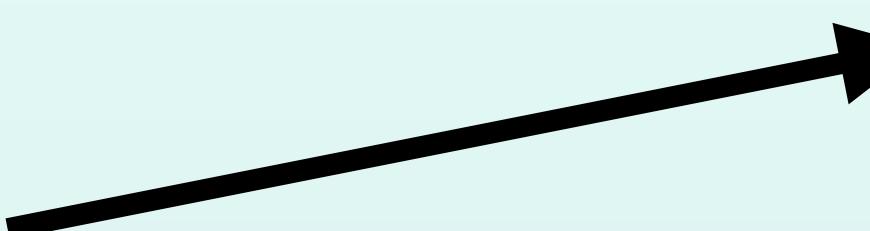
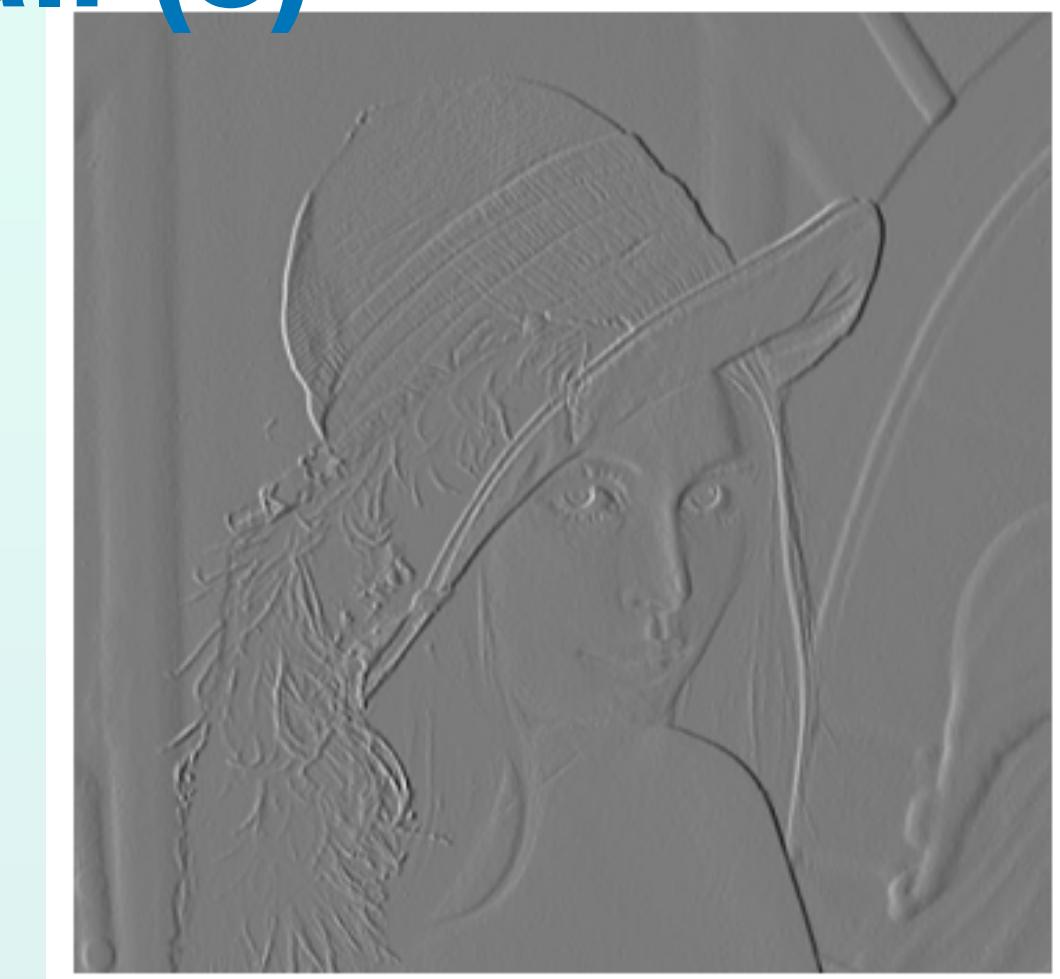

# Introduzione alla computer vision

## Image processing attraverso Kernel convoluzionali (3)



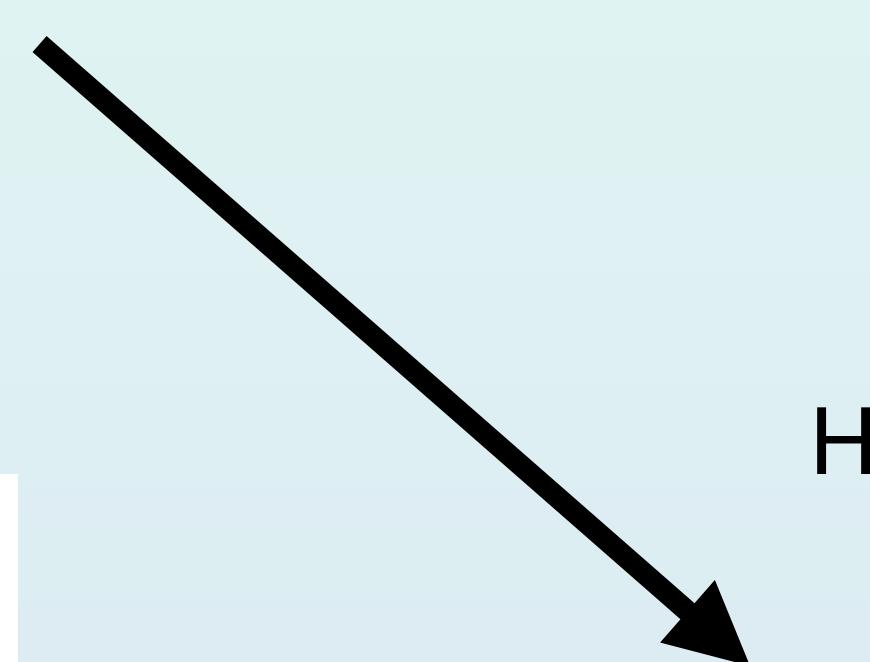
Vertical sobel operation

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



Horizontal sobel operation

1	2	1
0	0	0
1	2	1



## Alcuni algoritmi per la computer vision

- **Algoritmi tradizionali:**

- **SIFT (Scale-Invariant Feature Transform):** Utilizzato per il rilevamento di punti chiave e la descrizione di caratteristiche invarianti rispetto a scala e rotazione.
- **ORB (Oriented FAST and Rotated BRIEF):** Algoritmo rapido ed efficiente per rilevare e descrivere caratteristiche.
- **Watershed Algorithm:** Metodo di segmentazione delle immagini basato sul concetto di bacini idrografici.

### Vantaggi

- Interpretabilità.
- Meno risorse computazionali rispetto ai metodi moderni.

### Limiti

- Difficoltà con dati complessi o non strutturati.
- Richiede un design manuale delle caratteristiche.
- Difficoltà con task complessi

## Approccio Deep Learning

**Sottodisciplina del ML basata su reti neurali profonde**

Modelli molto più complessi

Richiesta di grande quantità di dati per addestrarli

### Vantaggi

- Risolve task complessi
- Approccio data-driven

### Limiti

- Modello poco interpretabile
- Costoso a livello computazionale

# Introduzione alla computer vision

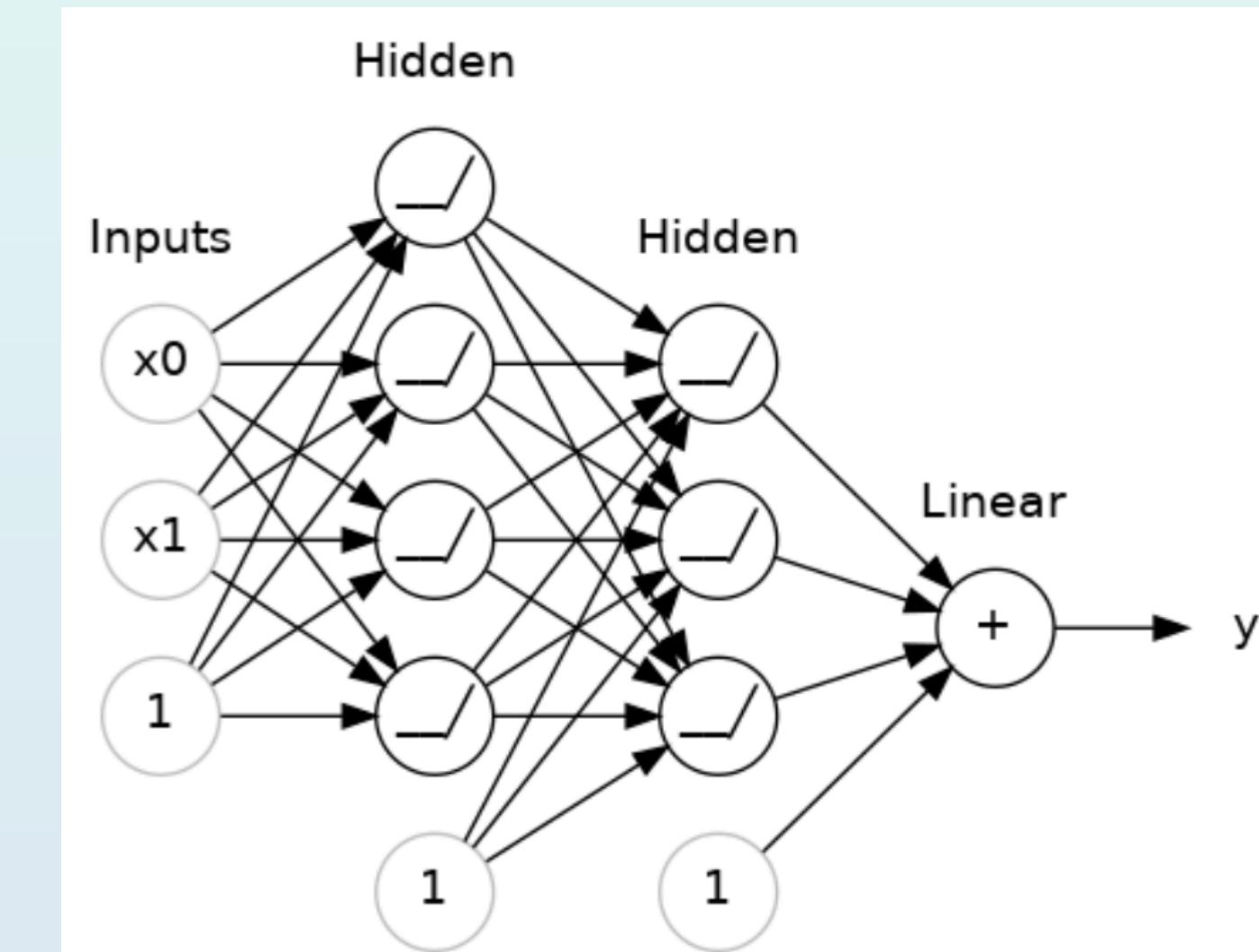
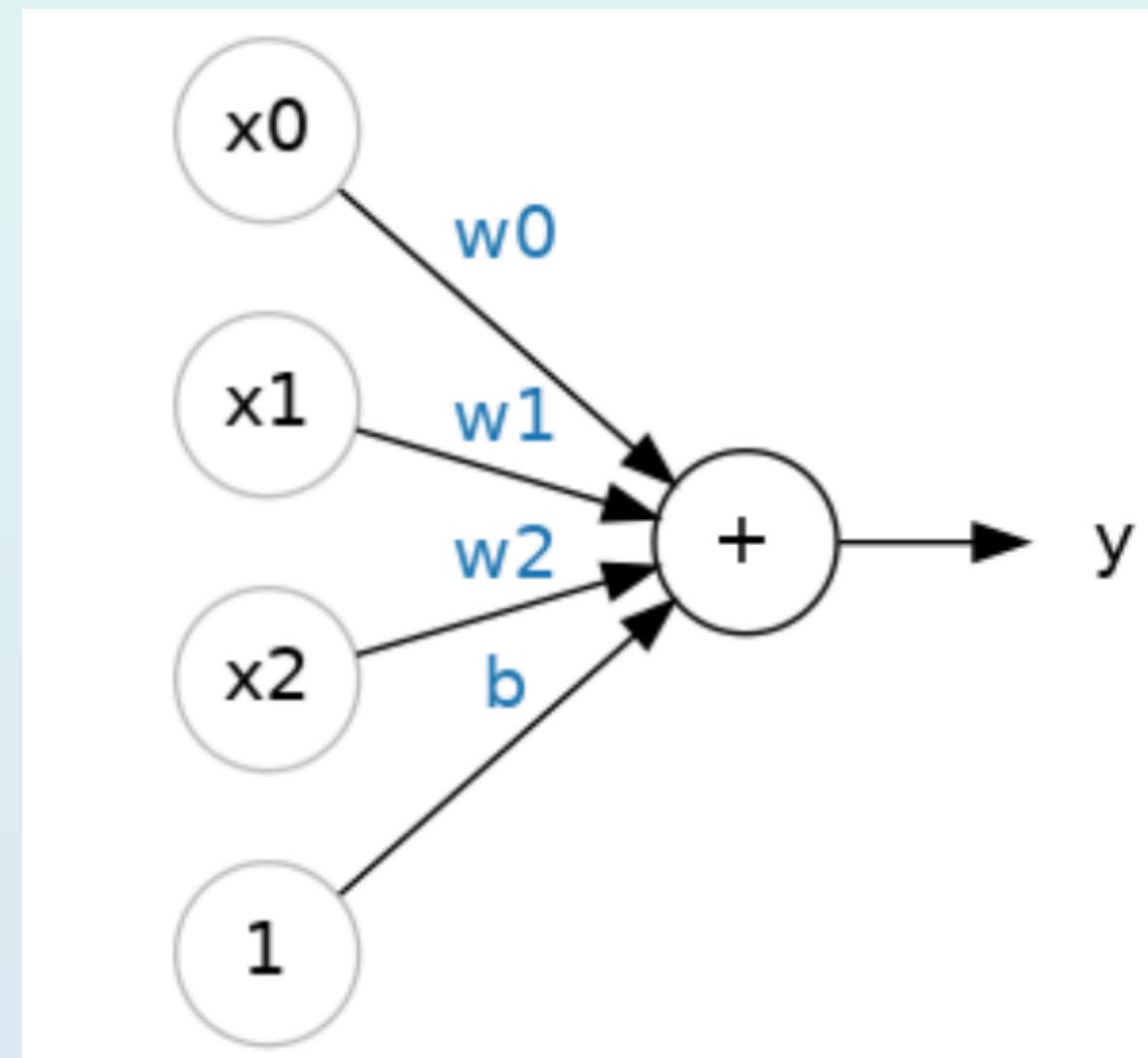
## Approccio Deep Learning

L'unità base di una rete neurale è il singolo neurone.

- Può modellare funzioni semplici

Vengono combinati più neuroni e aggiunte funzioni non lineari.

- Possibilità di modellare funzioni molto complesse
- Ogni strato della rete viene chiamato **layer**



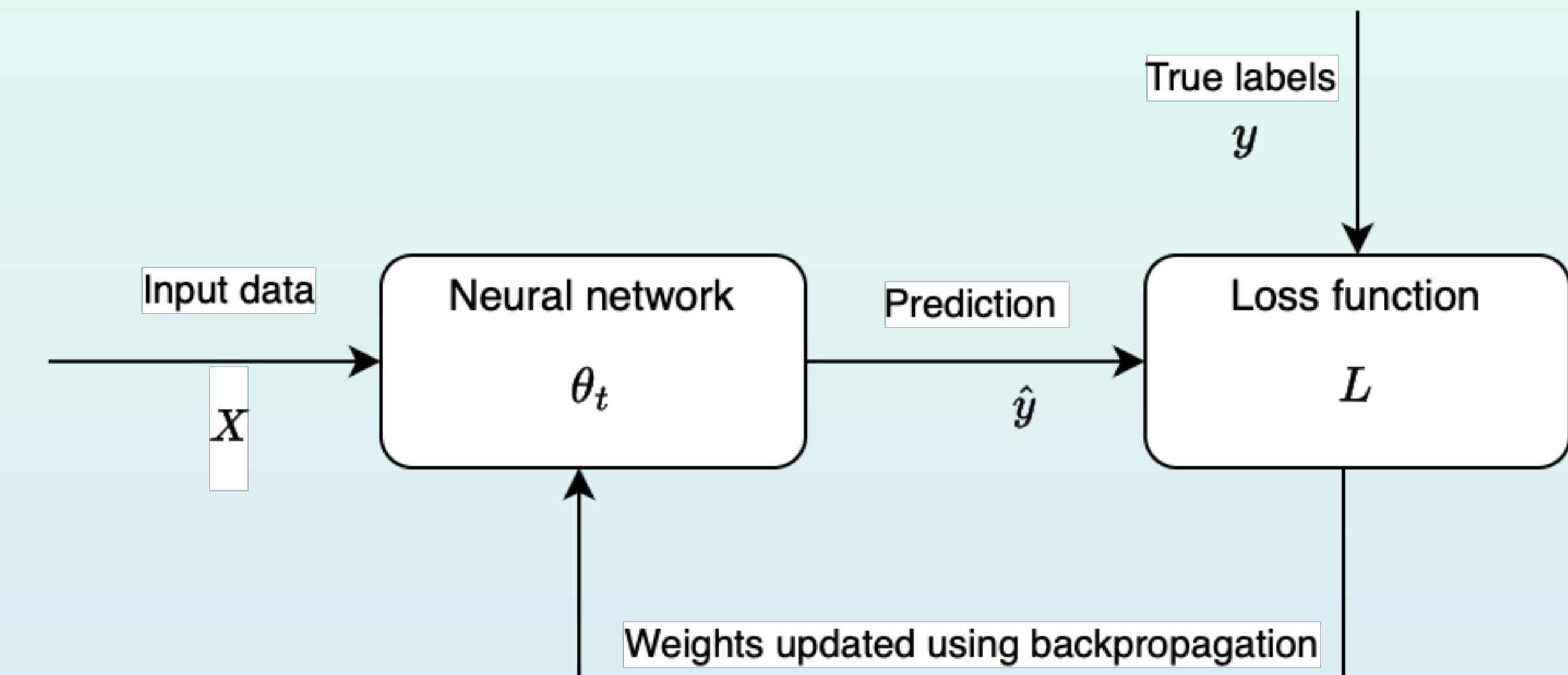
## Funzione di Loss e addestramento

### Loss

- Funzione di perdita o di errore che misura quanto la predizione della rete dista dalla true label.

### Backpropagation

- Algoritmo per ottimizzare reti neurali. L'errore sulla Loss viene propagato all'indietro verso i pesi della rete.



### Optimization

- Minimizza la Loss aggiornando i pesi della rete

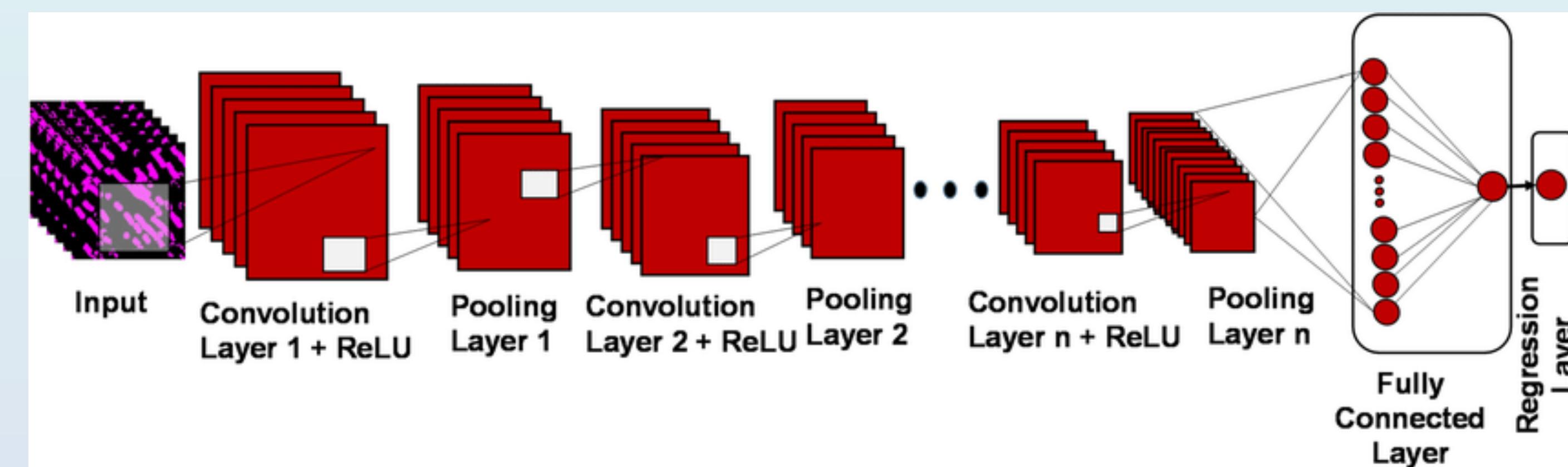
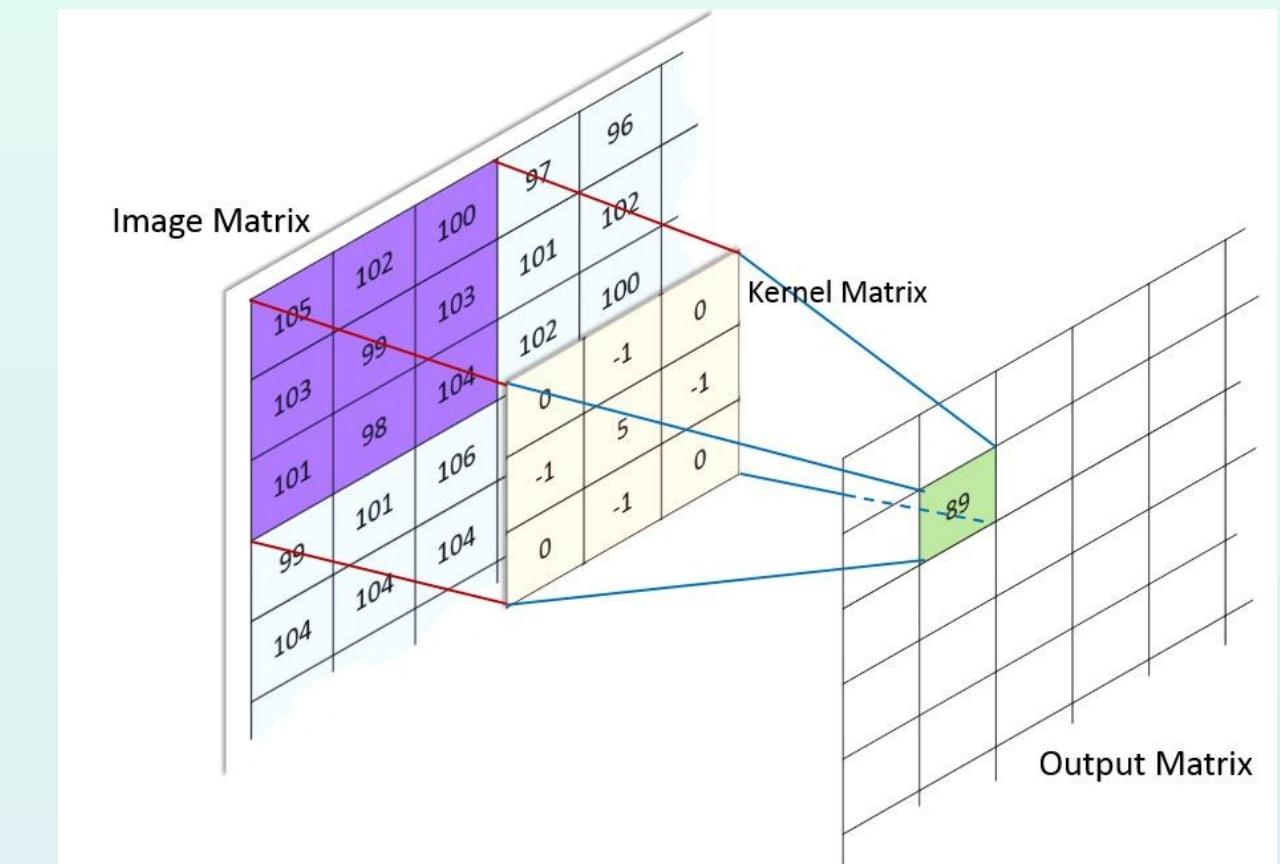
# Introduzione alla computer vision

## Convolutional layers

**Nella computer vision, vengono adottati solitamente i layer Convoluzionali**

- Sostituiscono i layer Fully-connected che invece sono general purpose
- Posseggono proprietà che si desiderano nell'elaborazione delle immagini (invarianza spaziale)
- Meno parametri, più efficienza

Operazione di convoluzione





**Sezione 2**

# **Progetto count-coins**

# Progetto count coins

## Introduzione

Goal:

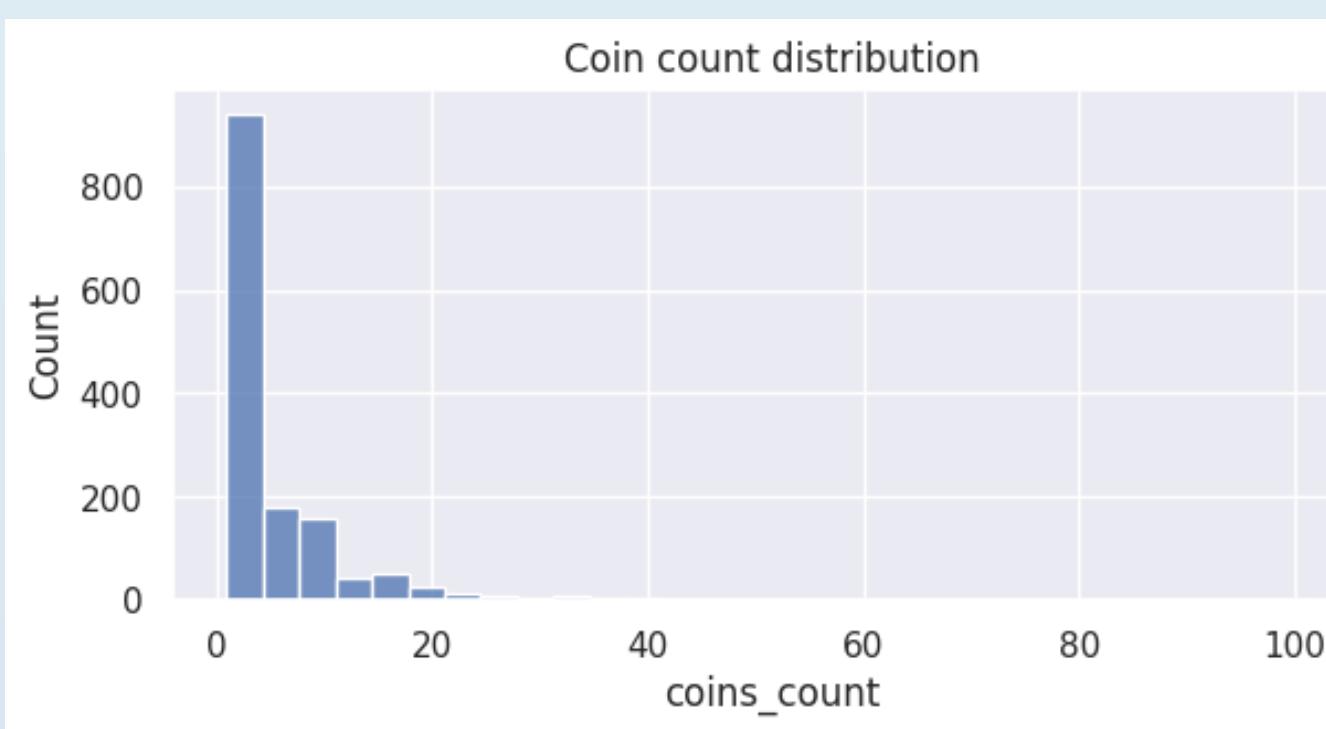
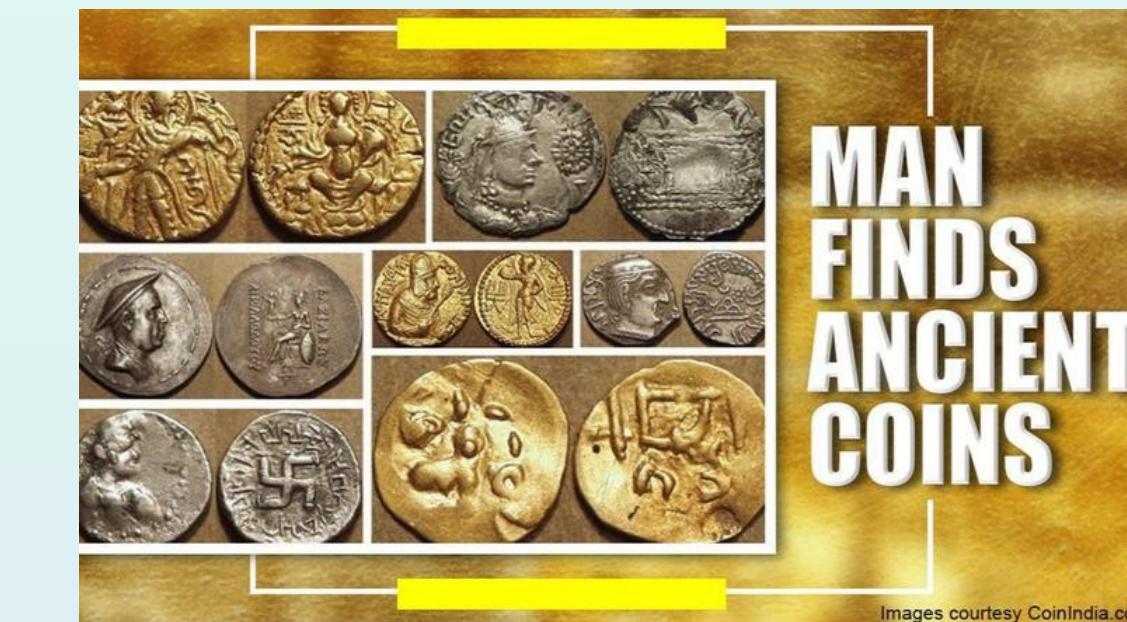
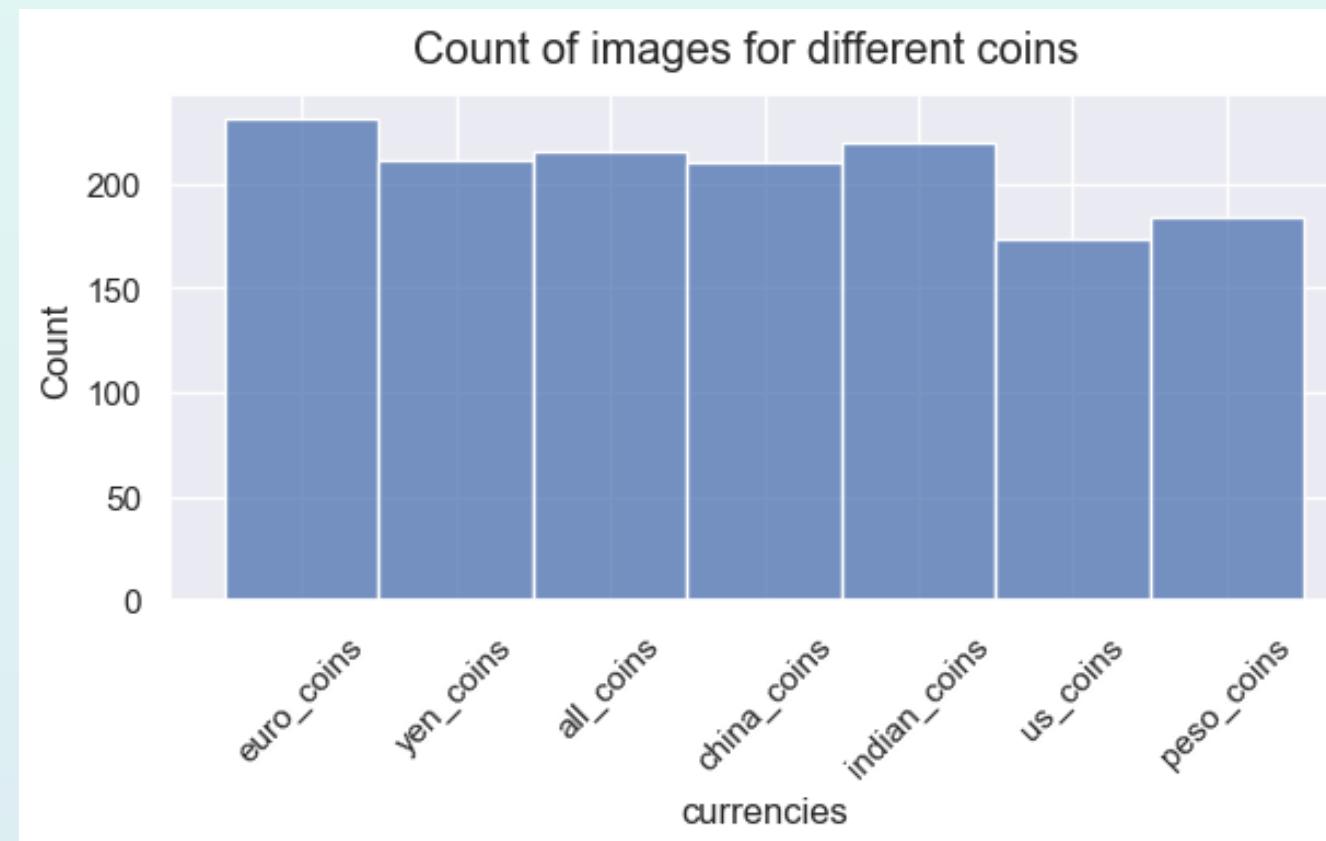
**Sviluppare un algoritmo che, data una immagine, conta il numero di monete presenti in essa.**

- Problema di regressione
- E' possibile approcciare il problema in due modi:
  - Computer Vision classico
  - Deep Learning

# Progetto count coins

## Exploration data analysis

- 1444 immagini
- Monete di diverse valute

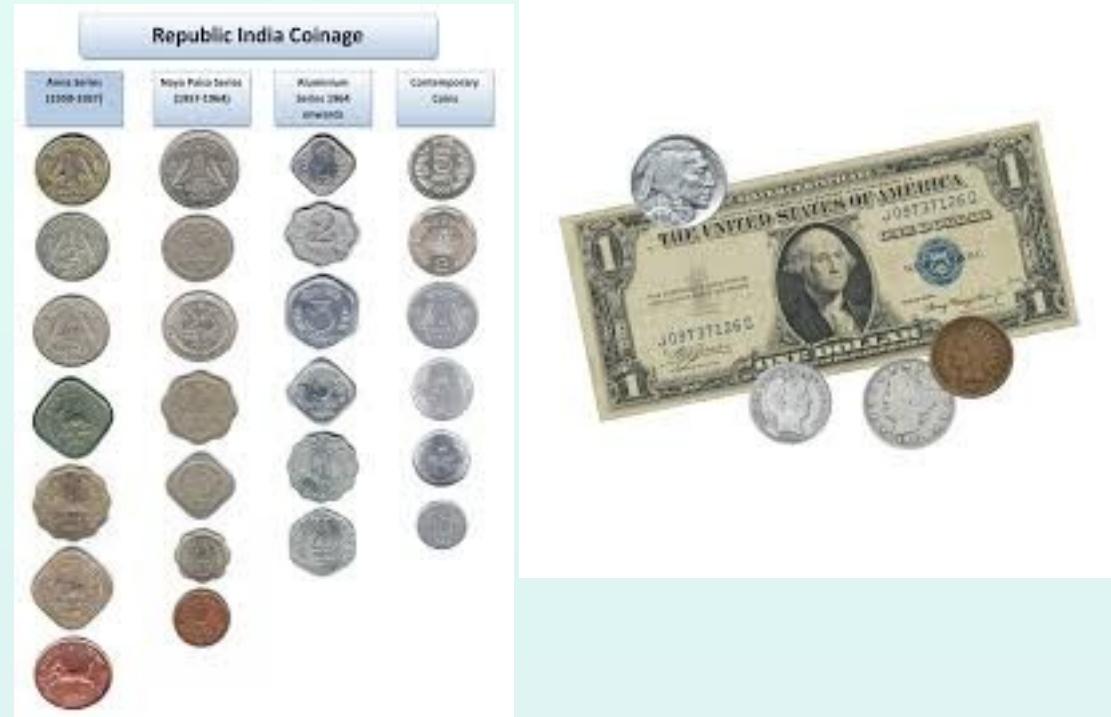


# Progetto count coins

## Dataset

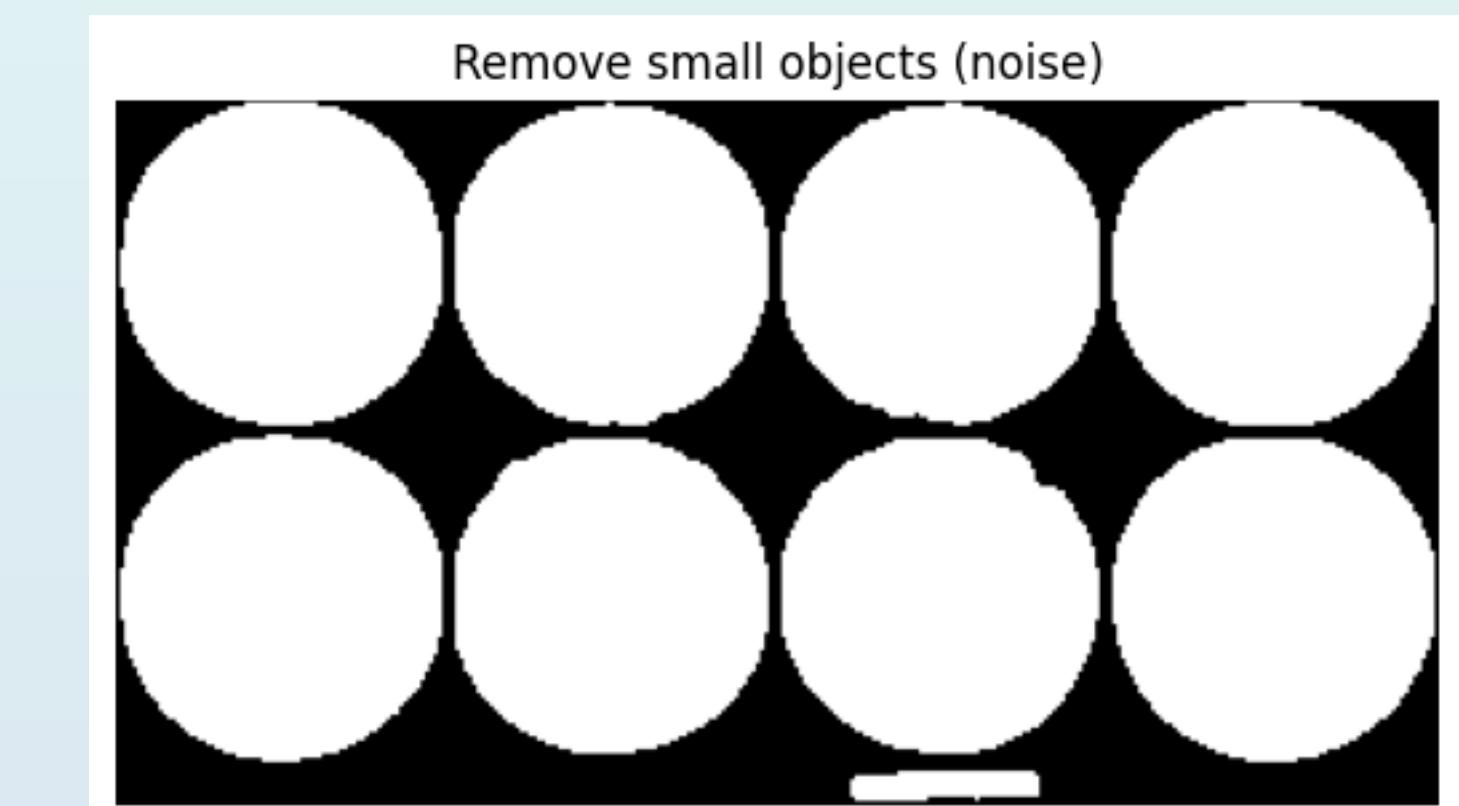
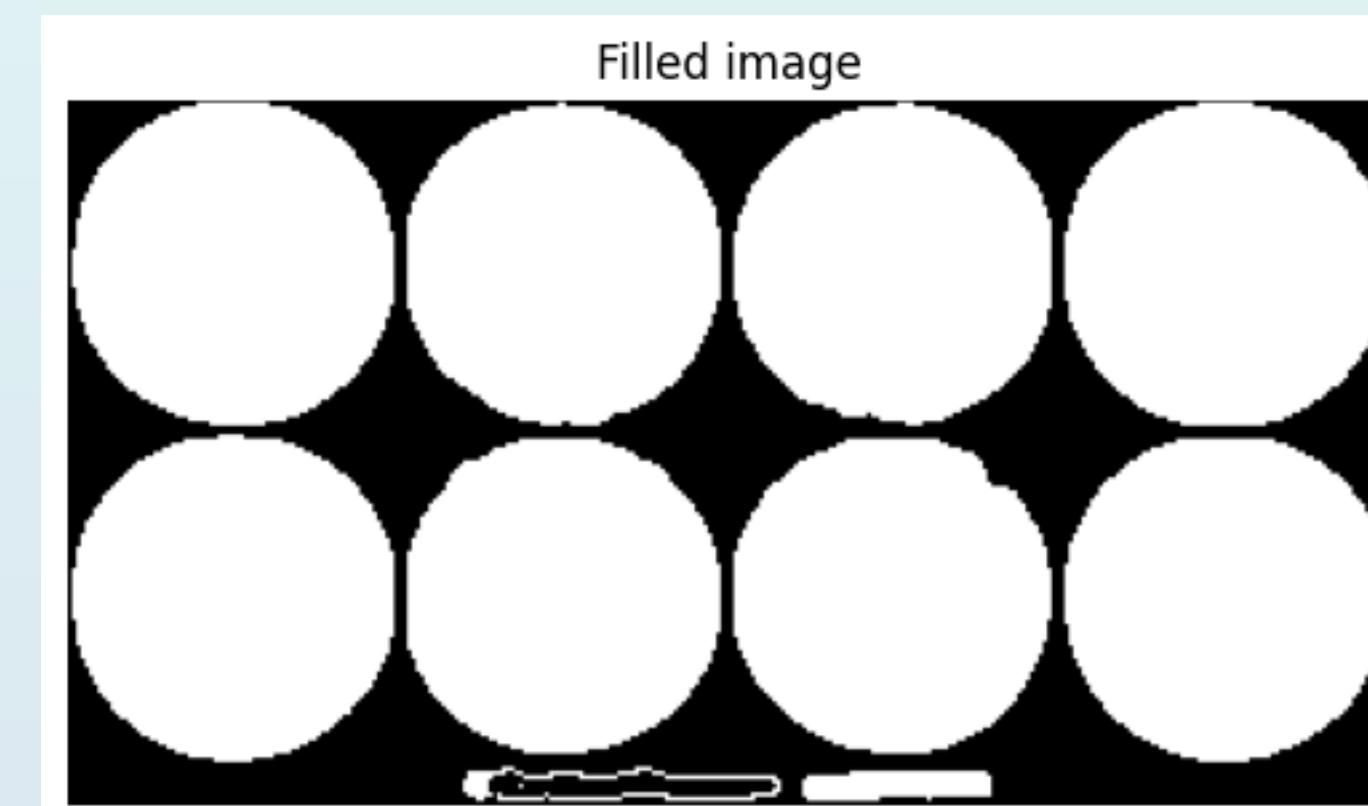
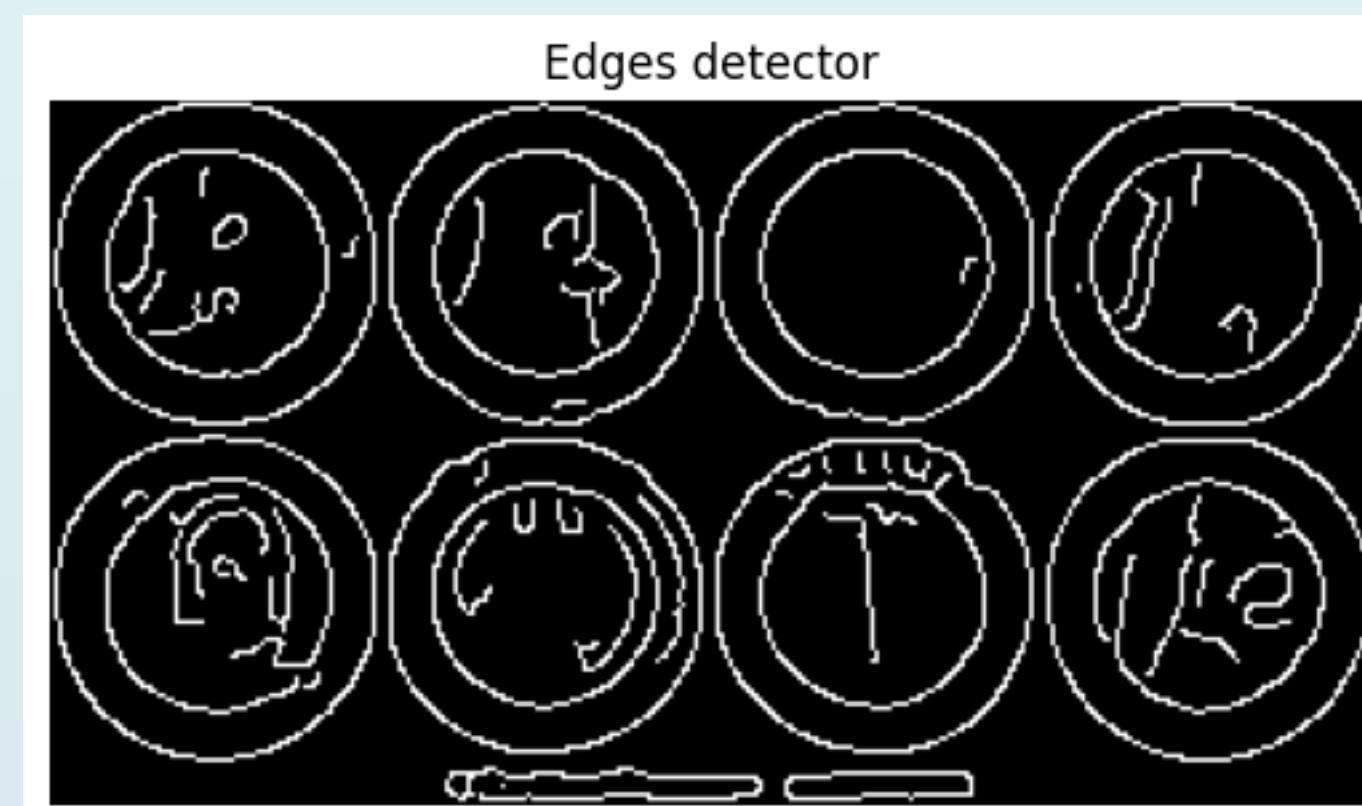
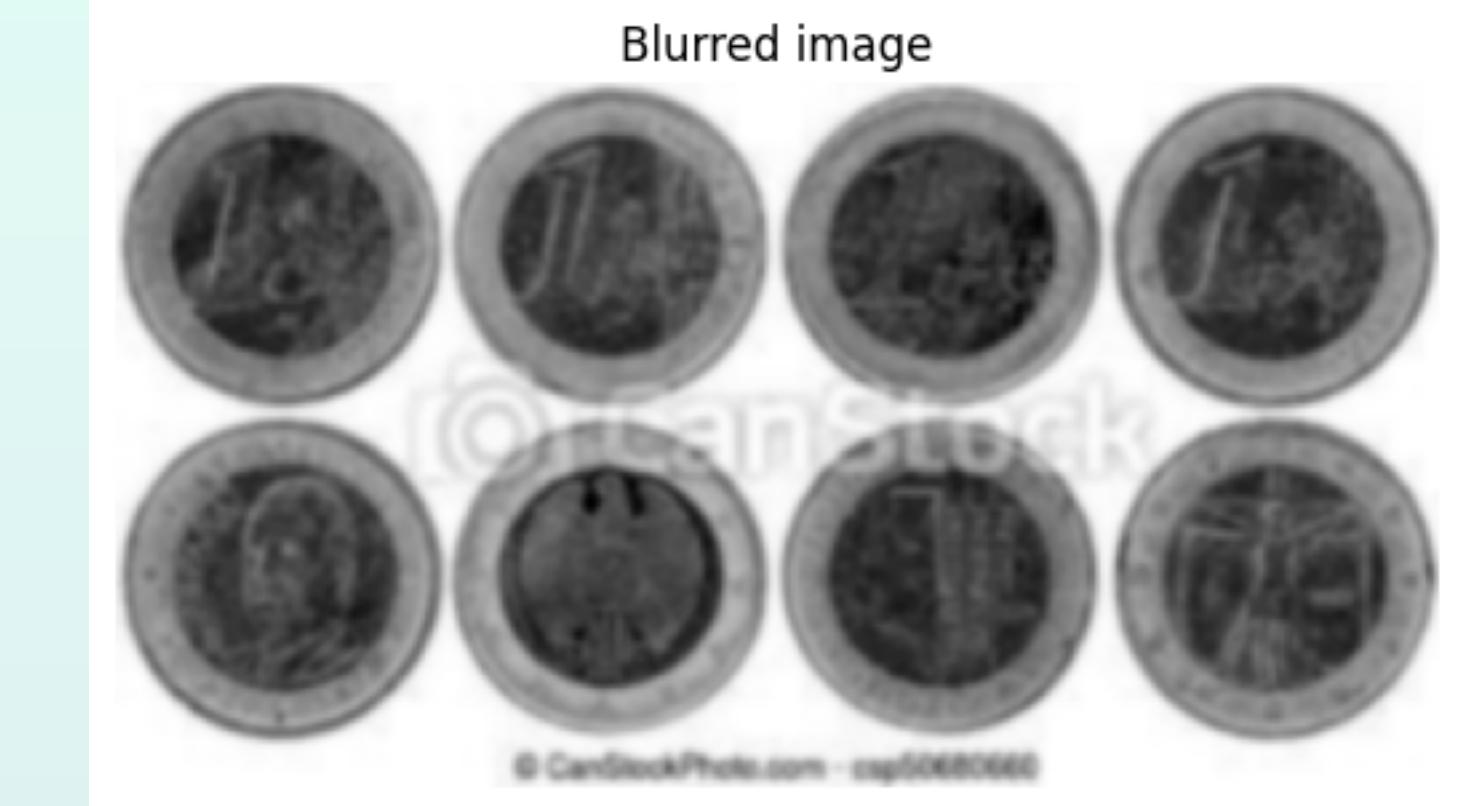
### Sfide:

- Non tutte le monete sono di forma circolare
- Il background non è lo stesso
- Il colore delle monete non è lo stesso
- Alcune monete sono sovrapposte
- Possono essere presenti altri elementi nelle immagini



# Progetto count coins

## Approccio Computer Vision classico



# Progetto count coins

## Approccio Computer Vision classico

- Questo approccio utilizza tecniche algoritmiche di Computer Vision.
- E' possibile trovarne l'implementazione su [GitHub](#).
- Il codice è implementato utilizzando la libreria skimage

# Progetto count coins

## Approccio Computer Vision classico

### Step I

Trasformazione in bianco e nero

- Non abbiamo bisogno dell'informazione sui colori
- L'immagine diventa una matrice 2D di intensità



# Progetto count coins

## Approccio Computer Vision classico

### Step II

Applicazione di un filtro Gaussiano

- Riduce il rumore dell'immagine



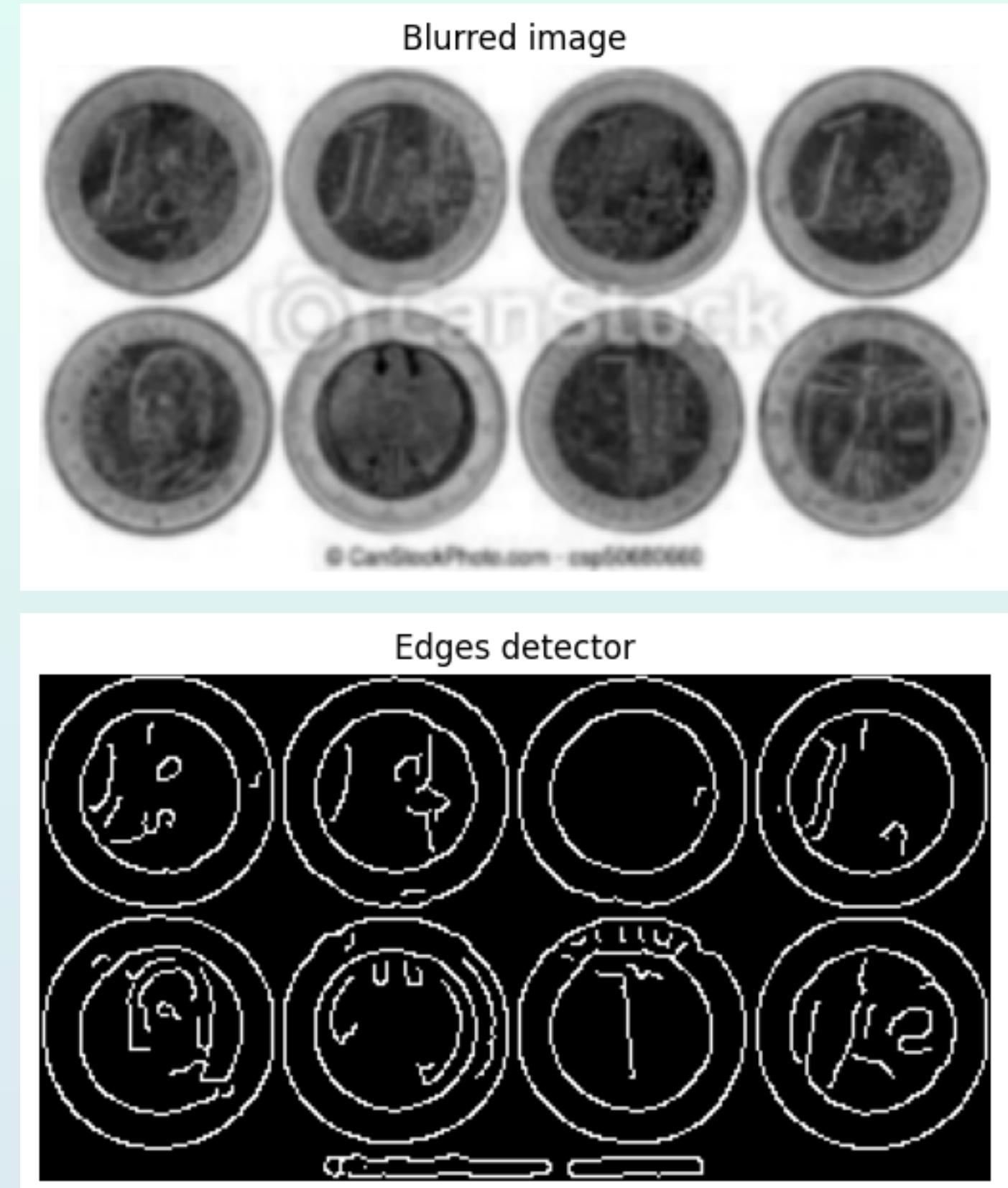
# Progetto count coins

## Approccio Computer Vision classico

### Step III

Utilizzo di un algoritmo di edge detection (Canny)

- Individuazione di contorni o i bordi degli oggetti in un'immagine

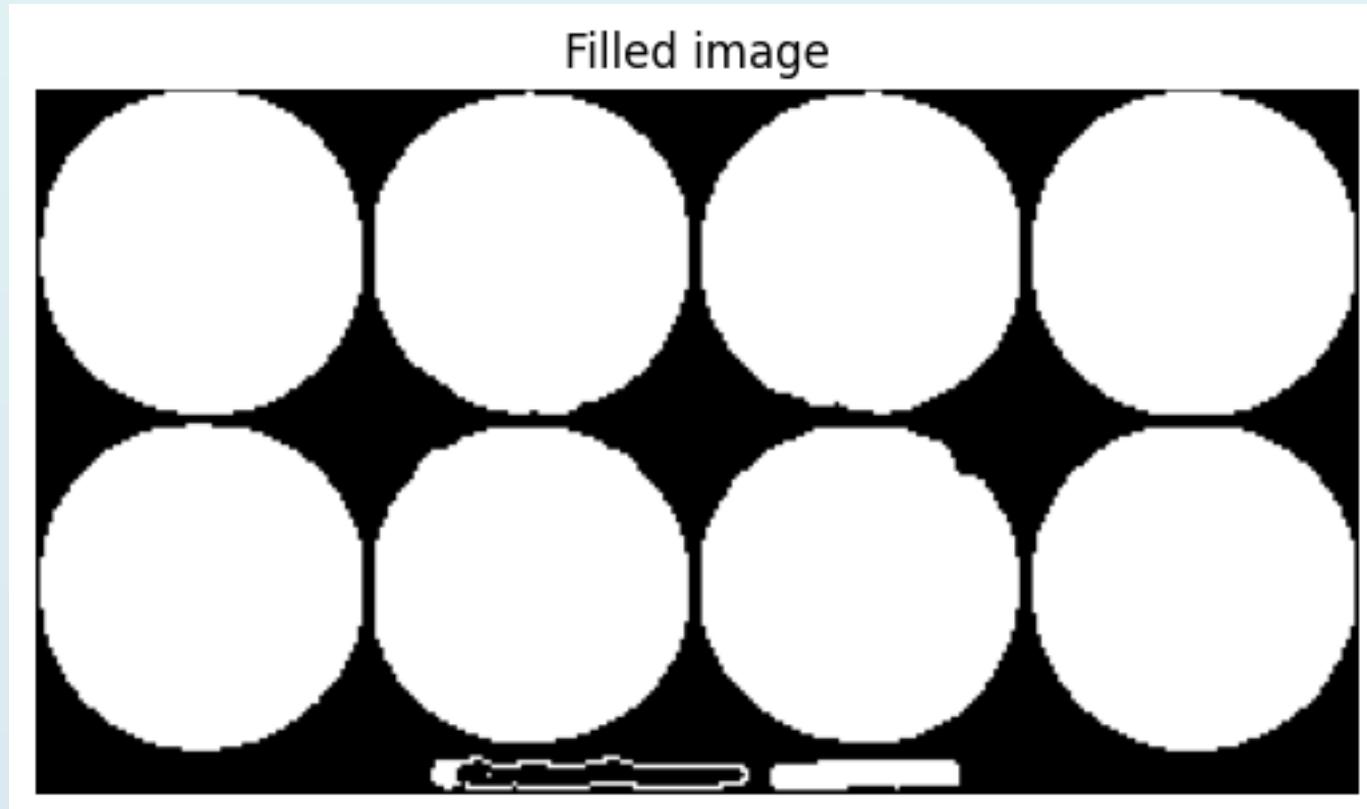
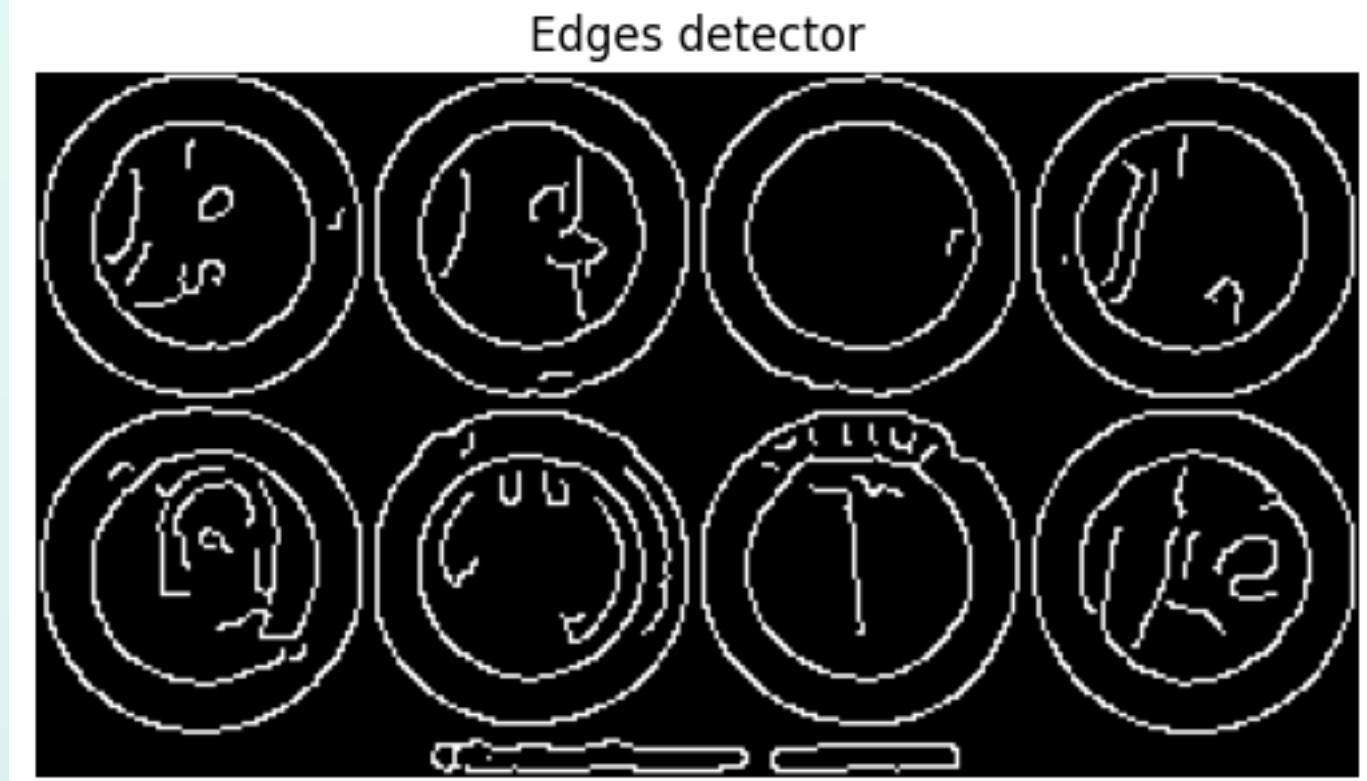


# Progetto count coins

## Approccio Computer Vision classico

### Step IV

Riempire i contorni chiusi per creare forme solide

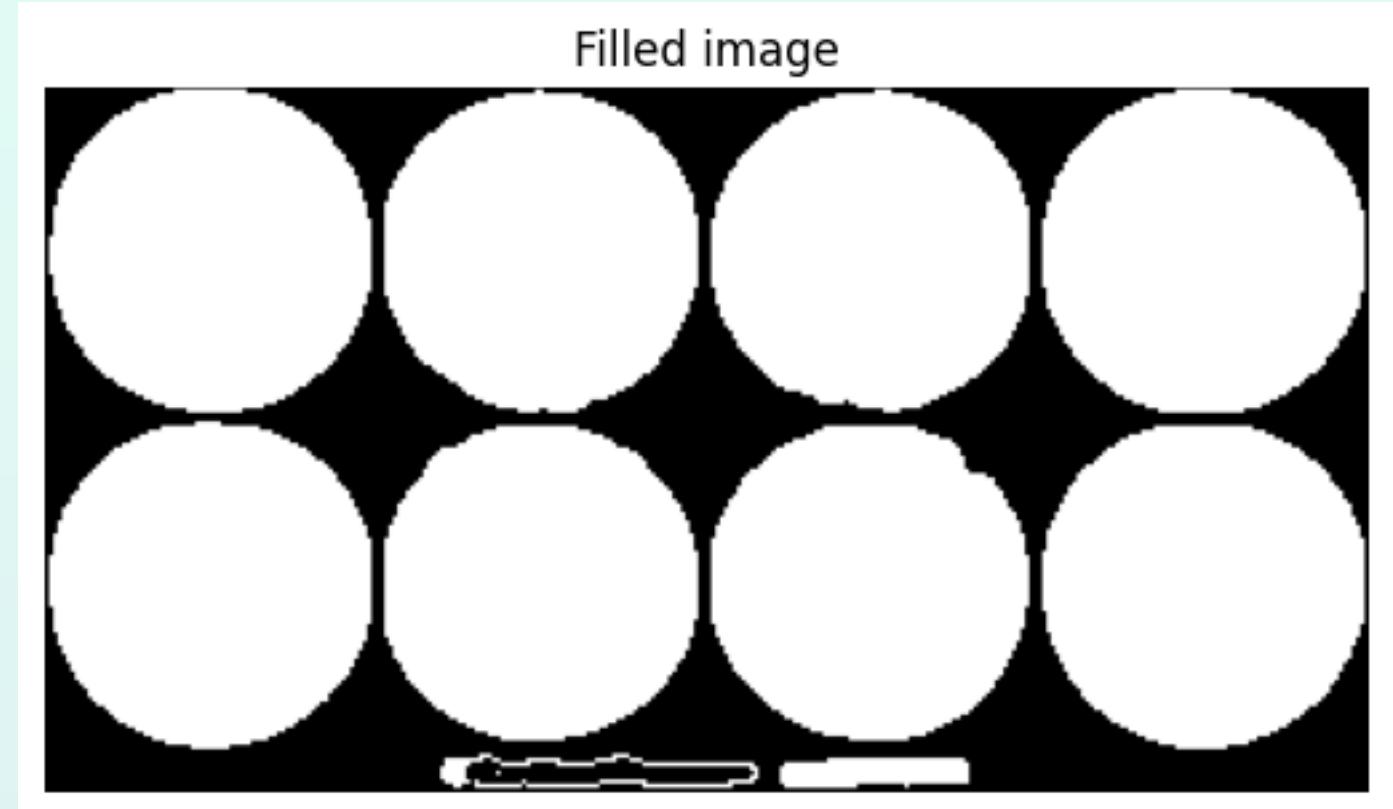


# Progetto count coins

## Approccio Computer Vision classico

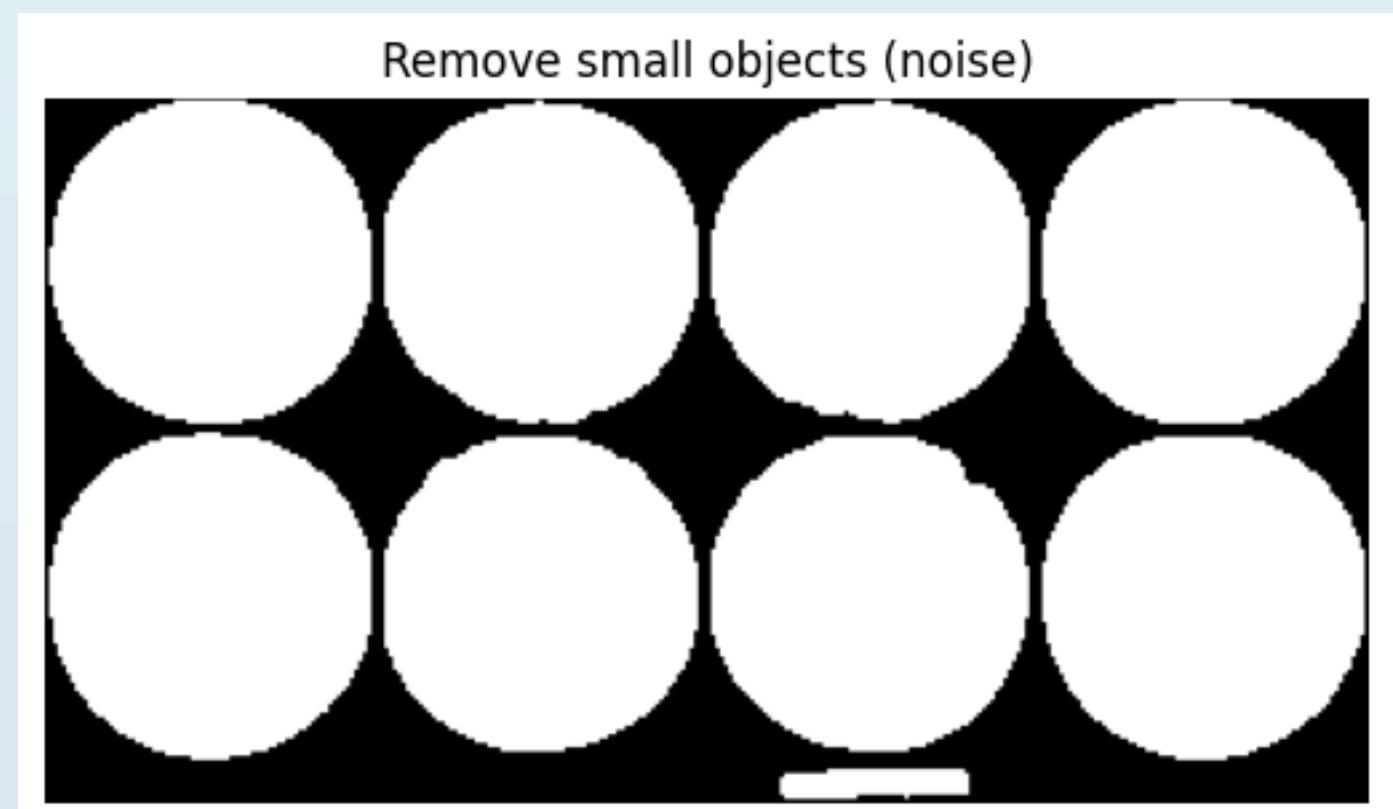
### Step V

Rimozione di oggetti piccoli



### Step VI

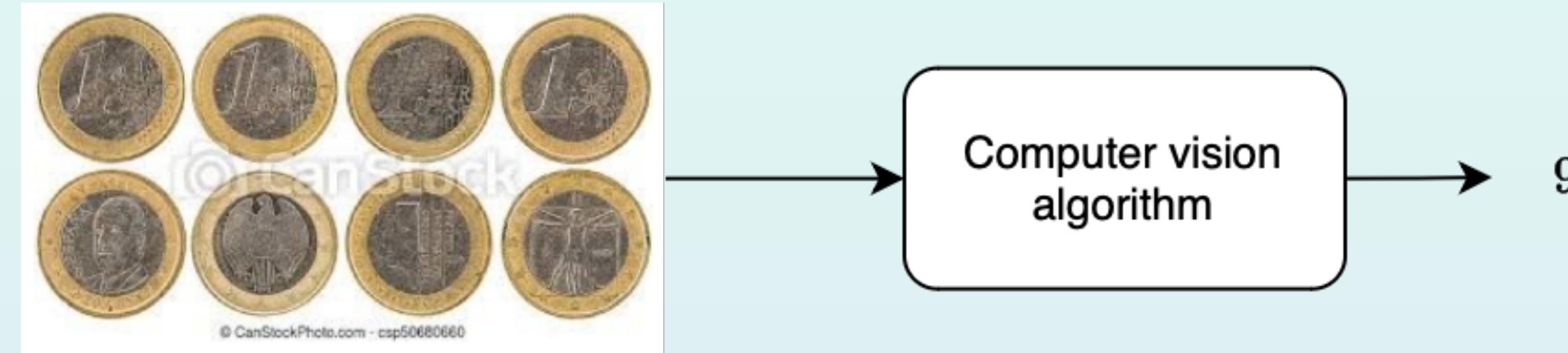
Conteggio delle forme chiuse



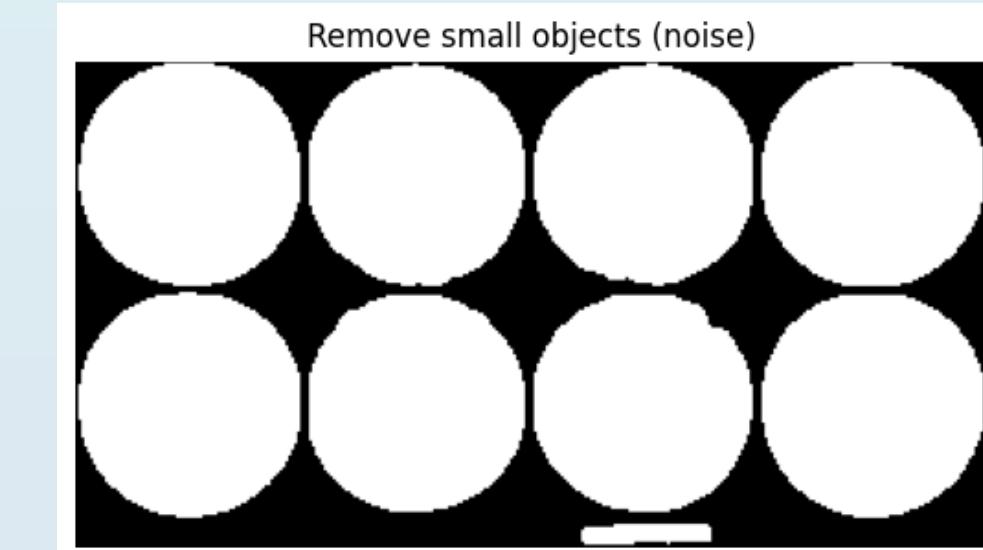
# Progetto count coins

## Approccio Computer Vision classico

L'intera sequenza di processing è incapsulata in un unico algoritmo



- E' stata trovata una regione sbagliata
- Viene predetto un risultati sbagliato!



# Progetto count coins

## Approccio Deep Learning

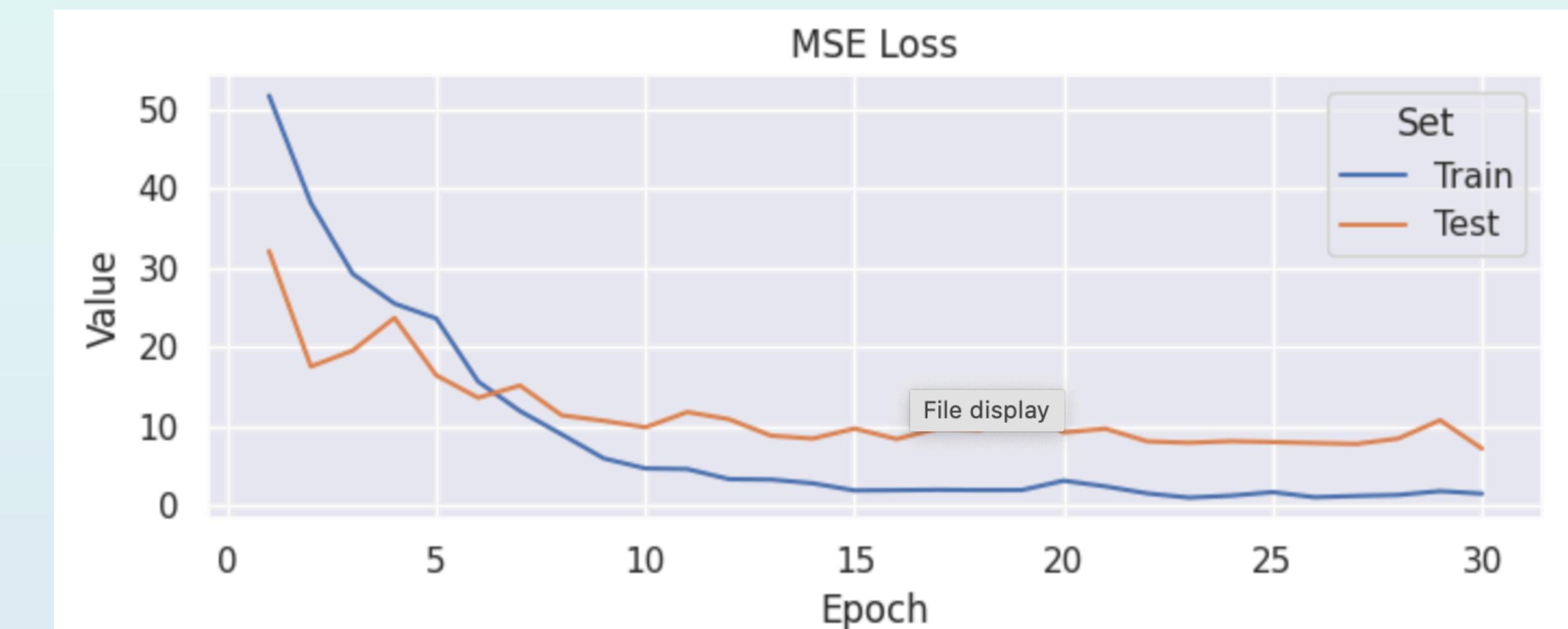
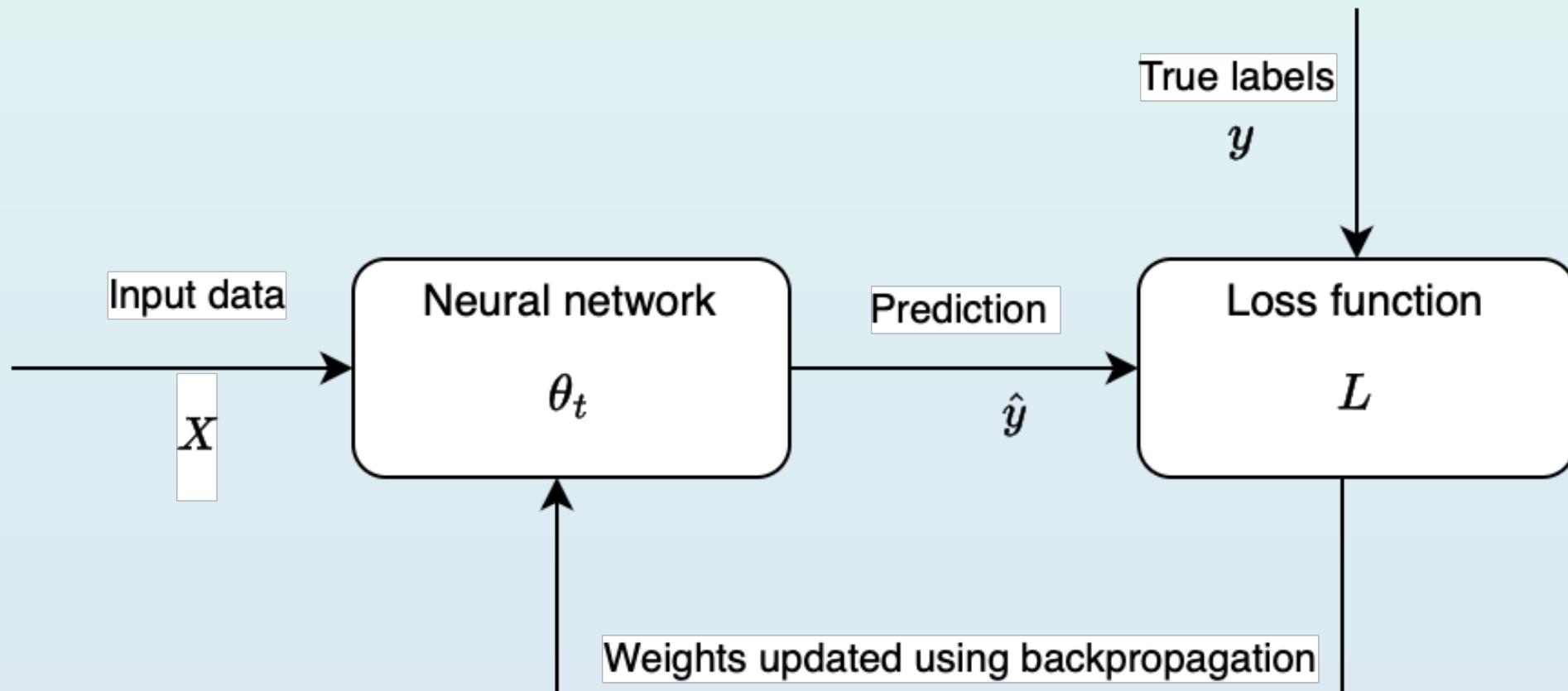
- Questo approccio utilizza tecniche di Deep Learning per la Computer Vision
- E' possibile trovarne l'implementazione su [GitHub](#).
- Il codice è implementato utilizzando la libreria PyTorch

# Progetto count coins

## Approccio Deep Learning

Utilizzo della Loss Mean squared error

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



# Progetto count coins

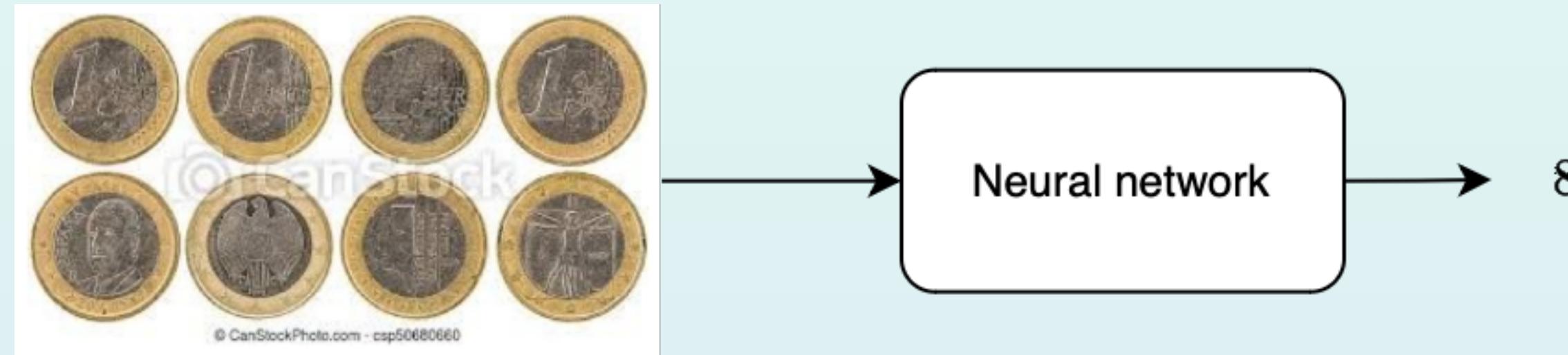
## Approccio Deep Learning

- Questo approccio utilizza tecniche di Deep Learning
- E' possibile trovarne l'implementazione su [GitHub](#).
- Il codice è implementato utilizzando la libreria PyTorch

# Progetto count coins

## Approccio Deep Learning

Possiamo utilizzare la rete neurale addestrata per eseguire predizioni



- In questo caso, la predizione è corretta!
- Non sappiamo come il modello ragiona (black-box)



## **Sezione 3**

# **Organizzazione di un progetto di ML**

# Organizzazione di un progetto di ML

## Cookiecutter Data science template

*A logical, flexible, and reasonably standardised project structure for doing and sharing data science work*

Link

```
LICENSE           <- Open-source license if one is chosen
Makefile          <- Makefile with convenience commands like `make data` or `make train`
README.md         <- The top-level README for developers using this project.

data              |
|   external      <- Data from third party sources.
|   interim       <- Intermediate data that has been transformed.
|   processed     <- The final, canonical data sets for modeling.
|   raw            <- The original, immutable data dump.

docs              <- A default mkdocs project; see www.mkdocs.org for details

models             <- Trained and serialized models, model predictions, or model summaries

notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
                      the creator's initials, and a short '-' delimited description, e.g.
                      '1.0-jqp-initial-data-exploration'.

pyproject.toml    <- Project configuration file with package metadata for
                      {{ cookiecutter.module_name }} and configuration for tools like black

references         <- Data dictionaries, manuals, and all other explanatory materials.

reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
|   figures        <- Generated graphics and figures to be used in reporting

requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
                      generated with `pip freeze > requirements.txt`

setup.cfg          <- Configuration file for flake8

{{ cookiecutter.module_name }}  <- Source code for use in this project.

|   __init__.py     <- Makes {{ cookiecutter.module_name }} a Python module
|   config.py       <- Store useful variables and configuration
|   dataset.py      <- Scripts to download or generate data
|   features.py     <- Code to create features for modeling

|   modeling        |
|   |   __init__.py  <- Code to run model inference with trained models
|   |   predict.py   <- Code to train models
|   |   train.py

plots.py           <- Code to create visualizations
```



**Grazie per l'attenzione!**

Github repo

luigidamico.1010@gmail.com