Relazione Progetto 3D Perception

Francesca Stefano e Giovanni Schianchi

23 ottobre 2023

1 Dataset

Nell'ambito della realizzazione del progetto, il primo passo è consistito nella selezione di un dataset. In particolare, è stato scelto il PandaSet, il quale si prefigge di promuovere e favorire la ricerca e lo sviluppo nel settore della guida autonoma e dell'apprendimento automatico. Questo dataset rappresenta il primo set di dati open-source per veicoli autonomi disponibile a fini accademici e commerciali. Il PandaSet integra i migliori sensori LiDAR di Hesai con l'annotazione di dati di alta qualità fornita da Scale AI.

1.1 Struttura del Dataset

A partire dal dataset selezionato(disponibile al link Dataset),è stato fondamentale esaminarlo in modo approfondito al fine di comprendere non solo la sua composizione, ma anche per determinare la modalità di elaborazione dei dati, necessaria per la successiva scelta di un adeguato algoritmo di rilevamento delle immagini. I dati sono stati raccolti tramite un LiDAR rivolto in avanti con una risoluzione simile a quella delle immagini, unitamente a un LiDAR a rotazione meccanica. Le informazioni raccolte sono state annotate utilizzando una combinazione di annotazioni cuboidi e segmentazione Dalla pagina web ufficiale del dataset, è possibile constatare che sono disponibili le seguenti informazioni:

- Oltre 48.000 immagini della telecamera;
- Oltre 16.000 scansioni LiDAR;
- 100+ scene di 8 secondo l' una;
- 28 classi annotate;
- 37 etichette per la sematic segmentation;

Nel secondo stadio dell'implementazione del progetto, è stato essenziale eseguire una selezione delle caratteristiche (feature selection) al fine di concentrarsi esclusivamente su quelle che risultassero effettivamente rilevanti. Tali caratteristiche rappresentano i dati utilizzati come input per l'algoritmo YOLO al fine di generare le bounding box, e quindi va precisato che la scelta di queste caratteristiche è stata guidata proprio dai requisiti di formato imposti da YOLOv5. Tra le numerose caratteristiche disponibili, soltanto cinque si sono rivelate significative:

- **label**, la classe dell'oggetto
- **position.x**, la coordinata x dell'oggetto
- **position.y**, la coordinata y dell'oggetto
- **dimension.x**, la dimensione lungo l'asse x dell'oggetto
- **dimension.y**, la dimensione lungo l'asse y dell'oggetto

2 Obiettivo e Algoritmo di Object Detection

Sulla base dell'analisi dei dati e della revisione degli algoritmi di Object Detection disponibili in letteratura, è stato posto l'obiettivo di implementare la rilevazione tramite bounding box dei principali elementi presenti nelle strade, quali pedoni, alberi, segnaletica stradale, veicoli, ecc. A tal fine, è stato scelto l'utilizzo di YOLOv5 come algoritmo.

2.1 Implementazione

Una volta compreso il funzionamento di tale versione, si è proceduto con l' elaborazione di uno script Python che si occupa di annotazioni di immagini, definendo un dizionario che associa i nomi delle classi ai rispettivi ID. Il procedimento prosegue con la lettura delle annotazioni da una directory, l'ordinamento delle stesse e la creazione di un altro dizionario che associa gli ID delle classi ai loro nomi. In seguito, si è definita una funzione denominata *"plotboundingbox"* che accetta un'immagine e un elenco di annotazioni come input. Tale funzione traccia i bounding box di ciascuna annotazione sull'immagine e la visualizza. L'elaborazione procede con la selezione casuale di un file di annotazione, la lettura delle annotazioni da tale file e il caricamento dell'immagine corrispondente. Il processo continua con la lettura di tutti i file di immagine e annotazione dalle rispettive directory, ordinandoli. Successivamente, il dataset viene suddiviso in insiemi di **training**, **validation** e **test**. Infine, lo script sposta i file nelle rispettive cartelle. L'obiettivo di tale script risulta essere proprio quello di selezionare una serie di immagini per poter eseguire l' operazione di Fine Tuning.

2.2 Training

Si è successivemante proceduto con l' esecuzione del file di *train* di YOLOv5 a cui sono stati passati una serie di argomenti, fra cui i più importanti:

- yolov5s.yaml, file con le specifiche della rete, in cui sono state modificate il numero di classi;
- hyp.scratch.yaml, che è il file di configurazione degli iperparamentri;
- il numero di epoche;
- batch size;
- i pesi di default di YOLOv5;
- road_sign_data.yaml,file in cui sono stati aggiunti i path delle cartelle contenenti le immagini e le labels su cui fare object detenction;

2.3 Inference

Terminata la fase precedente con la creazione di un nuovo file contenente i nuovi pesi aggiornati (*best.pt*), si è proceduto testando le performance della rete neurale sulle immagini contenuto del test set. Per fare questo si è utilizzato il file detect.py di YOLO5; tale script ha fornito come risultato le immagini del testing con applicate le relative Bounding Box.

3 Risultati

I risultati del Fine Tuning si presentano tramite un insieme di informazioni mostrate per ogni epoca:

Epoch	Box loss	Obj loss	Cls loss	Precision	Recall
0	0.11704	0.06529	0.073509	0.000675	0.02756
1	0.11009	0.07568	0.070582	0.002073	0.12154
2	0.09992	0.08988	0.064842	0.006578	0.21019
3	0.09024	0.08233	0.057798	0.009563	0.07019
46	0.04046	0.04844	0.01028	0.751570	0.41139
47	0.041025	0.053769	0.010702	0.77146	0.41656
48	0.039495	0.048068	0.0098606	0.75416	0.41273
49	0.04014	0.05213	0.0091841	0.7378	0.43617

Tabella 1: Le prime 4 e ultime 4 epoche con relative informazioni.