

# Relazione di progetto

Andrea Perricone

Corso di Deep Learning  
UniCt - Informatica Magistrale

28 Febbraio 2025

## Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
<b>2</b>	<b>Dataset</b>	<b>4</b>
2.1	Acquisizione . . . . .	4
2.2	Strumenti . . . . .	5
2.3	Etichette . . . . .	5
2.3.1	Distribuzione delle annotazioni rispetto alle classi . . . . .	5
2.4	Dati reali . . . . .	5
2.4.1	Distribuzione delle annotazioni rispetto alle classi . . . . .	6
2.5	Composizione dei set di dati . . . . .	6
2.6	Esempi visuali . . . . .	7
2.6.1	Dati reali . . . . .	7
2.6.2	Dati sintetici . . . . .	7
2.7	Informazioni aggiuntive sui dati . . . . .	8
2.7.1	Data augmentation . . . . .	8
2.7.2	Organizzazione della directory contenente i vari dataset . . . . .	8
<b>3</b>	<b>Approcci</b>	<b>9</b>
<b>4</b>	<b>Valutazione</b>	<b>10</b>
4.1	Object detection . . . . .	10
<b>5</b>	<b>Esperimenti</b>	<b>11</b>
5.1	Upper Bound . . . . .	11
5.1.1	Dataset . . . . .	11
5.1.2	Fase di training . . . . .	11
5.1.3	Fase di test . . . . .	13
5.2	Lower Bound . . . . .	14
5.2.1	Dataset . . . . .	14

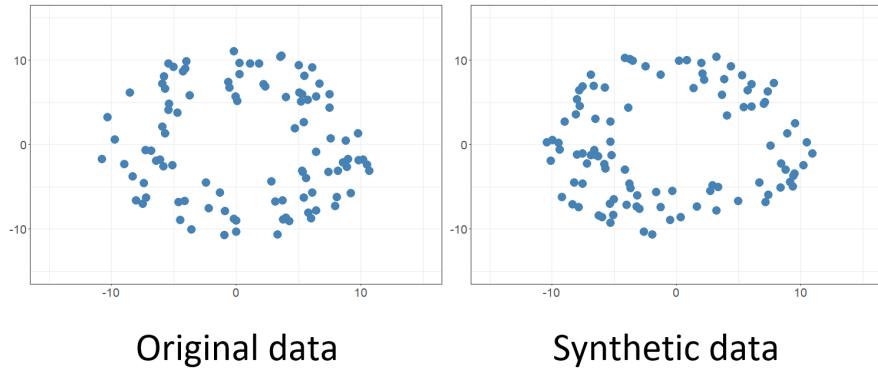
5.2.2	Fase di training . . . . .	14
5.2.3	Fase di test . . . . .	16
5.3	Misto sintetico + 25% reale . . . . .	17
5.3.1	Dataset . . . . .	17
5.3.2	Fase di training . . . . .	18
5.3.3	Fase di test . . . . .	20
5.4	Misto sintetico + 25% reale + CycleGAN D.A. . . . .	21
5.4.1	Dettagli riguardo la CycleGAN . . . . .	21
5.4.2	Dataset per CycleGAN . . . . .	21
5.4.3	Prima versione della CycleGAN . . . . .	22
5.4.4	Dataset per Object Detection . . . . .	23
5.4.5	Fase di training . . . . .	24
5.4.6	Fase di test . . . . .	25
5.4.7	Seconda versione della CycleGAN . . . . .	26
5.4.8	Fase di test (V2) . . . . .	27
5.5	Misto sintetico + 50% reale . . . . .	28
5.5.1	Dataset . . . . .	28
5.5.2	Fase di training . . . . .	28
5.5.3	Fase di test . . . . .	30
5.6	Confronto . . . . .	31
<b>6</b>	<b>Demo</b>	<b>32</b>
6.1	Menu principale . . . . .	32
6.2	Inferenza con YoloV11n . . . . .	33
<b>7</b>	<b>Codice</b>	<b>34</b>
7.1	Breve descrizione dei vari sorgenti . . . . .	34
7.1.1	demo.py . . . . .	34
7.1.2	Vari DATASET_training.ipynb . . . . .	34
7.1.3	Vari testing_DATASET.ipynb . . . . .	34
7.1.4	CycleGAN_training.ipynb . . . . .	34
7.1.5	CycleGAN_adaptation.ipynb . . . . .	34
7.2	Altri file presenti nel progetto . . . . .	35
7.3	Istruzioni per l'utilizzo . . . . .	35
<b>8</b>	<b>Conclusione</b>	<b>35</b>

# 1 Introduzione

L’obiettivo del progetto è verificare l’efficacia dell’utilizzo di dati sintetici per affrontare il problema dell’object detection. L’adozione di dati generati artificialmente è motivata dalla possibilità di ottenere, in modo efficiente e a costi ridotti, sia le immagini che le relative etichette, rispetto ai tradizionali metodi di acquisizione e annotazione manuale.

Tuttavia, l’impiego di dati sintetici presenta alcune criticità, in quanto le distribuzioni dei dati generati artificialmente non coincidono perfettamente con quelle dei dati reali. Di conseguenza, è spesso necessario adottare tecniche di Domain Adaptation per ridurre il divario tra le due distribuzioni e migliorare le prestazioni del modello nell’applicazione su dati reali.

Per questo motivo, verrà analizzata anche una forma di adattamento di dominio al fine di valutarne l’impatto sulle capacità del modello di Object Detection di generalizzare ai dati reali.



The synthetic data retains the structure of the original data but is not the same

## 2 Dataset

Il dataset che è stato utilizzato per effettuare gli esperimenti di questo progetto è EgoISM-HOI che è un nuovo dataset multimodale composto da immagini sintetiche di interazioni uomo-oggetto in prima persona (EHOI) in un ambiente industriale, arricchito con annotazioni dettagliate di mani e oggetti. Il dataset include 39.304 immagini RGB, 23.356 mappe di profondità e maschere di segmentazione istanziate, 59.860 annotazioni di mani, 237.985 istanze di oggetti suddivise in 19 categorie e 35.416 annotazioni di interazioni uomo-oggetto in prima persona. È stato progettato per affrontare la carenza di dataset pubblici in questo contesto. Il dataset, insieme al codice sorgente e ai modelli pre-addestrati, è disponibile pubblicamente.



Figure 1: Immagine di presentazione del dataset

### 2.1 Acquisizione

Il dataset è stato acquisito tramite una pipeline di generazione di immagini sintetiche basata su un ambiente virtuale industriale. Sono stati utilizzati motori grafici per simulare scene realistiche in prima persona (EHOI), con variazioni nelle pose delle mani, negli oggetti e nelle condizioni ambientali. Ogni immagine è stata annotata automaticamente con mappe di profondità, maschere di segmentazione, pose delle mani e interazioni uomo-oggetto.

## 2.2 Strumenti

Gli oggetti e l'ambiente vengono digitalizzati utilizzando **scanner commerciali**, generando modelli 3D realistici dei componenti dell'ambiente. Inoltre attraverso un **motore grafico** vengono create le scene sintetiche in prima persona.

## 2.3 Etichette

Sono stati considerati sia oggetti fissi (ad esempio, oscilloscopio, alimentatore) che oggetti mobili (ad esempio, cacciavite elettrico, tavola elettrica) presenti nel laboratorio. Sono presenti 20 classi diverse, tra cui alimentatori, strumenti di saldatura, cacciaviti elettrici, pinze e diverse prese elettriche.

### 2.3.1 Distribuzione delle annotazioni rispetto alle classi

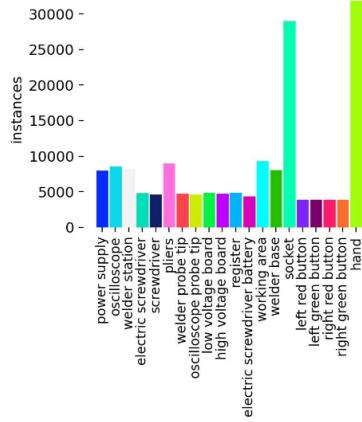


Figure 2: Numero di annotazioni per classe tra i dati sintetici di EgoISM-HOI

## 2.4 Dati reali

Il dataset EgoISM-HOI contiene anche i dati reali, in particolare **EgoISM-HOI-Real** espande il dataset reale originale ENIGMA, con 42 video egocentrici registrati in laboratorio. I video mostrano soggetti intenti a testare e riparare schede elettriche usando strumenti da laboratorio. L'acquisizione è stata guidata da un'applicazione per Microsoft Hololens 2, che forniva istruzioni audio, immagini esplicative e supportava comandi vocali per facilitare il processo. Sono state definite 8 procedure, ciascuna con diversi strumenti e schede elettriche, e coinvolti 19 partecipanti (17 uomini, 2 donne). Sono stati raccolti 18 ore e 48 minuti di registrazioni in alta risoluzione (2272x1278, 30fps). Le annotazioni, effettuate su 15.948 (di cui 12759 su **EgoISM-HOI-Real**) immagini, includono:

1. Bounding box per mani e oggetti.
2. Lato della mano (sinistra/destra).
3. Stato di contatto della mano con l'oggetto (contatto/no contatto).
4. Relazione mano-oggetto (es. *mano X tocca oggetto Y*).
5. **Categoria dell'oggetto.**

#### 2.4.1 Distribuzione delle annotazioni rispetto alle classi

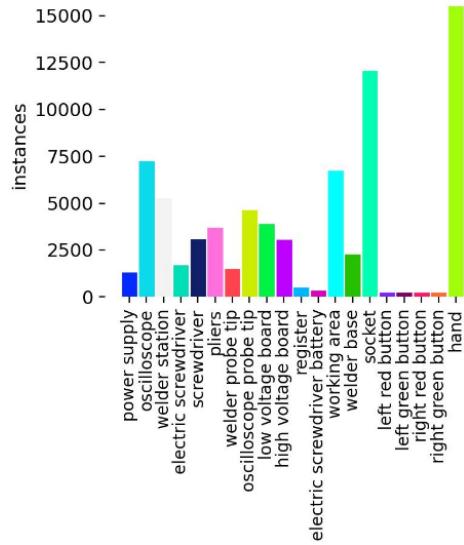


Figure 3: Numero di annotazioni per classe tra i dati reali di EgoISM-HOI

E' possibile notare un forte sbilanciamento tra le classi rispetto al dataset sintetico, questo perché è molto più facile generare dati sintetici con labels "gratis" rispetto ad acquisirli ed etichettarli per colmare questo tipo di divari.

#### 2.5 Composizione dei set di dati

Come è stato detto in precedenza sono presenti 39.304 immagini di cui 15.948 sono immagini reali e 23.356 sono immagini sintetiche. Questo dataset verrà utilizzato in vari modi differenti a seconda dell'esperimento per questo motivo la composizione verrà descritta insieme ad ogni esperimento successivamente. Ciò che è stato reso fisso per poter permettere il confronto dei risultati dei vari esperimenti è il **test set**, questo è il **20% dei dati reali disponibili in EgoISM-HOI-Real**. I rimanenti sono utilizzati per effettuare i vari esperimenti.

## 2.6 Esempi visuali

### 2.6.1 Dati reali



Figure 4: Vari esempi di dati reali

### 2.6.2 Dati sintetici



Figure 5: Vari esempi di dati sintetici

## 2.7 Informazioni aggiuntive sui dati

### 2.7.1 Data augmentation

Nel caso del modello per l'adattamento del dominio, che verrà presentata successivamente, sono stati applicati crop e flip casuali per migliorare la capacità di generalizzazione del modello.

### 2.7.2 Organizzazione della directory contenente i vari dataset

Il dataset originariamente si presenta nel formato COCO. I vari dataset ricavati da quello originario saranno nel formato YOLO tranne quello utilizzato per l'approccio di adattamento del dominio.

(È possibile scaricare i vari dataset in formato zip dal seguente [link](#).)

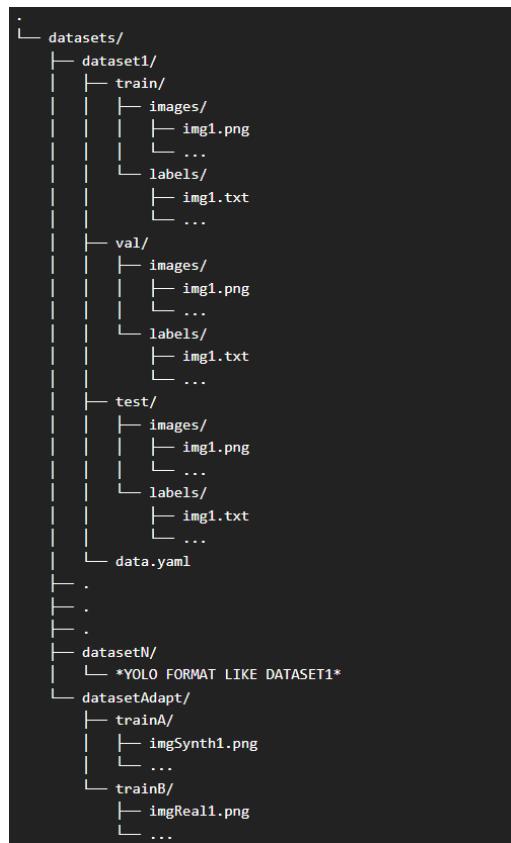


Figure 6: Struttura ad albero del dataset.

Il dataset originale EgoISM-HOI è disponibile al seguente [link](#).

### 3 Approcci

L’idea è quella di confrontare diversi approcci per l’addestramento di un modello di object detection, in particolare:

- Creare un **Upper Bound** da utilizzare come riferimento nel confronto, sfruttando l’intero set di dati reali disponibili nel dataset. In questo scenario, si assume idealmente di avere accesso a una grande quantità di dati reali, simulando una condizione ottimale per il modello.
- Creare un **Lower Bound** da utilizzare come riferimento nel confronto, sfruttando esclusivamente i dati sintetici presenti nel dataset. Questo approccio rappresenta il caso atteso con le prestazioni peggiori, poiché il modello viene addestrato su dati che potrebbero non riflettere accuratamente la realtà.
- Valutare le prestazioni di un modello addestrato utilizzando l’intero set di dati sintetici, integrato con il 25% dei dati reali disponibili nel dataset. Questo esperimento permette di quantificare l’impatto della presenza di una quantità **limitata** di dati reali nel migliorare la capacità di generalizzazione del modello.
- Valutare le prestazioni di un modello addestrato utilizzando l’intero set di dati sintetici, integrato con il 50% dei dati reali disponibili nel dataset. Questo esperimento permette di quantificare l’impatto della presenza di una quantità **significativa** di dati reali nel migliorare la capacità di generalizzazione del modello.
- Valutare le prestazioni di un modello addestrato su dati sintetici sottoposti a Domain Adaptation, utilizzando in questo caso un adattamento di stile a livello di pixel con **CycleGAN**. L’obiettivo è quantificare l’impatto dell’adattamento nel migliorare la capacità di generalizzazione del modello rispetto all’uso di dati sintetici non adattati.
- Analizzare e confrontare i risultati per valutare l’efficacia e il potenziale dell’utilizzo di dati sintetici, sia da soli che integrati con una quantità limitata di dati reali. Inoltre, verificare l’impatto delle tecniche di Domain Adaptation nel migliorare le prestazioni del modello.

## 4 Valutazione

### 4.1 Object detection

Per quanto riguarda la parte di object detection le misure di valutazione utilizzate sono:

- **Matrice di confusione:** È una tabella utilizzata per valutare le performance di un modello di classificazione supervisionato, in questo caso per valutare le performance della componente di classificazione del modello di object detection. Le righe rappresentano i valori predetti dal modello, mentre le colonne rappresentano i valori reali. Ogni cella della matrice contiene il numero di campioni per ciascuna combinazione di classe predetta e classe reale.

- **Precision:** è definita come la proporzione dei veri positivi (TP) rispetto al totale delle predizioni positive fatte dal modello:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

- **Recall:** è definito come la proporzione dei veri positivi (TP) rispetto al totale dei veri positivi presenti nei dati:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

- **F1-score:** una metrica utilizzata per valutare la precisione complessiva di un modello di classificazione. Questo valore sarà alto quando sia la Precision che il Recall avranno valori alti, essendo la media armonica tra di essi:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

- **mean Average Precision 50-95 (mAP50-95)** : è una metrica utilizzata per valutare le performance dei modelli di object detection su una gamma di soglie di Intersection over Union (IoU) da 0.5 a 0.95, in incrementi di 0.05. È calcolato come la media delle Average Precision (AP) per ogni classe su queste diverse soglie di IoU. L'AP calcola l'area sotto la curva Precision-Recall, fornendo una misura quantitativa della capacità del modello di fare predizioni precise su un insieme di dati.

## 5 Esperimenti

Tutti gli esperimenti sono stati condotti sulla piattaforma Kaggle, utilizzando le risorse GPU messe a disposizione per un tempo limitato. Per l'Object Detection, è stato utilizzato il modello YOLOv11n, scelto per la sua leggerezza e per la capacità di elaborare i dati di EgoISM-HOI in un tempo accettabile.

Tutti i training relativi a YOLO sono stati eseguiti ridimensionando le immagini a  $640 \times 640$  e limitando l'addestramento a 30 epoch, in modo da gestire al meglio l'utilizzo delle ore di GPU disponibili.

### 5.1 Upper Bound

#### 5.1.1 Dataset

Come precedentemente indicato, ogni esperimento utilizza split differenti per train e validation. In questo caso, i dati reali rimanenti, escluso il 20% assegnato al test set, sono stati suddivisi assegnando il 20% al validation set e il restante al train set. Di conseguenza, la suddivisione dei dati è stata effettuata nel seguente modo:

- Train: 80% dei dati reali disponibili
- Val: 20% dei dati reali disponibili
- Test: 20% dei dati reali totali (**fissato**)

#### 5.1.2 Fase di training

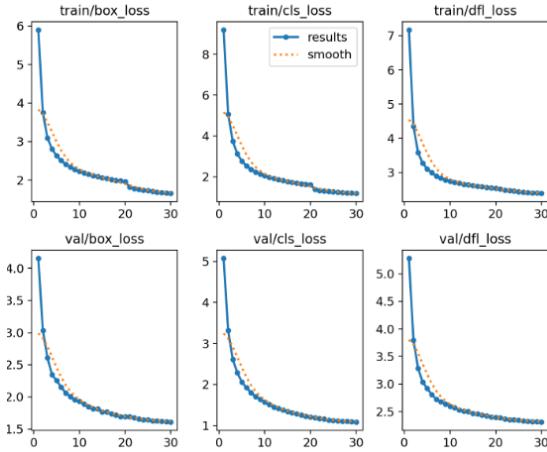


Figure 7: Loss di train e validation.

Si può notare che il modello non sembra essere arrivato a convergenza però si comporta già abbastanza bene nel training set e nel validation set con sole 30 epoche. Questo grazie al pretrain di YOLO che velocizza il processo.

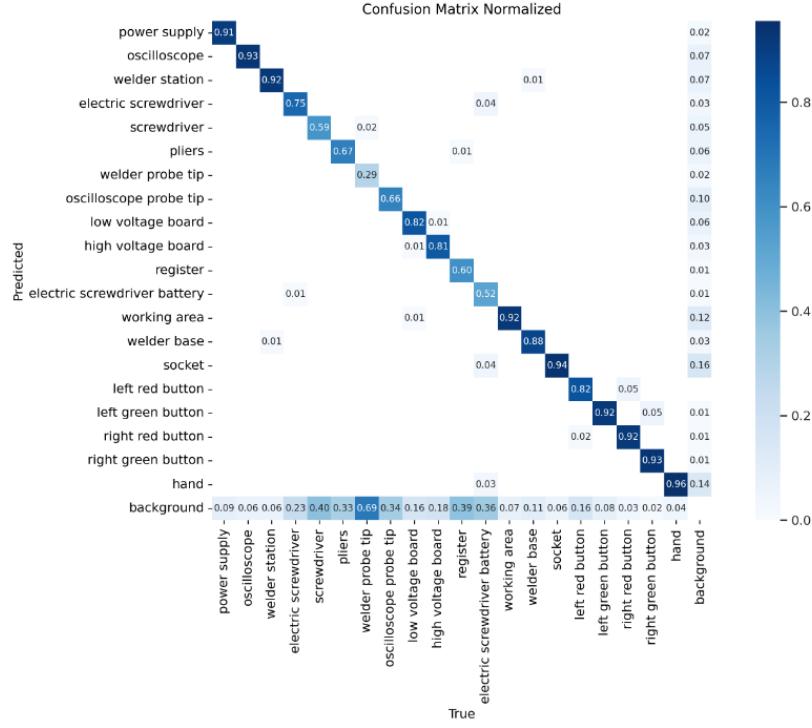


Figure 8: Matrice di confusione sul validation set.

Il modello incontra difficoltà già durante la fase di addestramento nel riconoscere alcune classi. Questo potrebbe essere causato dal numero limitato di epoche di training o dalla capacità ridotta del modello in termini di parametri. Se il problema fosse invece legato alla variabilità dei dati, ci si aspetta di osservare miglioramenti nei successivi esperimenti.

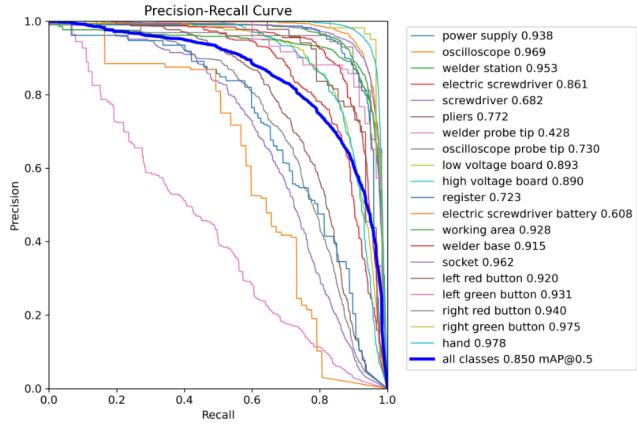


Figure 9: Curva P-R su train set.

### 5.1.3 Fase di test

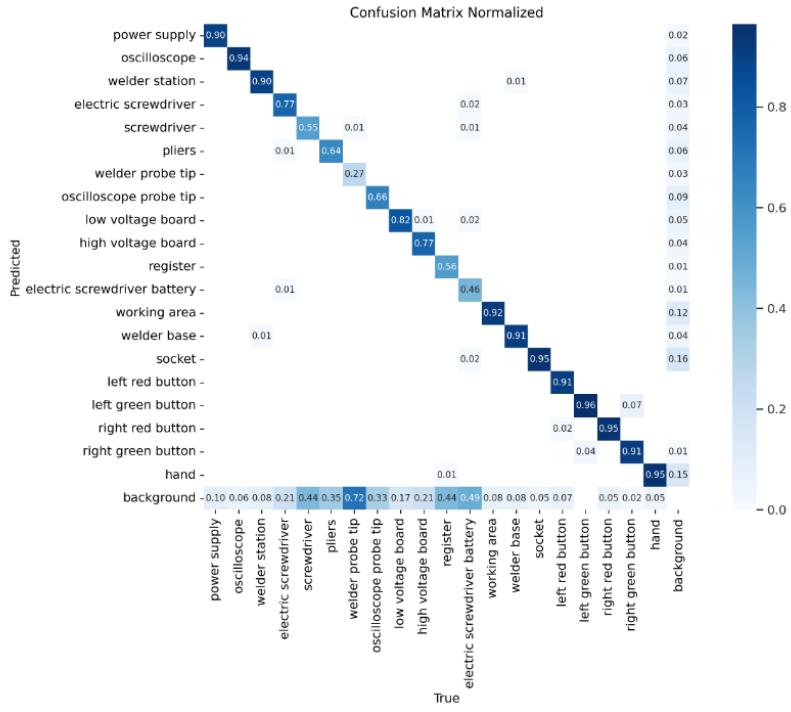


Figure 10: Matrice di confusione sul test set.

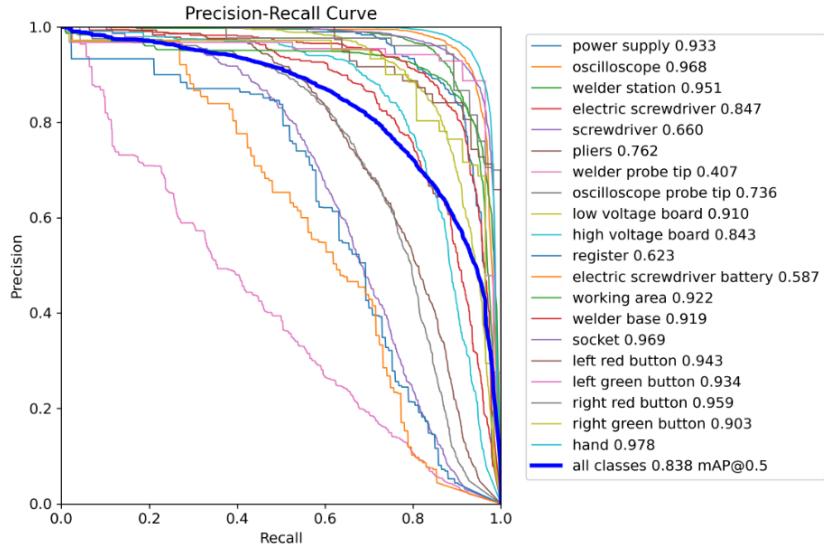


Figure 11: Curva P-R sul test set.

I risultati sembrano essere semplicemente leggermente peggiori. Per quanto riguarda questo esperimento c'è poco da attenzionare ma servirà appunto come punto di riferimento per gli esperimenti successivi.

## 5.2 Lower Bound

In questo esperimento, l'obiettivo è addestrare un object detector esclusivamente su dati sintetici e valutarne la capacità di generalizzazione su dati reali. Questo modello è previsto essere il meno performante tra quelli proposti in questo progetto, in quanto verrà addestrato su dati appartenenti a un dominio differente rispetto a quello reale.

### 5.2.1 Dataset

Di seguito è riportata la suddivisione del dataset:

- Train: 80% dei dati sintetici
- Val: 20% dei dati sintetici
- Test: 20% dei dati reali (**fissato**)

### 5.2.2 Fase di training

I risultati ottenuti durante il training sono eccellenti.

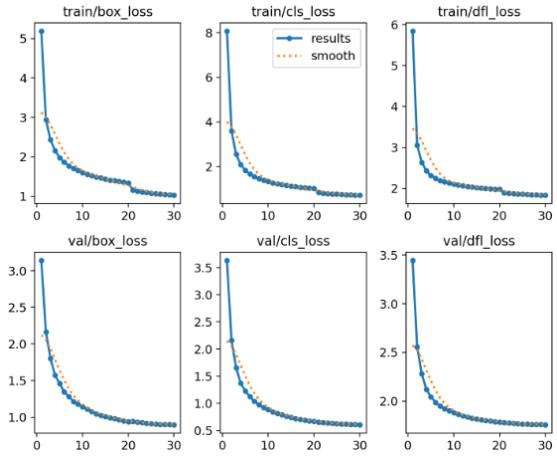


Figure 12: Loss di train e validation.

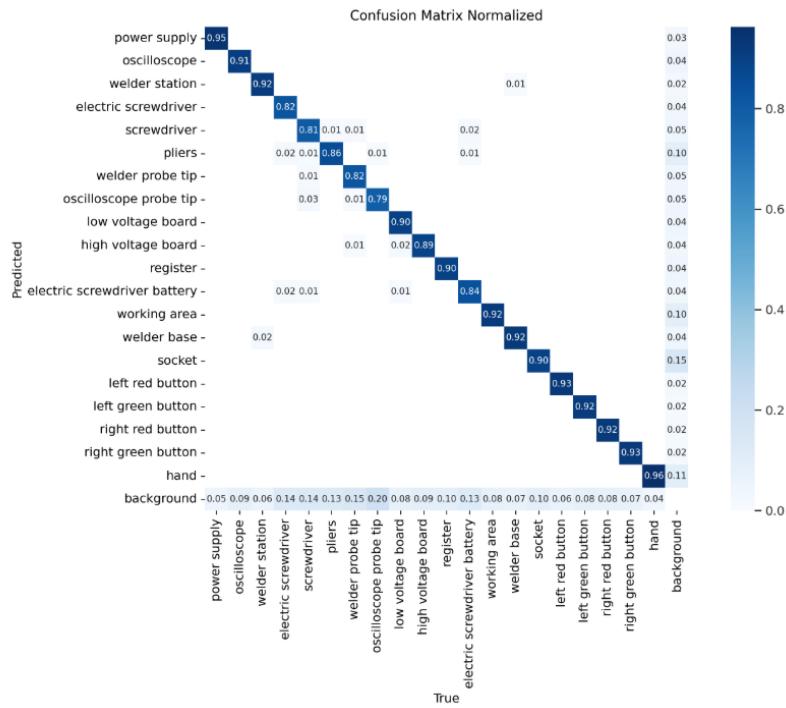


Figure 13: Matrice di confusione sul validation set.

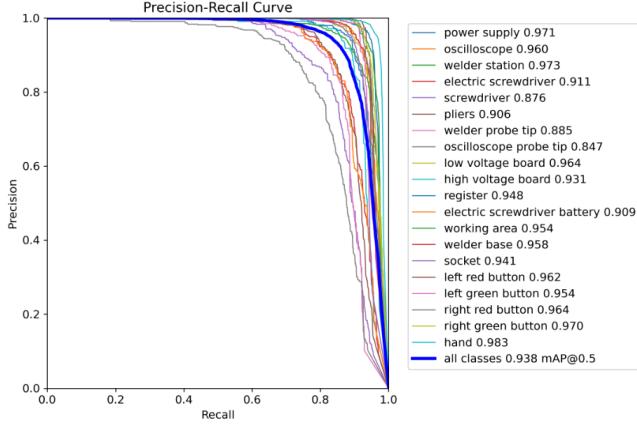


Figure 14: Curva P-R su validation set.

### 5.2.3 Fase di test

Valutando però il modello sul test set che è composto interamente da immagini reali abbiamo:

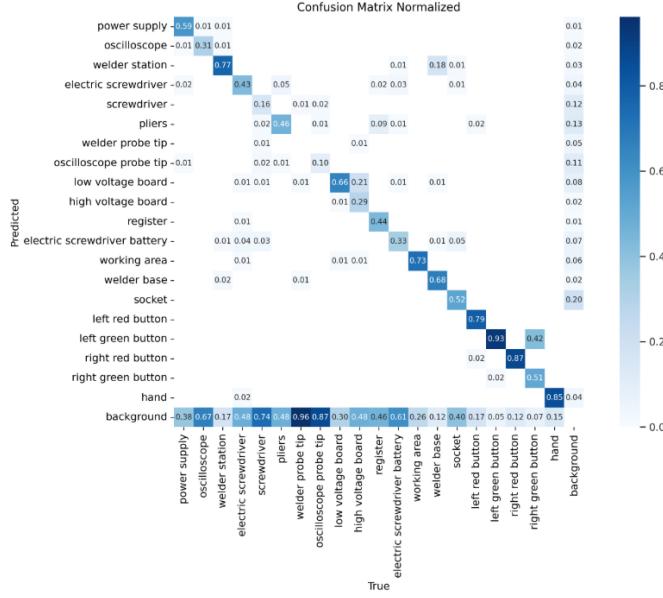


Figure 15: Matrice di confusione sul test set.

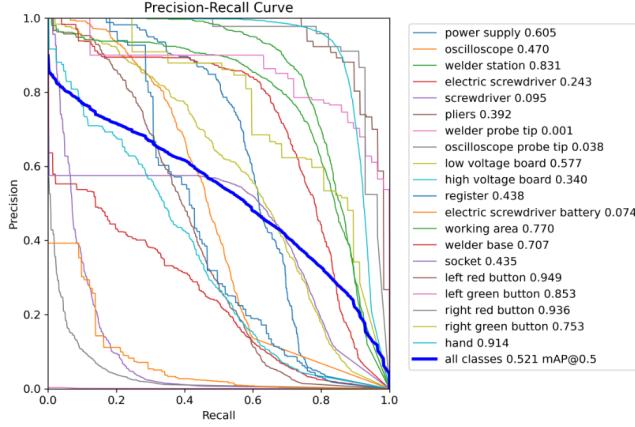


Figure 16: Curva P-R su test set.

I risultati ottenuti sul test set sono significativamente inferiori. È possibile osservare che questa differenza non è attribuibile a un tempo di training ridotto, bensì al fatto che i dati del dominio sintetico non sono sufficientemente simili a quelli del dominio reale, compromettendo la capacità del modello di generalizzare.

### 5.3 Misto sintetico + 25% reale

In questo caso, i dati sintetici sono stati ampliati utilizzando il 25% dei dati reali **disponibili**. L'obiettivo era verificare se l'inclusione di una piccola quantità di dati reali possa migliorare la capacità di generalizzazione del modello sul test set.

#### 5.3.1 Dataset

Il dataset in questo caso è composto nel seguente modo:

- **Set misto:** 100% dei dati sintetici + 25% dei dati reali disponibili.
- Train: 80% del set misto.
- Val: 20% del set misto.
- Test: 20% dei dati reali. (**fissato**)

### 5.3.2 Fase di training

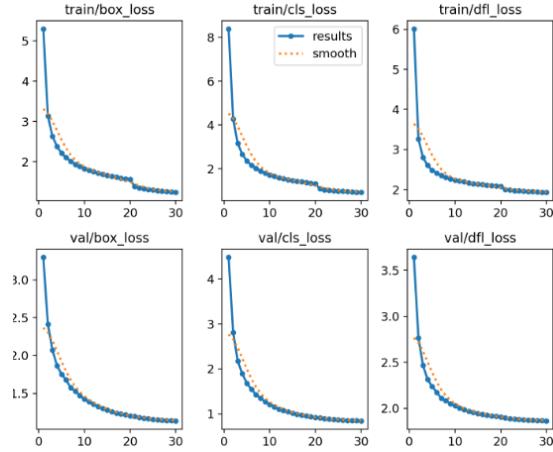


Figure 17: Loss di train e validation.

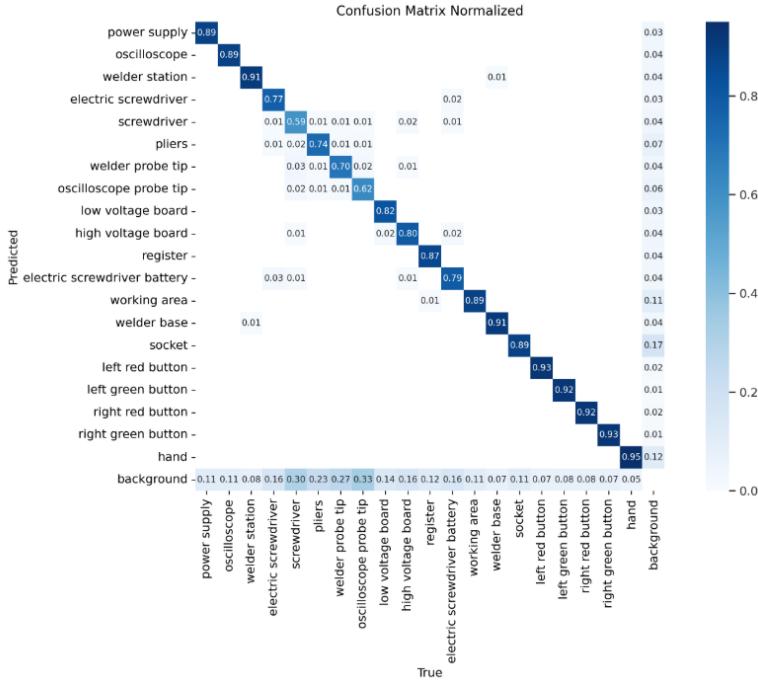


Figure 18: Matrice di confusione sul validation set.

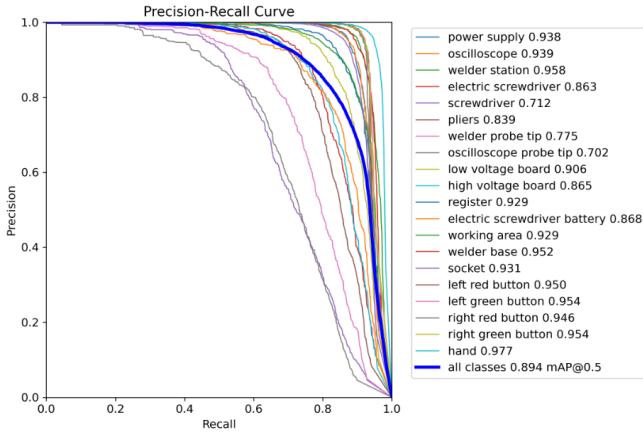


Figure 19: Curva P-R su validation set.

È possibile osservare che, in questo caso, il modello presenta prestazioni inferiori sul validation set. Questo accade perché l'introduzione di dati reali ha creato delle difficoltà durante il processo di apprendimento. I dati sintetici sono relativamente semplici, con oggetti facilmente riconoscibili, anche grazie a condizioni di illuminazione costanti, ecc...

### 5.3.3 Fase di test

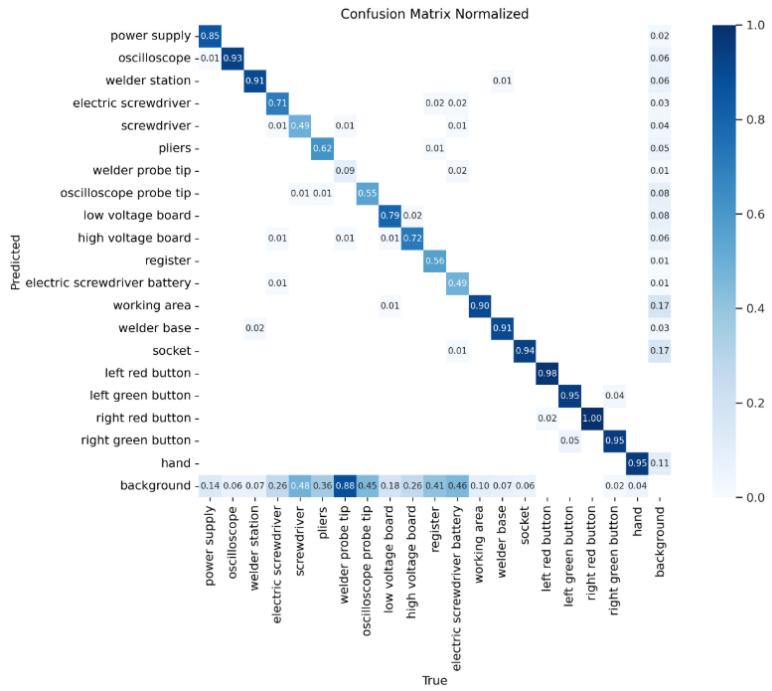


Figure 20: Matrice di confusione sul test set.

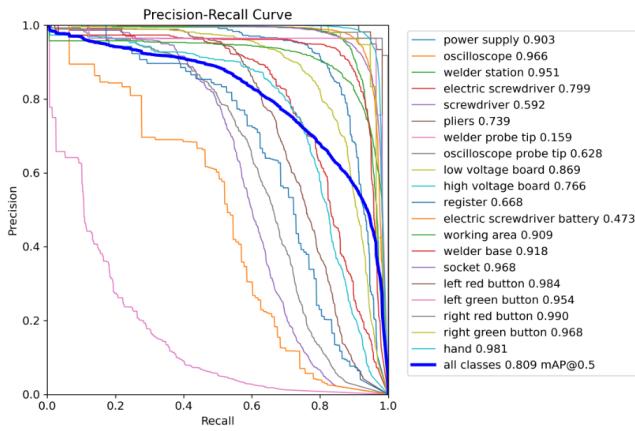


Figure 21: Curva P-R su test set.

Ora che il modello ha avuto la possibilità di osservare anche una piccola parte di dati reali durante la fase di addestramento, le prestazioni sul test set sono

migliorate significativamente. Analizzando la matrice di confusione, la curva Precision-Recall e la mAP@0.5 calcolate sul test set, si nota che i risultati si avvicinano a quelli dell'Upper Bound presentato in precedenza. Questo è un risultato rilevante, considerando che la maggior parte dei dati utilizzati è stata generata a un costo potenzialmente molto ridotto e che è stato sufficiente un numero limitato di esempi reali per migliorare la capacità di generalizzazione del modello.

## 5.4 Misto sintetico + 25% reale + CycleGAN D.A.

Nei precedenti esperimenti sono state analizzate le potenzialità dell'utilizzo di dati sintetici supportati da una piccola porzione di dati reali. Il presente esperimento, invece, cerca di adattare il dominio dei dati sintetici per allinearli a quello dei dati reali. In particolare, verrà applicata una tecnica di adattamento pixel-per-pixel mediante l'utilizzo di CycleGAN. Questo approccio consente di trasformare un'immagine appartenente a un dominio in un'immagine appartenente a un altro dominio, mantenendo invariato il contenuto ma rendendola visivamente simile alle immagini del dominio di destinazione.

### 5.4.1 Dettagli riguardo la CycleGAN

Un primo tentativo è stato quello di utilizzare direttamente il modello presentato nel relativo paper e visto a lezione. Tuttavia, la rete risultava eccessivamente complessa, rendendo il processo di addestramento troppo oneroso in termini di tempo, considerate le risorse disponibili su Kaggle. Di conseguenza, è stato necessario semplificare la rete e ottimizzare il training attraverso la modifica di alcuni iperparametri, in particolare:

- Ridurre il numero di Residual Block utilizzati nel modello, da 9 a 4.
- Resize delle immagini a 640x640
- Numero di elementi nei vari batch: 9.

L'obiettivo di queste modifiche era ridurre l'elevato impatto sulla VRAM. Tuttavia, nonostante gli adattamenti, è stato possibile addestrare il modello solo per 5 epoche, con un tempo di training complessivo di 7 ore e 15 minuti. Successivamente, è stato effettuato un resume del training per ulteriori 5 epoche al fine di ottenere un confronto più accurato.

### 5.4.2 Dataset per CycleGAN

Per fare il training della CycleGAN è stato utilizzato un dataset composto in questo modo:

- **Dominio A:** 12800 immagini sintetiche
- **Dominio B:** 12800 immagini reali

### 5.4.3 Prima versione della CycleGAN

Per quanto riguarda la CycleGAN è possibile osservare le seguenti loss:

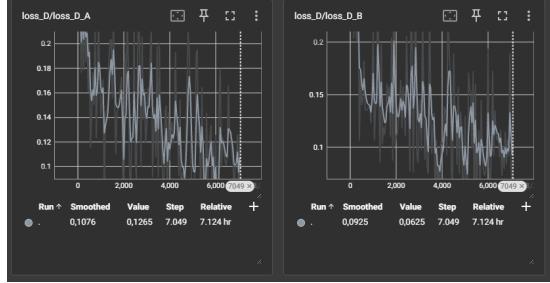


Figure 22: Loss dei discriminatori

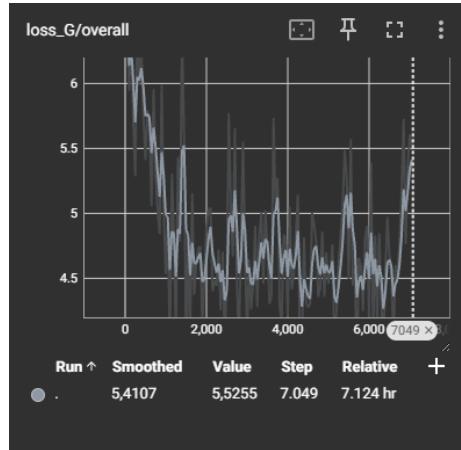


Figure 23: Loss complessiva dei generatori.

Come evidenziato dal grafico, la loss complessiva dei generatori risulta ancora elevata, il che è chiaramente attribuibile al numero limitato di epoche di training. Infatti, osservando alcuni esempi di applicazione della CycleGAN, è evidente che i risultati visivi non sono di qualità ottimale:

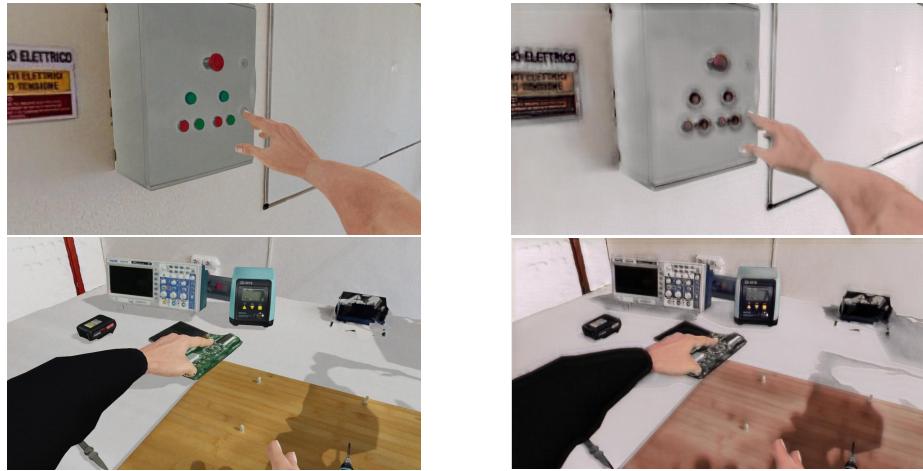


Figure 24: Vari esempi di dati sintetici adattati

#### 5.4.4 Dataset per Object Detection

Il 100% dei dati sintetici sono stati adattati dalla CycleGAN vista precedentemente e il dataset è formato nel seguente modo:

- Set di dati misto: 100% dati sintetici adattati + 25% dati reali
- Train: 80% del set di dati misto
- Validation: 20% del set di dati misto
- Test: 20% dei dati reali (**fissato**)

#### 5.4.5 Fase di training

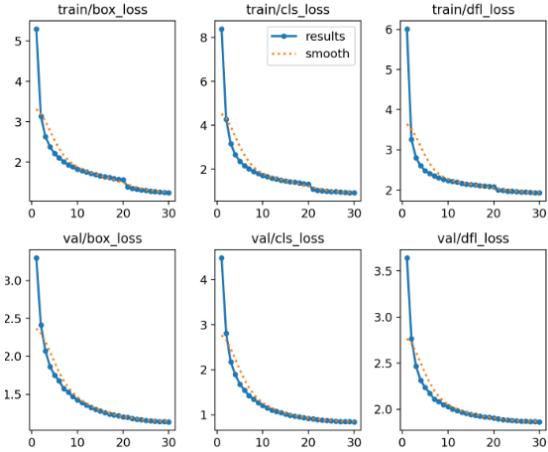


Figure 25: Loss di train e validation.

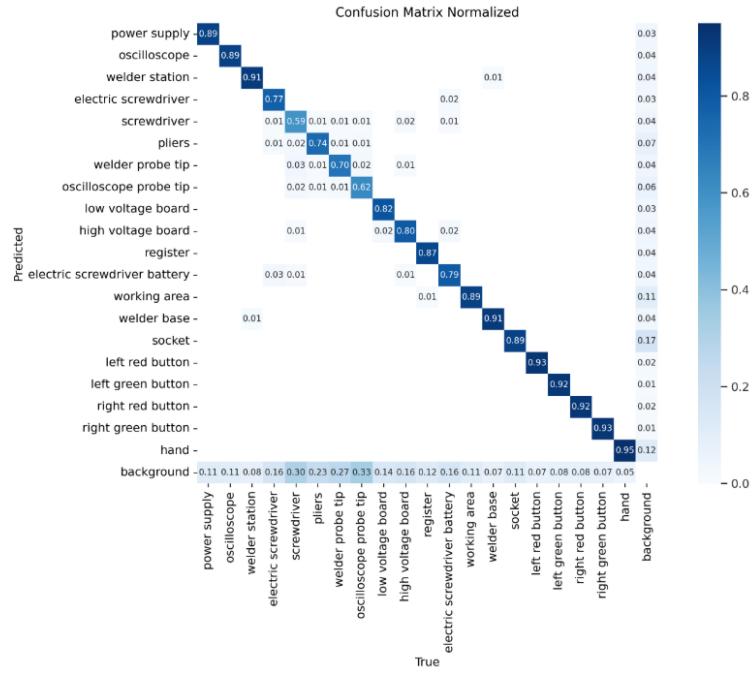


Figure 26: Matrice di confusione sul validation set.

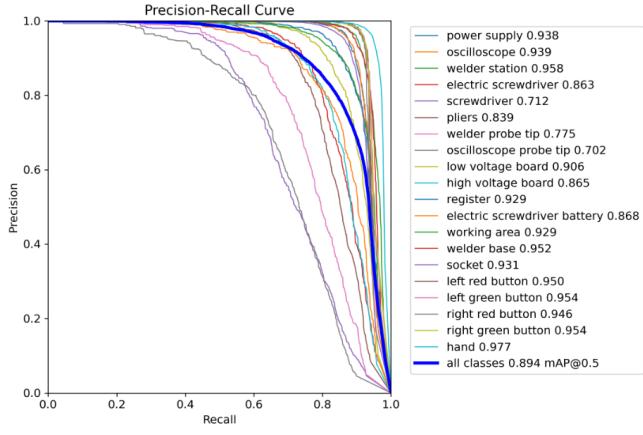


Figure 27: Curva P-R su validation set.

#### 5.4.6 Fase di test

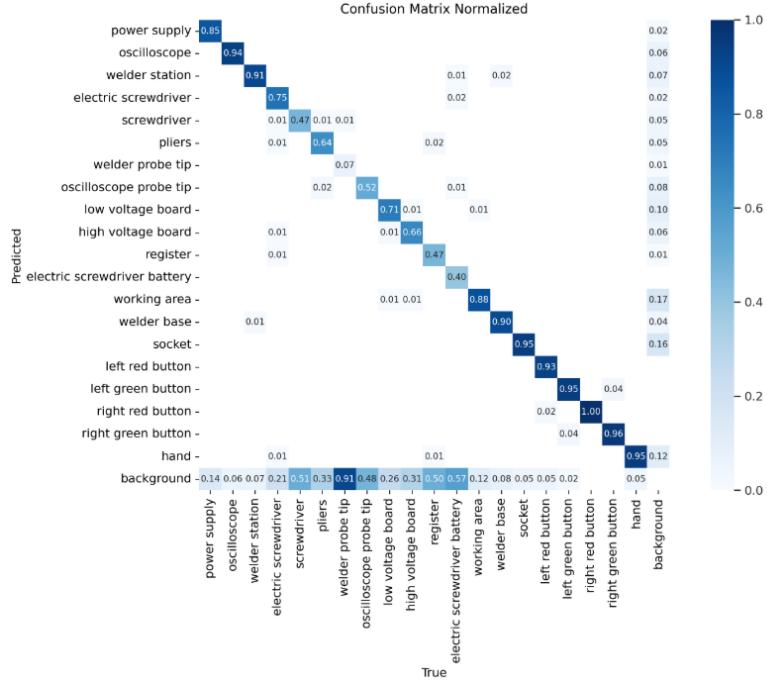


Figure 28: Matrice di confusione sul test set.

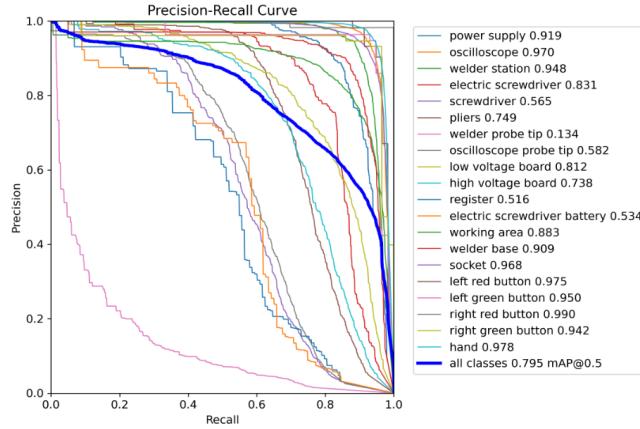


Figure 29: Curva P-R su test set.

La capacità di generalizzazione sul test set è peggiorata rispetto all'utilizzo di dati sintetici non adattati. Questo risultato è probabilmente dovuto a un addestramento insufficiente della CycleGAN, che ha portato alla generazione di artefatti, incluse distorsioni e alterazioni cromatiche.

#### 5.4.7 Seconda versione della CycleGAN

Alla luce dei risultati precedenti, è stato tentato un ulteriore addestramento della CycleGAN per altre 5 epoche, tuttavia senza ottenere miglioramenti significativi:

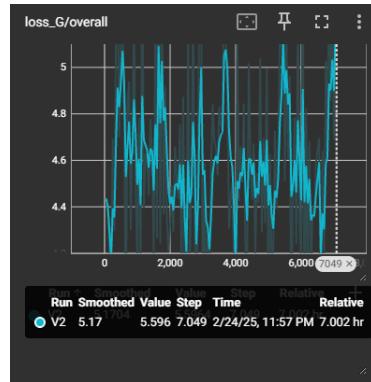


Figure 30: Loss complessiva dei generatori.

Poiché i migliori parametri della CycleGAN, in termini di loss complessiva dei generatori, non sono stati salvati, si è dovuto procedere utilizzando gli ultimi.

#### 5.4.8 Fase di test (V2)

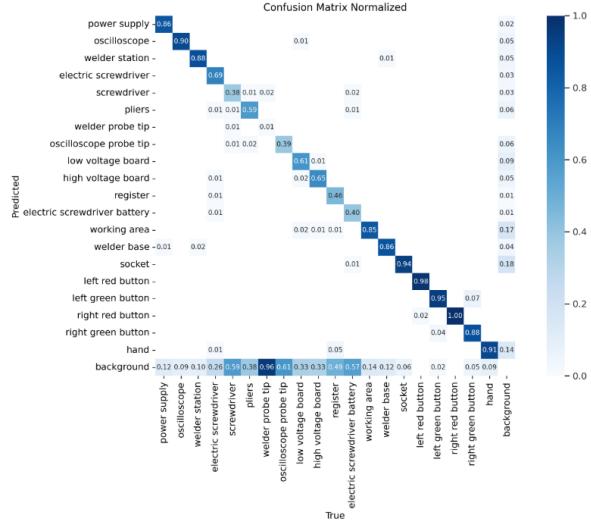


Figure 31: Matrice di confusione sul test set.

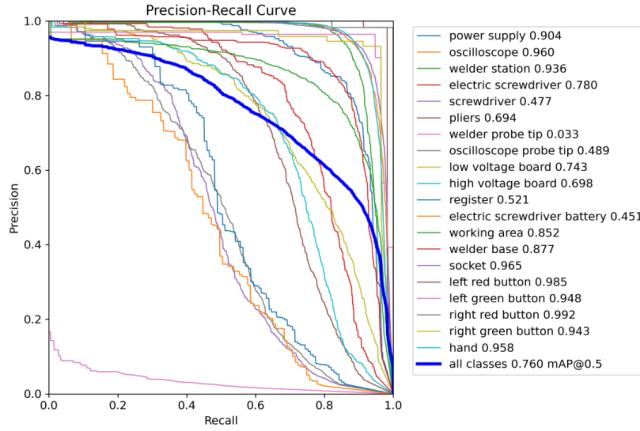


Figure 32: Curva P-R su test set.

Dalla confusione matrix e dalla curva Precision-Recall si può osservare chiaramente un calo delle prestazioni. Questo è una diretta conseguenza del fatto che la CycleGAN non ha raggiunto un miglioramento significativo; anzi, le forti oscillazioni durante l'addestramento potrebbero aver portato all'utilizzo di parametri che rendono il modello di Object Detection leggermente meno performante rispetto alla versione precedente.

## 5.5 Misto sintetico + 50% reale

Considerando le problematiche emerse nell'esperimento precedente, questo studio si propone di verificare se l'espansione del set di dati reali integrati ai dati sintetici abbia un impatto significativo sulle prestazioni del modello. Tuttavia, non verrà applicata alcuna tecnica di Domain Adaptation, poiché le risorse hardware e i limiti di tempo imposti da Kaggle ne rendono l'efficacia non ottimale.

### 5.5.1 Dataset

Il dataset è composto nel seguente modo:

- **Set misto:** 100% dei dati sintetici + 50% dei dati reali disponibili.
- Train: 80% del set misto.
- Val: 20% del set misto.
- Test: 20% dei dati reali. (**fissato**)

### 5.5.2 Fase di training

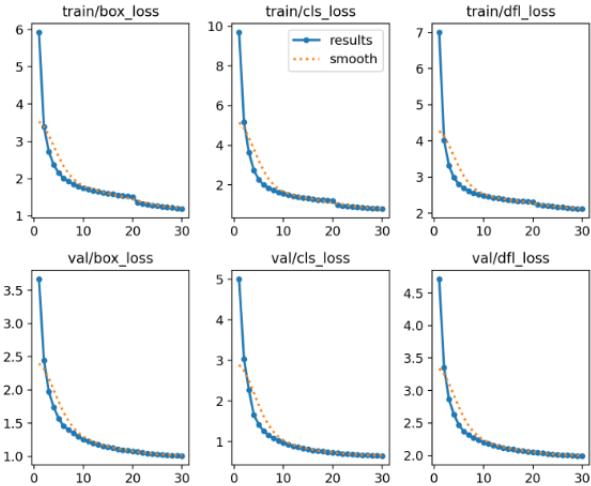


Figure 33: Loss di train e validation.

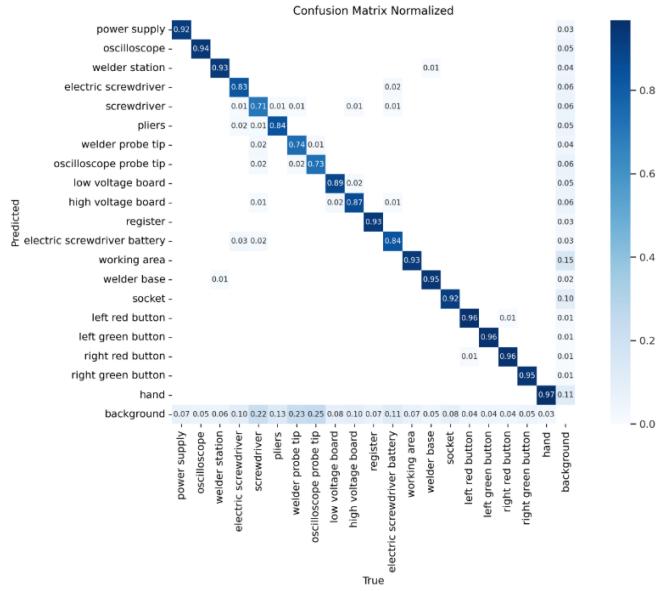


Figure 34: Matrice di confusione sul validation set.

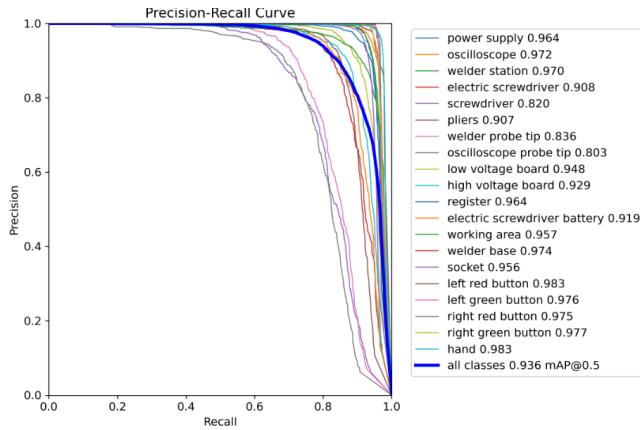


Figure 35: Curva P-R su validation set.

### 5.5.3 Fase di test

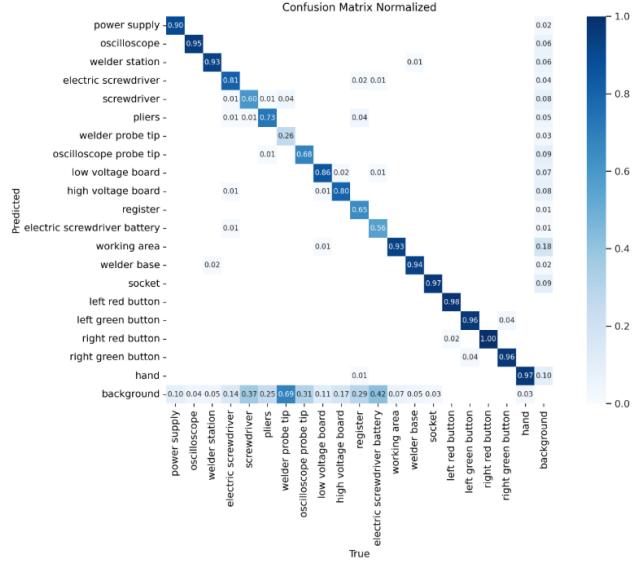


Figure 36: Matrice di confusione sul test set.

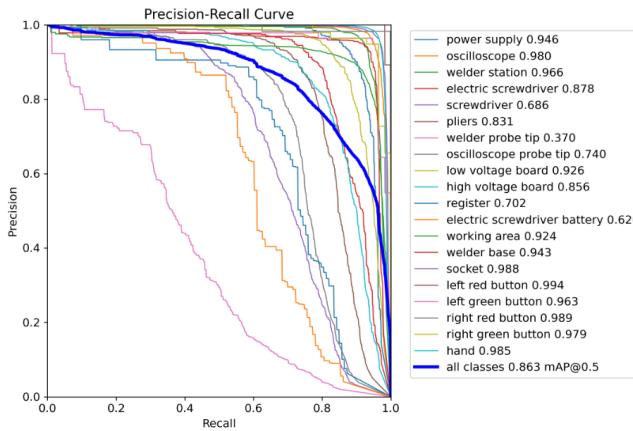


Figure 37: Curva P-R su test set.

Si osserva come le prestazioni del modello superino quelle dell'Upper Bound presentato in precedenza, confermando il potenziale dell'integrazione di dati sintetici con un set di dati reali, anche se di dimensioni significativamente inferiori rispetto ai dati sintetici.

## 5.6 Confronto

Dataset	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95
Reali	0.813	0.779	0.78 a 0.259 conf.	0.838	0.621
Sintetici	0.553	0.518	0.52 a 0.248 conf.	0.521	0.332
Sintetici + 25% reali	0.836	0.758	0.78 a 0.254 conf.	0.809	0.584
Sintetici con D.A. + 25% reali	0.822	0.744	0.77 a 0.252 conf.	0.795	0.575
Sintetici con D.A. + 25% reali V2	0.795	0.708	0.74 a 0.264 conf.	0.76	0.547
Sintetici + 50% reali	0.883	0.817	0.84 a 0.256 conf.	0.863	0.643

Table 1: Metriche di valutazione calcolate su tutte le classi

Riepilogando, l'integrazione di dati sintetici con una piccola quantità di dati reali si è dimostrata una strategia efficace, permettendo di ottenere prestazioni comparabili a quelle di un modello addestrato esclusivamente su dati reali. In particolare, il modello addestrato con il 50% di dati sintetici ha addirittura superato le prestazioni dell'Upper Bound, ovvero il modello allenato interamente su dati reali. Considerando che l'addestramento su dati reali comporta costi significativamente più elevati sia per l'acquisizione che per l'annotazione, questi risultati evidenziano il valore dell'utilizzo diretto di dati sintetici per ridurre i costi o migliorare la capacità di generalizzazione del modello su dati mai visti.

Tuttavia, i risultati ottenuti non hanno dimostrato l'efficacia delle tecniche di adattamento del dominio per i dati sintetici. Ciò è probabilmente dovuto al numero limitato di epoche di addestramento della CycleGAN, alla riduzione della complessità della rete e all'elevata loss complessiva dei generatori, che si attestava intorno a 5,5 in entrambe le versioni, suggerendo che i generatori non avevano ancora raggiunto un livello di performance ottimale. Nonostante questi limiti, l'adattamento del dominio con CycleGAN presenta un potenziale significativo. Con un addestramento più approfondito e una maggiore ottimizzazione della rete, è plausibile che avrebbe potuto migliorare ulteriormente le prestazioni dell'Object Detector, arrivando potenzialmente a eguagliare o superare quelle di un modello addestrato esclusivamente su un ampio dataset di dati reali.

## 6 Demo

La demo è un piccolo programma scritto in python con l'utilizzo di librerie per creare semplici interfacce grafiche. Tramite questo programma è possibile effettuare inferenza su delle immagini date in input dall'utente.

### 6.1 Menu principale

All'avvio del programma comparirà un menu con un pulsante che permetterà di effettuare inferenza su un input a scelta e dei radio button che permettono la scelta del modello da utilizzare per effettuare detection nelle immagini in input.

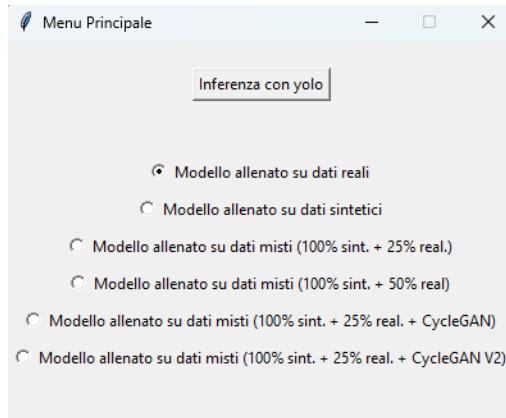


Figure 38: Menu principale del programma.

## 6.2 Inferenza con YoloV11n

Una volta premuto il pulsante nel menu principale, si apre una nuova finestra con un pulsante che permette di selezionare un’immagine di input. In alternativa, è possibile trascinare l’immagine all’interno della box evidenziata.



Figure 39: Finestra da cui è possibile selezionare il file.

Quando si seleziona un input che fa parte degli input di test e quindi sono disponibili le labels allora il programma visualizzerà l’input con le labels a sinistra e i risultati a destra.

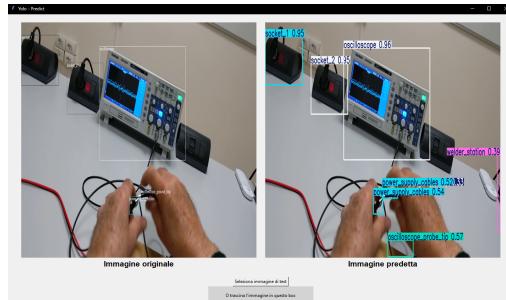


Figure 40: Esempio di risultato dopo aver selezionato un input.

Se l’input selezionato non dovesse far parte del test set (non sarà quindi un’immagine etichettata) allora verrà visualizzata semplicemente l’immagine originale affiancata da quella su cui è stata effettuata inferenza.

## 7 Codice

Il codice nella sua totalità è costituito da vari file:

- **demo.py**
- **DATASET\_training.ipynb x6**
- **CycleGAN\_training.ipynb**
- **CycleGAN\_adaptation.ipynb**
- **testing\_DATASET.ipynb x6**

### 7.1 Breve descrizione dei vari sorgenti

#### 7.1.1 **demo.py**

E' uno script che avvia un'interfaccia grafica con cui poter effettuare inferenza utilizzando YoloV11n con i parametri ricavati dal training dei modelli presentati precedentemente.

#### 7.1.2 **Vari DATASET\_training.ipynb**

Notebook jupyter utilizzati per il training dei modelli YOLO sui vari dataset.

#### 7.1.3 **Vari testing\_DATASET.ipynb**

Notebook jupyter utilizzati per valutare le performance dei modelli YOLO sul test set, per farlo vengono utilizzati i migliori parametri salvati all'interno del progetto.

#### 7.1.4 **CycleGAN\_training.ipynb**

Notebook jupyter che permette il training della CycleGAN.

#### 7.1.5 **CycleGAN\_adaptation.ipynb**

Notebook Jupyter che prende i parametri della CycleGAN e li utilizza per effettuare l'adattamento di tutti i dati sintetici sfruttando il dataset **"fullSynth"**

## 7.2 Altri file presenti nel progetto

Oltre ai codici sono presenti altri file all'interno del progetto:

- **models:** directory contenente i parametri per i vari modelli allenati.
- **scripts:** directory che contiene gli script utilizzati per creare i dataset utilizzati in questo progetto a partire dal dataset originale.
- **.env:** file che raccoglie e rende comuni i path tra i vari sorgenti.

## 7.3 Istruzioni per l'utilizzo

1. Il primo passo è quello di scaricare o clonare il contenuto del seguente [repository](#).
2. Dopo di ciò è sufficiente seguire le istruzioni presenti nel README.md situato nella repository.

# 8 Conclusione

Gli esperimenti condotti hanno evidenziato che l'utilizzo esclusivo di dati sintetici per affrontare un task nel dominio reale non è una soluzione praticabile, poiché le differenze tra i due domini impediscono al modello di generalizzare efficacemente. Tuttavia, è emerso che la combinazione di dati sintetici e reali può rappresentare un vantaggio significativo, riducendo la quantità di dati reali necessari per il training e migliorando al contempo le prestazioni del modello. Inoltre, è stata osservata la potenziale utilità delle tecniche di Domain Adaptation. Anche con un addestramento limitato, l'approccio adottato non ha compromesso in modo significativo le prestazioni del modello. Si può ipotizzare che, con un numero adeguato di epoche di training, i generatori della CycleGAN migliorerebbero sensibilmente, contribuendo a un migliore allineamento tra i due domini e portando ulteriori benefici all'utilizzo combinato dei dati sintetici e reali. In ogni caso, l'utilizzo di una CycleGAN rappresenta solo uno dei molteplici approcci possibili per effettuare Domain Adaptation. Esistono altre strategie che potrebbero rivelarsi ancora più efficaci a seconda del contesto e delle risorse disponibili.