AI & SUSTAINABILITY IN VET EDUCATION

Nome dell'autore/i: Sucessos Criativos, Lda | P. Porto

Titolo del WP: WP3 – AI-SAVES@ProjectHub

Riferimento dell'attività: 3.2 – Modello di progetto per AI-SAVES@ProjectHub

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Indice dei contenuti

1. Titolo del progetto: 2

2. Categoria del progetto: 2

3. Descrizione del progetto: 2

5. Pubblico target: 2

6. Conoscenze preliminari: 2

7. Obiettivi del progetto: 3

8. Tempistica del progetto: 3

9. Risorse fornite: 3

10. Strumenti e tecnologie utilizzati: 4

11. Piano di implementazione passo dopo passo: 6

12. Criteri di valutazione: 13

13. Lingue disponibili: 13

14. Collaboratori: 13

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

1. **Titolo del progetto:**

Sistema di segmentazione dei rifiuti in tempo reale con Raspberry Pi e YOLOv11

---

2. **Categoria del progetto:**
   - IA nell'IoT

---

3. **Descrizione del progetto:**

Questo progetto sviluppa un sistema avanzato di visione artificiale utilizzando un Raspberry Pi e la rete neurale YOLOv11 per rilevare e segmentare oggetti di rifiuti in tempo reale tramite il feed della fotocamera. Il sistema fornisce maschere di segmentazione precise a livello di pixel con sovrapposizioni visive, rendendolo ideale per applicazioni nella gestione dei rifiuti, nel monitoraggio ambientale e nell'automazione del riciclo. Gli studenti impareranno a implementare modelli di deep learning su dispositivi edge, contribuendo alla sostenibilità ambientale attraverso il rilevamento intelligente dei rifiuti.

---

4. **Parole chiave:**
   Visione artificiale, YOLOv11, Segmentazione, Raspberry Pi, Monitoraggio ambientale, Gestione dei rifiuti, Deep learning, Elaborazione in tempo reale, Edge AI

---

5. **Pubblico target:**
   Sia docenti che studenti

---

6. **Conoscenze preliminari:**
   - Programmazione Python a livello intermedio (classi, funzioni, gestione dei file)
   - Comprensione di base dei concetti di visione artificiale
   - Familiarità con reti neurali e fondamenti di deep learning
   - Esperienza con Raspberry Pi GPIO e modulo fotocamera
   - Conoscenza di base della libreria OpenCV
   - Comprensione dei concetti di rilevamento e segmentazione degli oggetti
   - Operazioni di base da riga di comando Linux

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

7. **Obiettivi del progetto:**
   - Introdurre gli studenti alla visione artificiale avanzata e al deep learning su dispositivi edge
   - Insegnare la segmentazione degli oggetti in tempo reale utilizzando modelli YOLO all'avanguardia
   - Sviluppare competenze pratiche nel deploy e nell'ottimizzazione dei modelli AI
   - Sensibilizzare sulle sfide ambientali e sulle soluzioni basate sull'AI
   - Dimostrare tecniche di segmentazione precise a livello di pixel per il rilevamento accurato degli oggetti
   - Incoraggiare lo sviluppo di soluzioni tecnologiche sostenibili
   - Costruire competenze nei sistemi AI basati su fotocamera e nell'elaborazione in tempo reale

8. **Tempistica del progetto:**

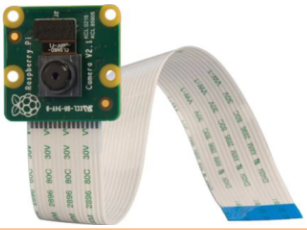
9. **Risorse fornite:**
   - Raspberry Pi 5
   - Modulo fotocamera Raspberry Pi v2 o v3
   - Tastiera per il controllo del sistema
   - Modelli di codice Python e librerie
   - Notebook Jupyter per l'addestramento YOLOv11
   - Dataset di esempio di rifiuti per il training (da Kaggle)
   - Guida passo-passo per l'implementazione
   - Documentazione per la risoluzione dei problemi

10. **Strumenti e tecnologie utilizzati:**

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

10.1 Hardware:

10 MATERIAL	
Raspberry Pi 5	
27W USB Type-C Power Supply for Raspberry Pi 5	
Raspberry Pi Camera Board V2 8MP 1080p (IM8)	
Official FPC camera cable for Raspberry Pi 5 200mm (IM11)	

10.2 **Software**:

- Sistema operativo: Raspberry Pi OS
- Linguaggio di programmazione: Python 3.8+
- Ambiente di sviluppo: Thonny IDE o VS Code
- Piattaforma per l'addestramento del modello: Google Colab con accelerazione GPU
- Framework AI: YOLOv11 (You Only Look Once v11)
- Formato del modello: PyTorch (.pt)

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Libraries: time, cv2, YOLO, PiCamera2, os, kagglehub, ultralytics.YOLO, shutil, yaml, pathlib.Path, cv2, numpy, sklearn, glob, matplotlib.pyplot, seaborn, PIL.Image, PIL.ImageDraw, PIL.ImageFont, pandas, json, datetime, torch, tqdm

---

11. Piano di implementazione passo dopo passo:

11.1 Configurazione hardware

Installazione del modulo fotocamera

1. Spegnere completamente il Raspberry Pi
2. Collegare il modulo fotocamera alla porta CSI
3. Fissare il collegamento e assicurarsi che sia correttamente inserito
4. Accendere il Raspberry Pi e abilitare l'interfaccia della fotocamera

Verifica dell'hardware

```
# Test camera detection
vcgencmd get_camera
# Expected output: supported=1 detected=1

# Test camera functionality


libcamera-hello --timeout 5000
```

11.2 Addestramento del modello (Google Colab)

Questa sezione descrive il processo di addestramento del modello di segmentazione dei rifiuti basato su YOLOv11 utilizzando Google Colab con accelerazione GPU. L'obiettivo è sviluppare un modello leggero ottimizzato per l'inferenza in tempo reale su un dispositivo Raspberry Pi.

Configurazione dell'ambiente

- Aprire Google Colab
- Abilitare la GPU:
  Runtime → Change runtime type → Hardware accelerator: GPU

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- Caricare e aprire il notebook `train_model.ipynb`
- Installare le librerie richieste e configurare i parametri di addestramento

Configurazione del dataset

- Dataset proveniente da Kaggle, contenente immagini annotate di rifiuti marini
- Le annotazioni includono maschere di segmentazione a livello di pixel e etichette di classe
- Convertito in formato compatibile con la segmentazione YOLOv11

Processo di addestramento

- **Preprocessing:** Ridimensionamento delle immagini, normalizzazione (rotazioni, ribaltamenti, variazioni di luminosità)
- **Modello:** YOLOv11n (variante per classificazione ottimizzata per dispositivi edge)
- **Esecuzione dell'addestramento**
- **Valutazione:** Effettuata automaticamente delle prestazioni dopo ogni epoca utilizzando metriche come mAP e IoU
- **Esportazione:** Modello finale salvato in formato `.pt` per il deploy su Raspberry Pi

Output

- Il modello risultante `yolov11_seg_trash.pt` è ottimizzato per l'inferenza su Raspberry Pi
- Fornisce rilevamento degli oggetti in tempo reale con maschere di segmentazione precise a livello di pixel
- Visualizzazioni e metriche possono essere consultate direttamente in Colab

11.3 Configurazione del software

Ora che l'hardware è pronto, è il momento di implementare il sistema di segmentazione dei rifiuti.

1. Creare un ambiente virtuale

```
# Create project directory
mkdir trash_segmentation_system
cd trash_segmentation_system

# Create virtual environment
python3 -m venv venv

# Activate virtual environment
      source venv/bin/activate
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

2. Installare le dipendenze

```
# Install requirements


pip install -r requirements.txt
```

3. Aggiungere il codice

- Copia il codice Python del sistema di segmentazione dei rifiuti in `main.py`
- Assicurati che il file si trovi nella directory del tuo progetto

4. Aggiungere il modello

- Copia il modello del sistema di segmentazione dei rifiuti in `model/yolov11_seg_trash.pt`
- Assicurati che il file si trovi nella directory del tuo progetto

5. Esegui il sistema

```
# Make sure virtual environment is activated
source venv/bin/activate

# Run the light trash segmentation system python
main.py
```

5. Arrestare il sistema

- Premi **Ctrl+C** per fermare il programma in modo sicuro

Comportamento del sistema

Sistema di segmentazione dei rifiuti in tempo reale con Raspberry Pi e YOLOv11.

Sequenza di avvio

- Caricamento del modello da `yolov11_seg_trash.pt`
- Inizializzazione della fotocamera (800x600 RGB888)

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- Il sistema mostra "✅ Model loaded successfully"
- Avvio del rilevamento in tempo reale

Stati operativi

- **Funzionamento normale:** acquisizione e elaborazione continua dei frame
- **Rilevamento attivo:** maschere di segmentazione colorate sovrapposte ai rifiuti rilevati
- **Nessun rilevamento:** il sistema continua il monitoraggio
- **Recupero errori:** gestione automatica dei guasti durante l'elaborazione

Feedback visivo

- **Maschere di segmentazione:** sovrapposizioni colorate che mostrano con precisione i contorni degli oggetti
- **Bounding box:** riquadri rettangolari attorno agli oggetti rilevati
- **Etichette:** tipo di oggetto e punteggio di confidenza visualizzati
- **Visualizzazione in tempo reale:** finestra OpenCV con risultati live

Controlli interattivi

- **Terminazione:** Ctrl+C per uno spegnimento sicuro
- **Pulizia delle risorse:** chiusura automatica della fotocamera e delle finestre
- **Output console:** conteggio dei rilevamenti e messaggi di stato del sistema

CODICE

```
import cv2
import numpy as np
from picamera2 import Picamera2
from ultralytics import YOLO
import time
import os

# Initialize the Picamera2
picam2 = Picamera2()
picam2.preview_configuration.main.size = (800, 600)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()

# Load the custom YOLO segmentation model
print("Loading segmentation model...")
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
model = YOLO("model/yolov11_seg_trash.pt")
print("✅ Model loaded successfully")

# Configure parameters
conf_threshold = 0.01  # Confidence threshold
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255),
          (0, 255, 255), (128, 0, 128), (255, 165, 0), (255, 192, 203), (165,
42, 42)]

print("Starting real-time detection...")

try:
    while True:
        # Capture frame-by-frame
        frame = picam2.capture_array()

        # Convert from RGB to BGR for OpenCV (if necessary)
        frame_bgr = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        img_height, img_width = frame_bgr.shape[:2]

        # Run YOLO inference
        results = model(frame_bgr, conf=conf_threshold, verbose=False)

        # Create copy for overlay
        overlay = frame_bgr.copy()

        # Process results following your successful method
        for result in results:
            # Check if there are masks and detections
            if result.masks is not None and len(result.masks) > 0:
                # Get mask data, boxes and confidences
                masks = result.masks.data.cpu().numpy()
                boxes = result.boxes.xyxy.cpu().numpy()
                confidences = result.boxes.conf.cpu().numpy()
                class_indices = result.boxes.cls.cpu().numpy()

                print(f"Detected {len(masks)} masks")

                # Process each prediction
                    for i, (mask, box, conf, cls_idx) in enumerate(zip(masks,
boxes, confidences, class_indices)):
                    if conf > conf_threshold:
                        # Resize mask to image size
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
                                mask_resized = cv2.resize(mask, (img_width,
img_height))

                        # Get color for this class
                        color = np.array(colors[int(cls_idx) % len(colors)])

                        # Create colored mask overlay
                        mask_indices = mask_resized > 0.5
                            overlay[mask_indices] = overlay[mask_indices] * 0.6 +
color * 0.4

                        # Draw bounding box
                        x1, y1, x2, y2 = box.astype(int)
                                    cv2.rectangle(overlay, (x1, y1), (x2, y2),
color.tolist(), 2)

                        # Get class name
                                    class_name = result.names[int(cls_idx)] if
hasattr(result, 'names') else f"Class {int(cls_idx)}"

                        # Add label with confidence
                        label = f"{class_name}: {conf:.2f}"
                            (text_width, text_height), _ = cv2.getTextSize(label,
cv2.FONT_HERSHEY_SIMPLEX, 0.5, 2)

                        # Background for text
                                cv2.rectangle(overlay, (x1, y1-text_height-10),
(x1+text_width, y1), color.tolist(), -1)
                                    cv2.putText(overlay, label, (x1, y1-5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

                        # Optional: draw contours for better definition
                        mask_binary = (mask_resized > 0.5).astype(np.uint8)
                                contours, _ = cv2.findContours(mask_binary,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
                        cv2.drawContours(overlay, contours, -1, color.tolist(),
2)

            else:
                # If there are detections but no masks (only bounding boxes)
                if result.boxes is not None and len(result.boxes) > 0:
                    boxes = result.boxes.xyxy.cpu().numpy()
                    confidences = result.boxes.conf.cpu().numpy()
                    class_indices = result.boxes.cls.cpu().numpy()
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
                        for box, conf, cls_idx in zip(boxes, confidences,
class_indices):
                    if conf > conf_threshold:
                        x1, y1, x2, y2 = box.astype(int)
                        color = colors[int(cls_idx) % len(colors)]

                        # Draw only bounding box if no masks available
                           cv2.rectangle(overlay, (x1, y1), (x2, y2), color,
2)

                                    class_name = result.names[int(cls_idx)] if
hasattr(result, 'names') else f"Class {int(cls_idx)}"
                        label = f"{class_name}: {conf:.2f}"
                                  cv2.putText(overlay, label, (x1, y1-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

       # Display frame with overlays
       cv2.imshow("YOLO Segmentation", overlay)

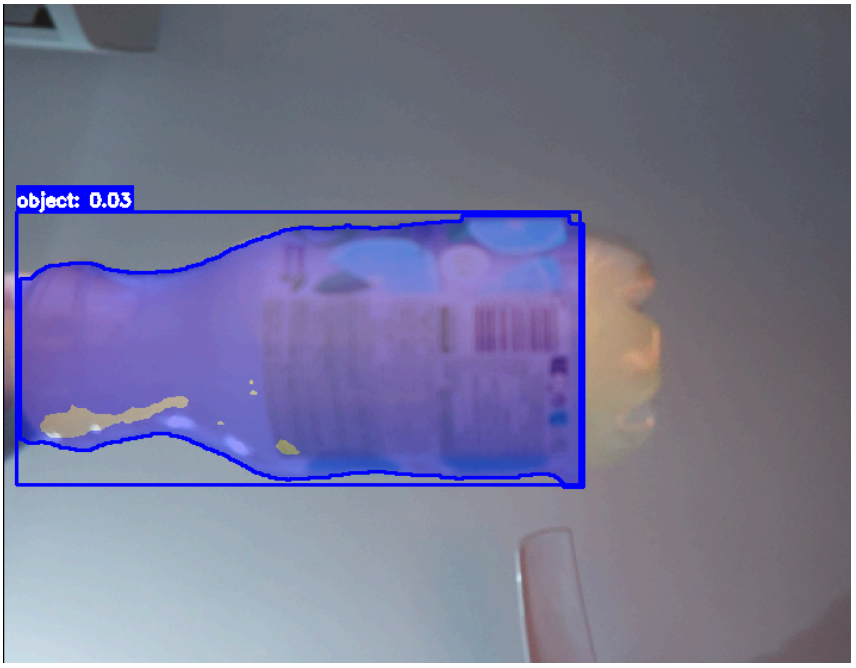

except KeyboardInterrupt:
   print("\nInterrupted by user")
except Exception as e:
   print(f"Error: {e}")

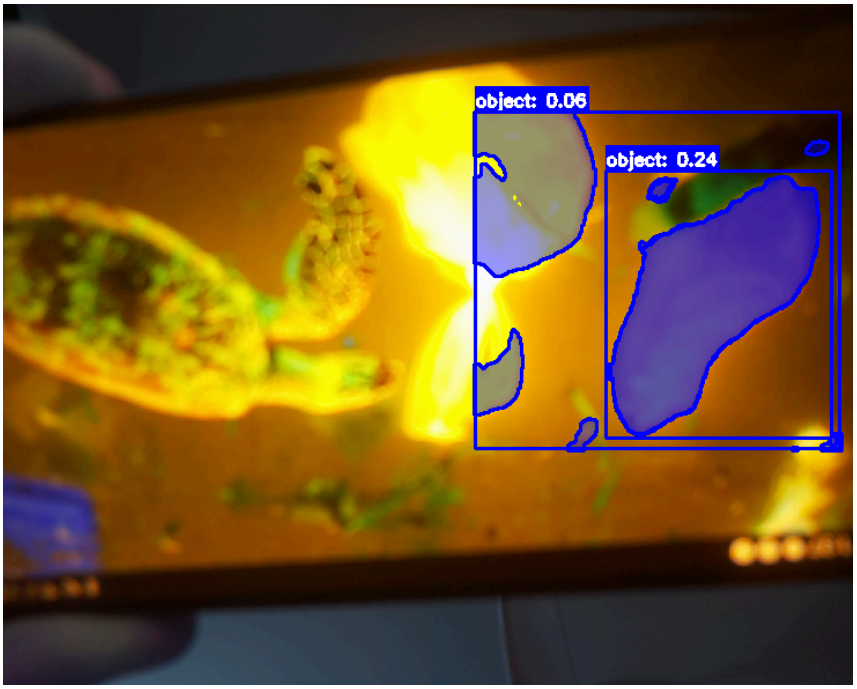
finally:
   # Clean up resources
   print("Closing application...")
   picam2.stop()
   cv2.destroyAllWindows()
   print("✅ Resources freed successfully")
```

Ora puoi eseguire il programma e vedere il risultato:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.





12. Criteri di valutazione:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- **Esecuzione dell'addestramento del modello (60%)**: Corretta esecuzione del notebook Jupyter per l'addestramento del modello di segmentazione YOLOv11 utilizzando il dataset fornito. Include la corretta configurazione degli iperparametri, un processo di training fluido senza errori, il monitoraggio della convergenza e la generazione di un modello utilizzabile pronto per il deploy.
- **Configurazione hardware (10%)**: Collegamento e configurazione di base corretti del Raspberry Pi e del modulo fotocamera, garantendo una cattura video stabile e un flusso dati continuo durante l'inferenza.
- **Funzionalità del software (20%)**: Funzionamento in tempo reale del sistema di segmentazione con rilevamento accurato e sovrapposizione delle maschere sul feed della fotocamera. Elaborazione efficiente senza crash o blocchi. Corretta gestione di errori o guasti durante l'esecuzione.
- **Comprensione del progetto e documentazione (10%)**: Spiegazioni chiare delle fasi di addestramento, dell'architettura del modello, del flusso del sistema e delle istruzioni d'uso. Capacità di interpretare i risultati del modello e risolvere problemi comuni.

---

13. Lingue disponibili:

---

14. Collaboratori:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.