Caso di Studio:
Parking Spot Recognition System

Corso di Sistemi ad Agenti
Università degli Studi di Bari

Gianmarco Rutigliano
Flavio Valerio

September 13, 2023

1 Introduzione

Il progetto affronta un task di computer vision, nell'ambito della object detection, ovvero il rilevamento di auto e parcheggi liberi all'interno di un parcheggio. il progetto nasce dal desiderio esemplificare l'operazione di parcheggio, rendendo segnalabili i parcheggi liberi. Il progetto si pone di:

- Disegnare bounding boxes attorno ai parcheggi liberi
- Oscurare, nel rispetto della privacy, le auto parcheggiate

Il modello è stato realizzato attraverso l'utilizzo della yolo family, nello specifico della versione yolov8. [1] Per un più facile utilizzo da parte dell'utente, è stata realizzata una interfaccia grafica completa ma intuitiva con cui è possibile accedere alle funzioni precedentemente esposte e permette in aggiunta di esportare l'immagine così modificata e ottenere in tempo reale un conteggio dei posti vuoti riconosciuti.

2 Requisiti per l'utilizzo del sistema di riconoscimento

L'utilizzo personale del modello necessita dell'installazione delle seguenti librerie:

- Ultralytics viene utilizzata per le operazioni concernenti il modello yolov8;
- Opencv-python viene utilizzata per le funzioni di image processing e computer vision;
- Pillow viene utilizzata per una gestione fluida dei file di immagini;

Questi requisiti sono raccolti nel file requirements.txt. Il comando bash **pip install -r requirements.txt** permette l'installazione immediata di ogni libreria sopracitata.

3 Dataset scelto

Il dataset utilizzato è il PKLot Dataset, di Roboflow, condiviso dall' Universidade Federal do Parana.[2]

Il dataset contiene 12416 immagini di parcheggi, sulle quali è già stato effettuato il task di data annotation: per ogni immagine sono stati individuati i posteggi vuoti e i posteggi occupati, e per ciascuno sono stati compilati lo status di occupato oppure libero (la classe del task di image recognition) e le coordinate della bounding box che isola il posteggio. È necessario segnalare che

[1] La documentazione è reperibile al seguente indirizzo: https://docs.ultralytics.com/

[2] Il dataset, riportato comunque nel progetto, è reperibile al seguente indirizzo: https://public.roboflow.com/object-detection/pklot

non tutti i parcheggi e le auto sono state annotate, bensì solo la maggior parte: è quindi auspicabile in futuro il completamento del task di data annotation.

4 Addestramento del modello

Il training del modello è stato effettuato con la libreria PyTorch. PyTorch mette a disposizione funzioni per la costruzione ed elaborazione di tensori con la specifica possibilità di utilizzare le potenti risorse delle GPU per rendere le operazioni più efficienti.[3]

Siccome la funzione di training offerta da ultralytics permette di stabilire il numero di epoche su cui effettuare il training, sono stati effettuati diversi tentativi per scegliere il modello più appropriato. Analizzando le misure di precision e recall dei modelli risultanti, è possibile notare che i valori si stabilizzano intorno alle 10 epoche, quindi la scelta finale è ricaduta sul modello addestrato per 10 epoche. Le stesse misure di precision e recall raggiungono valori di 99% quindi il modello si comporta molto bene sul dataset.

È doveroso segnalare che il modello non raggiunge risultati degni di nota su immagini che si discostano di molto da quelle presenti nel dataset. Questo è un fenomeno abbastanza comune nei task di image recognition, ma nondimeno degno di nota.

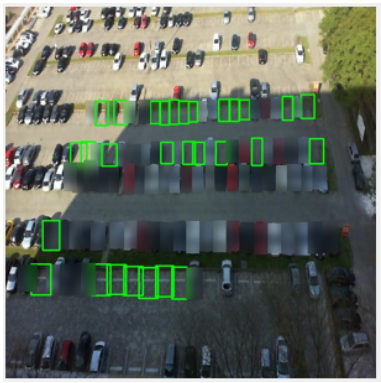

Figure 1: Esempio di Output del Modello

[3]La libreria PyTorch non è stata inserita nei requisiti perché non è necessaria per l'utilizzo dell'applicazione principale. La versione utilizzata è la 2.0.1, quindi chi fosse interessato a usufruire dei file inerenti il training deve aggiungere al file requirements.txt la riga **torch =2.0.1** prima di installarli.

5 Utilizzo dell'applicazione

Il modello per riconoscimento di auto e parcheggi è accessibile agli utenti mediante la seguente interfaccia grafica, realizzata con l'ausilio della libreria python tkinter. La GUI presenta una sidebar sulla destra con ogni funzione utilizzabile

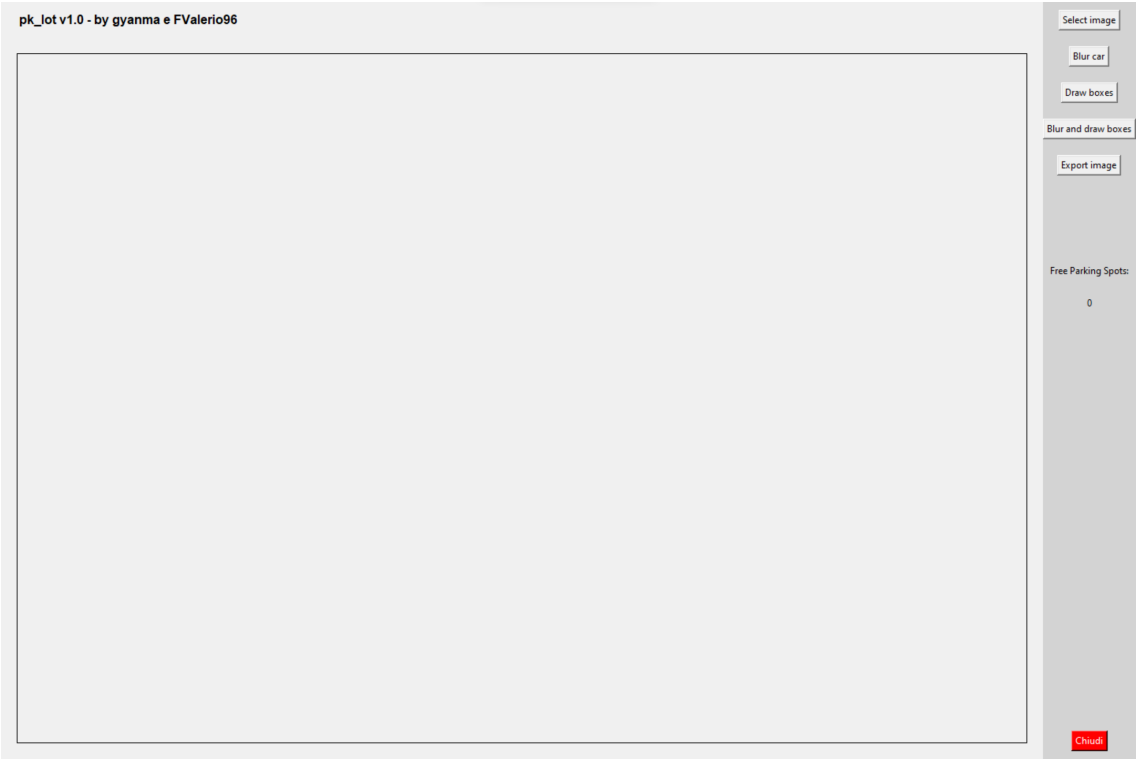

Figure 2: Interfaccia Utente

- Select image consente di importare un'immagine;
- Blur car effettua l'oscuramento delle macchine rilevate;
- Draw boxes disegna bounding boxes attorno ai parcheggi liberi rilevati;
- Blur and draw boxes riunisce le operazioni effettuate dai pulsanti precedenti;
- Export image permette l'esportazione dell'immagine in una cartella predefinita nell'architettura dell'applicazione;

In aggiunta, sono presenti un contatore che mostra il numero di posti liberi rilevati dopo la creazione delle bounding boxes e, in fondo alla sidebar, un pulsante di chiusura.