

# YOLOv8

RICONOSCIMENTO  
ANIMALE

by Sharif, Mannone, Castrillo

# INTRODUZIONE ALL'IA



## INTELLIGENZA ARTIFICIALE

permette alle macchine di simulare capacità cognitive umane, come l'apprendimento, il ragionamento e il riconoscimento.

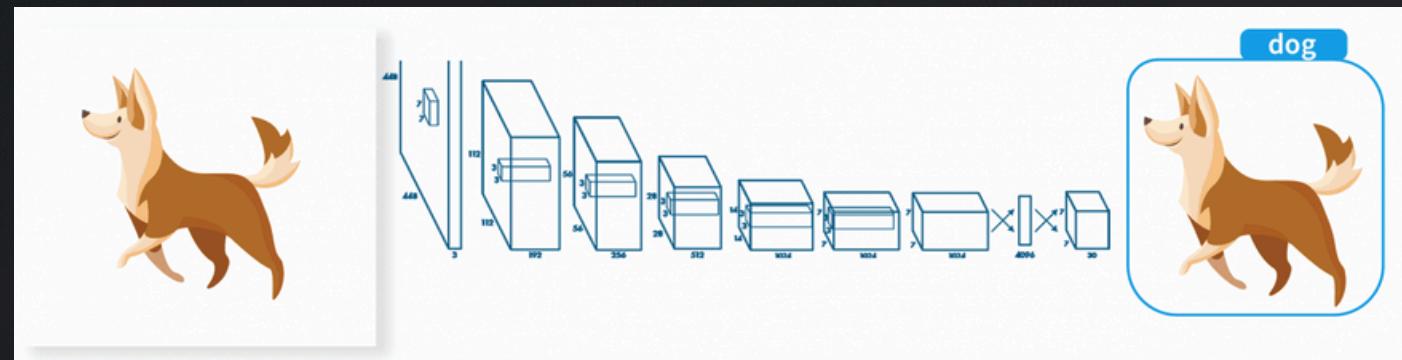


## RICONOSCIMENTO ANIMALE

Una delle applicazioni dell'IA che consente di identificare e classificare animali in immagini o video.

È importante per:

- Monitoraggio della biodiversità.
- Sorveglianza di ecosistemi e habitat.
- Applicazioni industriali, come agricoltura e sicurezza.



## YOLO (YOU ONLY LOOK ONCE)

Una tecnologia di object detection particolarmente veloce ed efficiente, progettata per riconoscere oggetti (come animali) in tempo reale. La versione 8 rappresenta l'ultima evoluzione, con miglioramenti nella precisione e nelle prestazioni.

# SETTORI DI IMPIEGO DELL'IA



## AGRICOLTURA

Monitoraggio delle colture, ottimizzazione delle risorse e identificazione di parassiti.



## SICUREZZA

Riconoscimento facciale, sorveglianza e rilevamento delle minacce.



## SALUTE

Diagnosi medica, sviluppo di farmaci, analisi di immagini radiologiche.



## AMBIENTE

Monitoraggio della fauna, gestione delle risorse naturali e lotta ai cambiamenti climatici.

# YOLO V8 OVERVIEW

## Riconoscimento in tempo reale

YOLO V8 offre capacità di riconoscimento in tempo reale per immagini e video, migliorando l'efficienza dei progetti.

## Accuratezza elevata

L'algoritmo garantisce un'elevata accuratezza nel rilevamento di oggetti, riducendo al minimo gli errori.

## Facilità di integrazione

YOLO v8 è progettato per un'integrazione semplice in vari progetti, facilitando l'uso da parte degli sviluppatori.

## Tecniche avanzate

Utilizza tecniche come la normalizzazione e il trasferimento di apprendimento per migliorare le prestazioni.

# YOLO V8 - CARATTERISTICHE E PROCESSI

1

Dividere l'immagine in una griglia

YOLO v8 analizza le immagini suddividendole in celle per un'analisi dettagliata.

2

Predire bounding boxes e probabilità

Calcola la probabilità e i confini precisi degli oggetti rilevati nella scena.

3

Eliminare sovrapposizioni con Non-Max Suppression

Rimuove le sovrapposizioni per assicurare un'identificazione accurata degli oggetti.

# YOLO V8 PER ANIMALI

## Velocità

Processa video in tempo reale, migliorando l'efficienza.

## Accuratezza

Rilevamento preciso anche in condizioni difficili.

## Versatilità

Applicabile in ricerca e conservazione, utile in vari contesti.

# FUTURO DELL'IA E YOLO V8



## Sviluppo di modelli accurati

Modelli più precisi migliorano il riconoscimento degli animali

---



## Integrazione con droni e IOT

Tecnologie cambiate offrono nuove opportunità di monitoraggio

---



## Attenzione a etica e sostenibilità

Importanza di soluzioni etiche per la conservazione della fauna

lion

habitat: Savane  
carnivore: Carnivoro  
comportamento: Cacciatore



```

import cv2
from ultralytics import YOLO

# Configurazione
MODEL_PATH = '/Users/raja/runs/detect/train2/weights/best.pt'
VIDEO_SOURCE = 0 # Usa 0/1 per l'iphone
CONFIDENCE_THRESHOLD = 0.4

# Dizionario delle caratteristiche degli animali
animal_characteristics = {
    "butterfly": {"habitat": "Foreste, giardini", "alimentazione": "Nettare", "comportamento": "Impollina"}, 
    "cat": {"habitat": "Domestico", "alimentazione": "Carnivoro", "comportamento": "Curioso"}, 
    "crocodile": {"habitat": "Fiumi", "alimentazione": "Caricamento modello"}, 
    "deer": {"habitat": "Foreste", "alimentazione": "Altri animali"}, 
    "dog": {"habitat": "Domestico", "alimentazione": "Carne"}, 
    "elephant": {"habitat": "Savane", "alimentazione": "Altri animali"}, 
    "frog": {"habitat": "Zone umide", "alimentazione": "Insetti"}, 
    "giraffe": {"habitat": "Savane", "alimentazione": "Piante"}, 
    "goat": {"habitat": "Montagne", "alimentazione": "Piante"}, 
    "hippo": {"habitat": "Fiumi", "alimentazione": "Altri animali"}, 
    "kangaroo": {"habitat": "Australia", "alimentazione": "Piante"}, 
    "lion": {"habitat": "Savane", "alimentazione": "Altri animali"}, 
    "parrot": {"habitat": "Foreste", "alimentazione": "Frutta"}, 
    "shark": {"habitat": "Oceani", "alimentazione": "Altri animali"}, 
    "sheep": {"habitat": "Pascoli", "alimentazione": "Piante"}, 
    "spider": {"habitat": "Vari", "alimentazione": "Insetti"}, 
    "tiger": {"habitat": "Foreste", "alimentazione": "Altri animali"}, 
    "zebra": {"habitat": "Savane", "alimentazione": "Piante"}, 
    "unknown": {"habitat": "Sconosciuto", "alimentazione": "Altri animali"}
}

# Funzione per rilevare animali
def detect_animals(frame):
    results = model.predict(source=frame, save=False, conf=CONFIDENCE_THRESHOLD)[0]

    animals_detected = []
    annotated_frame = frame.copy()

    for box in results.boxes:
        cls_id = int(box.cls[0])
        cls_name = model.names.get(cls_id, "unknown")
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        confidence = box.conf[0]

        # Ottieni caratteristiche dell'animale
        characteristics = animal_characteristics.get(cls_name, animal_characteristics["unknown"])

        # Disegna il box e l'etichetta
        cv2.rectangle(annotated_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        label = f'{cls_name}'
        cv2.putText(annotated_frame, label, (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 147, 57), 2)

        # Mostra le caratteristiche sul frame
        y_offset = y1 + 15 # Posizione iniziale sotto il box
        for key, value in characteristics.items():
            text = f'{key.capitalize()}: {value}'
            cv2.putText(annotated_frame, text, (x1, y_offset),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 1)
            y_offset += 15 # Sposta il testo in basso

        animals_detected.append({
            "name": cls_name,
            "confidence": confidence,
            "characteristics": characteristics
        })

    return annotated_frame, animals_detected

```

```

# Funzione principale per processare il video
def process_video(source):
    cap = cv2.VideoCapture(source)
    if not cap.isOpened():
        print("Errore nell'apertura della sorgente video")
        return

    while cap.isOpened():
        success, frame = cap.read()
        if not success:
            print("Exiting: Unable to read from source")
            break

        frame = cv2.flip(frame, 1)
        frame_with_detections, detected = detect_animals(frame)

        cv2.imshow("Animal Detection", frame_with_detections)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

# Esegui
process_video(VIDEO_SOURCE)

```

## animal detection

```
# Funzione per rilevare animali
def detect_animals(frame):
    results = model.predict(source=frame, save=False, conf=CONFIDENCE_THRESHOLD)[0]

    animals_detected = []
    annotated_frame = frame.copy()

    for box in results.boxes:
        cls_id = int(box.cls[0])
        cls_name = model.names.get(cls_id, "unknown")
        x1, y1, x2, y2 = map(int, box.xyxy[0])
        confidence = box.conf[0]

        # Ottieni caratteristiche dell'animale
        characteristics = animal_characteristics.get(cls_name, animal_characteristics["unknown"])

        # Disegna il box e l'etichetta
        cv2.rectangle(annotated_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        label = f'{cls_name}'
        cv2.putText(annotated_frame, label, (x1, y1 - 10),
                   cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 147, 57), 2)

        # Mostra le caratteristiche sul frame
        y_offset = y1 + 15 # Posizione iniziale sotto il box
        for key, value in characteristics.items():
            text = f'{key.capitalize()}: {value}'
            cv2.putText(annotated_frame, text, (x1, y_offset),
                       cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 1)
            y_offset += 15 # Sposta il testo in basso

        animals_detected.append({
            "name": cls_name,
            "confidence": confidence,
            "characteristics": characteristics
        })

    return annotated_frame, animals_detected
```

**fine**