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:**

Rilevamento delle mele e classificazione delle malattie con Raspberry Pi e visione artificiale

---

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

---

3. **Descrizione del progetto:**

Questo progetto sviluppa un sistema intelligente di monitoraggio agricolo utilizzando un Raspberry Pi e la rete neurale YOLOv11 per rilevare le mele e classificare in tempo reale il loro stato di salute. Il sistema sfrutta la visione artificiale per identificare quattro condizioni delle mele: Sane, Macchie (Blotch), Marciume (Rot) e Crosta (Scab). Attraverso l'elaborazione del feed video in tempo reale, il sistema fornisce un feedback visivo immediato con punteggi di fiducia, risultando prezioso per l'agricoltura di precisione, il controllo qualità e la valutazione automatizzata delle colture.

---

4. **Parole chiave:**
   Rilevamento delle mele, Classificazione delle malattie, YOLOv11, Visione artificiale, Raspberry Pi, AI agricola, Elaborazione in tempo reale, Monitoraggio delle colture, Controllo qualità

---

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

---

6. **Conoscenze preliminari:**
   - Comprensione di base della programmazione Python (funzioni, cicli, programmazione orientata agli oggetti)
   - Introduzione ai concetti di visione artificiale e elaborazione delle immagini
   - Conoscenza della configurazione del Raspberry Pi e del controllo GPIO
   - Comprensione di base delle reti neurali e dei concetti di machine learning
   - Familiarità con moduli fotocamera e elaborazione video
   - Conoscenza delle malattie delle piante agricole (opzionale ma utile)

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 ad applicazioni avanzate di AI in agricoltura
- Insegnare tecniche di visione artificiale e rilevamento oggetti in tempo reale
- Sviluppare competenze nell'implementazione e nel deploy di reti neurali
- Sensibilizzare sull'agricoltura di precisione e sulla coltivazione sostenibile
- Apprendere i processi di training, validazione e deploy dei modelli
- Creare soluzioni pratiche per il controllo qualità e il monitoraggio delle colture
- Comprendere l'integrazione dell'AI con dispositivi IoT per applicazioni intelligenti

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 mele per il training (da Kaggle)
- Guida passo-passo per l'implementazione
- Documentazione per la risoluzione dei problemi

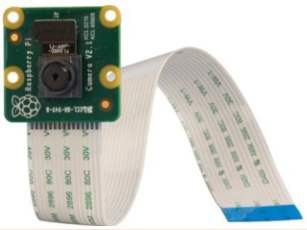
10. Strumenti e tecnologie utilizzati:

10.1 Hardware:

10	MATERIALE	

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.

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)

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

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.

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 di un modello di classificazione delle mele basato su YOLOv11 utilizzando Google Colab con accelerazione GPU. L'obiettivo è sviluppare un modello leggero in grado di classificare le mele in quattro categorie distinte in tempo reale su un Raspberry Pi.

Configurazione dell'ambiente

- Aprire Google Colab
- Abilitare la GPU:
  Runtime → Change runtime type → Hardware accelerator: GPU
- Caricare e aprire il notebook `train_model.ipynb`
- Installare le librerie richieste e configurare i parametri di addestramento

Configurazione del dataset

- Il dataset è costituito da immagini etichettate di mele in quattro categorie:
  - Sane
  - Marciume (Apple Rot)
  - Crosta (Scab)
  - Macchie (Blotch)
- Le immagini provengono da dataset open (ad es. Kaggle) e sono organizzate in cartelle specifiche per classe

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.

- Convertite in formato compatibile con YOLOv11 per classificazione, con file `.txt` di etichetta per ogni immagine

Processo di addestramento

- **Preprocessing:** Ridimensionamento delle immagini, normalizzazione e eventuale data augmentation (rotazioni, ribaltamenti, variazioni di luminosità)
- **Modello:** YOLOv11n (variante per classificazione ottimizzata per dispositivi edge)
- **Esecuzione dell'addestramento**
- **Valutazione:** Effettuata automaticamente ad ogni epoca, monitorando accuratezza di classificazione, loss e matrice di confusione
- **Esportazione:** Modello finale salvato in formato `.pt` per il deploy su Raspberry Pi

Output

- Il modello risultante `yolov11_apple.pt` è ottimizzato per l'inferenza su Raspberry Pi
- Capace di classificare immagini di mele nelle quattro categorie di salute definite in tempo reale

Visualizzazioni dei risultati e metriche possono essere consultate direttamente su Colab

11.3 Configurazione del software

Ora che l'hardware è pronto, è il momento di implementare il sistema software per il rilevamento e la classificazione delle mele in tempo reale sul Raspberry Pi.

1. Creare un ambiente virtuale

```
# Create project directory
mkdir apple_detection_system
cd apple_detection_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 classificazione delle mele in `main.py`
- Assicurati che il file si trovi nella directory del tuo progetto

4. Aggiungere il modello

- Copia il modello del sistema di classificazione delle mele in `model/yolov11_apple.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 apple detection system
python main.py
```

5. Arrestare il sistema

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

Comportamento del sistema

Il sistema di rilevamento delle mele funziona con schemi intelligenti progettati per un monitoraggio agricolo affidabile.

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.

Flusso logico del rilevamento

Inizializzazione

- **Condizione:** Avvio del sistema
- **Comportamento:** Attivazione anteprima fotocamera
- **Feedback visivo:** Video pulito

Caricamento del modello

- **Condizione:** Modello AI pronto
- **Comportamento:** Rete neurale YOLOv11 attiva
- **Feedback visivo:** Conferma dello stato

Scansione attiva

- **Condizione:** Elaborazione dei frame
- **Comportamento:** Inferenza in tempo reale in esecuzione
- **Feedback visivo:** Sovrapposizione live delle rilevazioni

Mela rilevata

- **Condizione:** Confidenza ≥ 70%
- **Comportamento:** Migliore rilevazione visualizzata
- **Feedback visivo:** Testo rosso con classe + punteggio di confidenza

Bassa confidenza

- **Condizione:** Confidenza < 70%
- **Comportamento:** Rilevazione filtrata
- **Feedback visivo:** Nessuna sovrapposizione mostrata

Più mele rilevate

- **Condizione:** Diverse rilevazioni
- **Comportamento:** Selezionata la rilevazione con maggiore confidenza
- **Feedback visivo:** Visualizzato solo il miglior risultato

Nessuna rilevazione

- **Condizione:** Nessuna mela trovata
- **Comportamento:** Elaborazione pulita

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.

- **Feedback visivo:** Video chiaro

Stati di rilevamento e risposte

🍎 **Rilevamento mela sana**

- **Trigger:** Classe APPLE_Healthy_Apple rilevata
- **Display:** Testo verde "APPLE_Healthy_Apple XX.XX%"
- **Significato:** Frutto normale, commerciabile
- **Valore agricolo:** Conferma controllo qualità

🟤 **Stati di rilevamento malattie**

Rilevamento Blotch

- **Trigger:** Classe APPLE_Blotch_Apple rilevata
- **Significato:** Infezione fungina rilevata
- **Azione consigliata:** Isolamento e valutazione trattamento

Rilevamento Rot

- **Trigger:** Classe APPLE_Rot_Apple rilevata
- **Significato:** Decadimento avanzato rilevato
- **Azione consigliata:** Rimozione immediata dal lotto

Rilevamento Scab

- **Trigger:** Classe APPLE_Scab_Apple rilevata
- **Significato:** Rilevata malattia della crosta della mela
- **Azione consigliata:** Separazione per lavorazione

Comportamento in base alla soglia di confidenza

Range 90-100%

- **Risposta sistema:** Rilevazione alta confidenza
- **Visualizzazione:** Testo in grassetto e colore forte
- **Affidabilità:** Eccellente – Classificazione definitiva

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.

Range 80-89%

- **Risposta sistema:** Buona confidenza
- **Visualizzazione:** Standard
- **Affidabilità:** Buona – Classificazione affidabile

Range 70-79%

- **Risposta sistema:** Confidenza minima accettabile
- **Visualizzazione:** Testo più chiaro
- **Affidabilità:** Accettabile – Monitorare la coerenza

Sotto 70%

- **Risposta sistema:** Rilevazione ignorata
- **Visualizzazione:** Nessun display
- **Affidabilità:** Non affidabile – Ignorata per evitare falsi positivi

Pipeline di elaborazione dei frame

1. Acquisizione immagine (5-10 FPS)
2. Inferenza AI (~100-500 ms per frame)
3. Filtraggio per confidenza (soglia 70%)
4. Selezione della migliore rilevazione (massima confidenza)
5. Rendering della sovrapposizione visiva (testo in tempo reale)
6. Aggiornamento display (feed video continuo)

CODICE

```
import cv2
from picamera2 import Picamera2
from ultralytics import YOLO
import time

# Initialize the camera
picam2 = Picamera2()
picam2.preview_configuration.main.size = (800, 600)
```

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.

```
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()

# Load the YOLO model

model = YOLO("model/yolov11_apple.pt")

# Detection loop
while True:
    # Capture frame
    frame = picam2.capture_array()

    # Run YOLO inference on the frame
    results = model(frame)

    # Create a copy of the frame for annotation
    annotated_frame = frame.copy()

    # Initialize variables to track the best detection
    best_label_text = ""

    # Process detection results
    for result in results:
        highest_conf = 0
        best_label = ""

        # Iterate through all detected boxes
        for box in result.boxes:
            # Get confidence score as percentage
            confidence = float(box.conf[0]) * 100

              # Only consider detections with confidence > 70% and higher than
current best
            if confidence > 70 and confidence > highest_conf:
                cls_id = int(box.cls[0])  # Get class ID
                best_label = result.names[cls_id]  # Get class name
                highest_conf = confidence  # Update highest confidence

        # Format the best detection text if found
        if best_label:
            best_label_text = f"{best_label} {highest_conf:.2f}%"
```

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.

```
    # Display the detection text on the frame
    if best_label_text:
        cv2.putText(
            annotated_frame,
            best_label_text,
            (30, 60),  # Position (x, y)
            cv2.FONT_HERSHEY_SIMPLEX,  # Font type
            1.5,  # Font scale
            (0, 0, 255),  # Color (BGR format - red)
            3,  # Thickness
            cv2.LINE_AA  # Anti-aliasing
        )

    # Display the frame in a window
    cv2.imshow("Camera", annotated_frame)

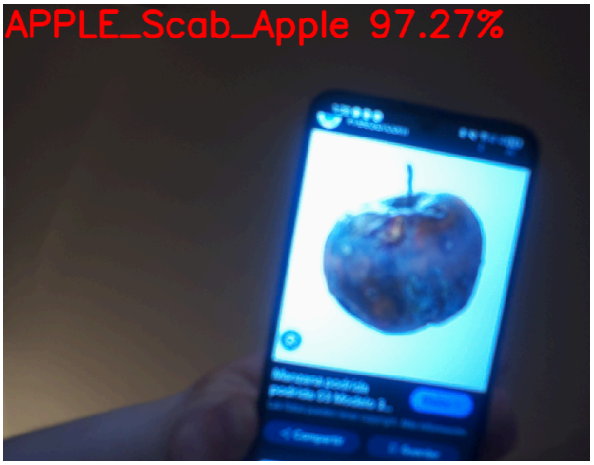
# Clean up resources
cv2.destroyAllWindows()
```

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:

- **Addestramento del modello e prestazioni (60%)**: Corretta esecuzione del notebook Jupyter fornito per l'addestramento del modello, configurazione appropriata dei parametri di training, convergenza del modello, valutazione dell'accuratezza e delle soglie di confidenza, e capacità di distribuire ed eseguire il modello YOLOv11 addestrato sul Raspberry Pi con inferenza in tempo reale.
- **Configurazione hardware (10%)**: Installazione e collegamento corretti del Raspberry Pi e del modulo fotocamera, garantendo un'acquisizione stabile e continua del flusso video durante il funzionamento del sistema.
- **Funzionalità del software (20%)**: Corretta esecuzione del codice di rilevamento senza crash, elaborazione in tempo reale dei frame video, classificazione accurata delle condizioni delle mele, filtraggio delle rilevazioni a bassa confidenza e feedback visivo chiaro.
- **Documentazione e comprensione del progetto (10%)**: Spiegazione chiara dell'architettura del sistema, della metodologia di addestramento, istruzioni

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.

passo-passo per la configurazione e l'esecuzione del sistema, e interpretazione dei risultati di rilevamento.

---

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.