



UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI MATEMATICA E INFORMATICA
CORSO DI LAUREA TRIENNALE IN INFORMATICA

Andrea Bellomo
Matricola : 1000055721

Relazione progetto Machine Learning

Docenti : Giovanni Maria Farinella, Francesco Ragusa

Anno Accademico 2022 - 2023

Indice

1	Introduzione	2
1.1	Problema	2
1.2	Motivazioni	2
2	Dataset	4
2.1	Struttura del Dataset	5
2.2	Caratteristiche dei frame	6
3	Strumenti Utilizzati	8
3.1	ResNet50	8
3.2	Feature Pyramid Network	9
3.3	Faster R-CNN ResNet50 FPN	9
3.4	Metriche di Valutazione	10
3.4.1	mean Average Precision	10
4	Training e Risultati	12
4.1	Training e ambiente di sviluppo	12
4.1.1	Settings	12
4.1.2	Ambiente di sviluppo	13
4.2	Training	13
4.3	Risultati	14
	Conclusione	15
	Bibliografia	16

Capitolo 1

Introduzione

La hand detection, è un problema fondamentale nell'ambito della computer vision. L'obiettivo di questo elaborato è sviluppare un modello di machine learning in grado di individuare con precisione e rapidità le mani umane in immagini. Questa capacità è essenziale per una vasta gamma di applicazioni, tra cui l'interazione gestuale in dispositivi touchless, il controllo di gesti in realtà virtuale, la sorveglianza video avanzata e la robotica collaborativa.

1.1 Problema

Il problema della hand detection è una sfida significativa nell'ambito della computer vision e dell'elaborazione delle immagini. Si tratta di identificare e localizzare la presenza di mani umane in un'immagine o in un video. Questo problema è cruciale in molte applicazioni, che spaziano dalla robotica alla realtà virtuale, dall'interazione uomo-macchina senza contatto alla sorveglianza video avanzata. Il problema presenta diverse sfide, tra cui la variazione dell'illuminazione, la diversità di pose e orientamento delle mani, e la presenza di oggetti o persone in sfondo. Inoltre, è essenziale che il sistema sia in grado di operare in tempo reale per applicazioni come l'interazione uomo-macchina e la guida autonoma basata sui gesti.

1.2 Motivazioni

L'hand detection è fondamentale per migliorare l'interazione tra le persone e la tecnologia, facilitando il controllo e la comunicazione attraverso gesti delle mani. È un campo in rapida crescita che ha un impatto significativo in una vasta gamma di settori, migliorando l'efficienza, la sicurezza e l'esperienza utente in molte applicazioni. Le sfide tecnologiche legate alla hand detection

continuano a stimolare la ricerca e lo sviluppo di nuovi metodi e approcci per affrontare queste esigenze.

Capitolo 2

Dataset

In questo secondo capitolo verrà descritto il dataset utilizzato per realizzazione del task. Il dataset MECCANO [1] è un dataset di video egocentrici creato al fine di studiare le interazioni tra umani e oggetti in contesti simili a quelli industriali. Nello scenario proposto i partecipanti sono stati invitati a costruire un modello di giocattolo di una motocicletta (vedi 2.1) interagendo con strumenti come cacciaviti e chiavi inglesi, ma anche come oggetti di piccole dimensioni come viti e bulloni. Essi eseguono una serie di azioni sequenziali del tipo: prendere la chiave inglese, serrare il bullone o posare la chiave inglese. Nonostante si tratti di una semplificazione rispetto a ciò che si trova in ambienti industriali reali, questo scenario risulta comunque piuttosto complesso.

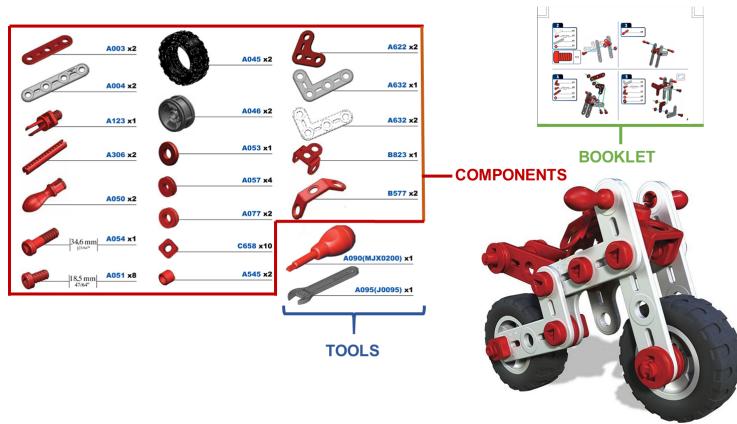


Figura 2.1: modello di giocattolo di una motocicletta(dataset MECCANO)

2.1 Struttura del Dataset

Il dataset è suddiviso in tre sottocartelle principali: "train", "val" e "test", che contengono i frame RGB relativi ai video utilizzati per l'addestramento, la validazione e il test del modello. In totale sono presenti 20 video dai quali sono stati estratti i frame RGB, in particolare sono utilizzati 11 video per la fase di training, 7 per la fase di test e 2 per quello di validation. (vedi figura 2.2).



Figura 2.2: in figura la suddivisione del dataset MECCANO

In aggiunta alle immagini sono presenti le annotazioni dei bounding box delle mani relative al dataset MECCANO. E' presente un file .csv per ciascun video appartenente ai tre sottoinsiemi del dataset MECCANO (train, val e test). Ogni istanza del file .csv è composta da 7 campi (vedi figura 2.3) :

- **filename:** Questo campo contiene il nome del frame o dell'immagine per cui sono fornite le annotazioni dei bounding box. Corrisponde ai singoli fotogrammi estratti dai video nel dataset.
- **file_size:** Questo campo indica la dimensione del frame corrente. Fornisce informazioni sulle dimensioni del frame.
- **file_attribute:** Questo campo contiene informazioni aggiuntive optionali sul file.
- **region_count:** Questo campo rappresenta il numero di bounding box presenti nell'immagine. Indica quanti regioni delle mani individuali sono state annotate nell'immagine.
- **region_id:** Ogni annotazione del bounding box viene assegnata un identificatore unico, rappresentato da questo campo. Questo ID vie-

ne utilizzato per distinguere tra diversi bounding box all'interno dello stesso frame.

- **region_shape_attributes**: Questo campo contiene le coordinate 2D del bounding box. Specifica le coordinate degli angoli superiore sinistro e altezza e larghezza del bounding box.
- **region_attributes**: Questo campo rappresenta la classe del bounding box corrente (SX_hand o DX_hand).

filename	# file_size	# file_attribu...	# region_co...	region_id	region_sha...	region_attr...
00083.jpg	23906	{}	1	0	{"name":"rect", "x":772, "y":221, "width":505, "height":324}	{"class":"SX_Hand"}
00084.jpg	24705	{}	1	0	{"name":"rect", "x":1096, "y":212, "width":181, "height":335}	{"class":"SX_Hand"}

Figura 2.3: in figura due record del file annotazione

2.2 Caratteristiche dei frame

Ogni frame del dataset ha una risoluzione di 1920 x 1080, salvate in formato jpg. In particolare sono presenti :

- 170.000 immagini per la fase di training, in percentuale 59.26%.
- 97.000 immagini per la fase di test, in percentuale 33.82%.
- 33.500 immagini per la fase di validation, in percentuale 11.72%.

Il dataset ha un peso totale di circa 32 gb in memoria. In 2.4 alcuni esempi di frame del dataset MECCANO.

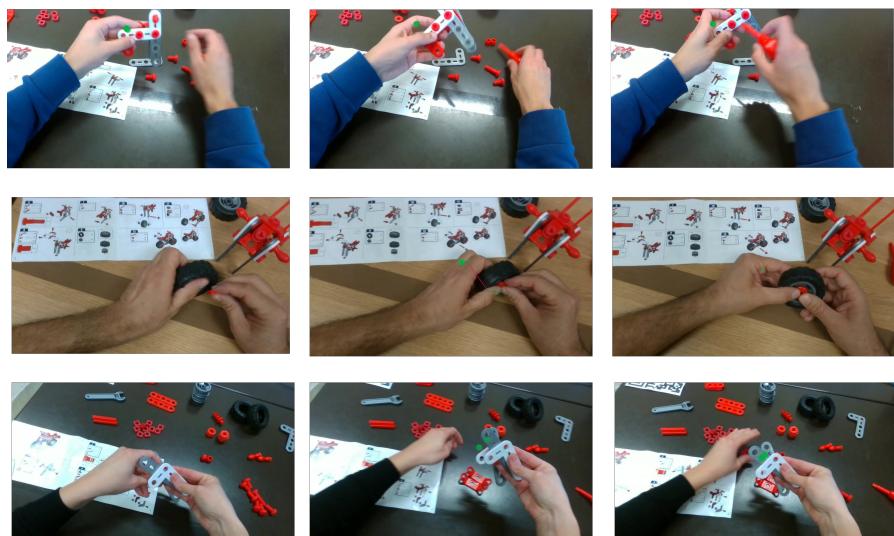


Figura 2.4: in figura un esempio di frame del dataset MECCANO

Capitolo 3

Strumenti Utilizzati

In questo capitolo verrà descritto il modello utilizzato con le relative scelte progettuali al fine di risolvere il task proposto.

3.1 ResNet50

L’architettura del modello sviluppato si basa sull’utilizzo di una rete neurale convoluzionale chiamata ResNet50 [2]. La ResNet50 è una rete neurale convoluzionale (CNN) profonda, sviluppata da Kaiming He et al. nel 2015, addestrata sui pesi del dataset ImageNet. Il termine ”ResNet” deriva da ”Residual Network”, che fa riferimento all’architettura che utilizza blocchi di collegamenti residui. La caratteristica distintiva della ResNet50 risiede nella sua capacità di affrontare il problema del vanishing gradient durante la backpropagation dei pesi attraverso molti strati della rete. Questo problema si verifica quando i gradienti diventano sempre più piccoli man mano che si propagano all’indietro, rallentando l’apprendimento.

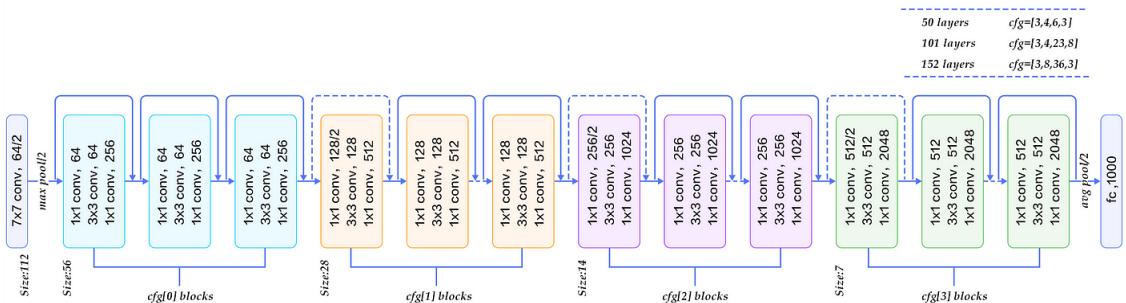


Figura 3.1: architettura ResNet50

3.2 Feature Pyramid Network

Le Feature Pyramid [3] è una tecnica che ha giocato un ruolo fondamentale nella rilevazione di oggetti a scale diverse per molti anni. Tuttavia, è stata abbandonata nei modelli di Deep Learning a causa del suo elevato costo computazionale.

L'idea chiave alla base della Feature Pyramid Network (FPN) è quella di generare una piramide di feature di un'immagine attraverso una rete neurale convoluzionale. Ciò significa che i diversi livelli della piramide non sono semplici sottocampionamenti delle feature map, ma vengono previsti direttamente dalla rete stessa. Questo approccio consente di ottenere rappresentazioni informative a ogni livello della piramide, riducendo al minimo la perdita di informazioni rilevanti.

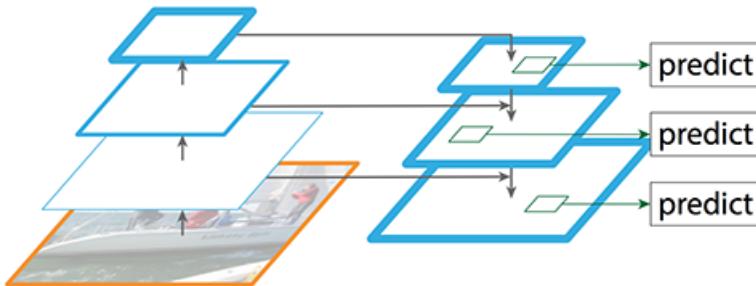


Figura 3.2: architettura Feature Pyramid Network (FPN)

3.3 Faster R-CNN ResNet50 FPN

Il modello 'Faster R-CNN ResNet50 FPN' [4] è una variante dell'architettura 'Faster R-CNN'. Questa è un'architettura utilizzata nell'ambito dell'object detection ed è la combinazione di due componenti principali: una rete neurale convoluzionale, nota come backbone, che funge da estrattore di features, e una rete neurale utilizzata per proporre le Region Of Interest (ROI) che potrebbero contenere un oggetto.

Nella versione originale di Faster R-CNN la babkbone utilizzata è una VCC, e le ROI sono trovate tramite una Region Proposal Network (RPN).

La Faster R-CNN utilizzata è composta da una ResNet50 (Sezione 3.3) come backbone, e da una FPN (Sezione 3.2) come rete per le ROI.

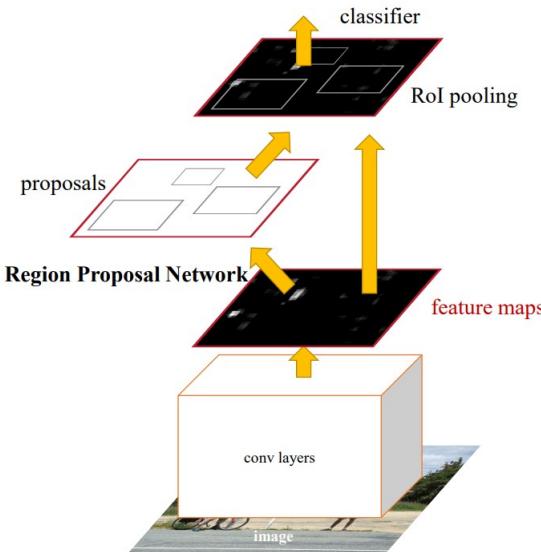


Figura 3.3: architettura 'Faster R-CNN ResNet50 FPN'

3.4 Metriche di Valutazione

Durante la valutazione delle performance del modello addestrato sono state usate diverse metriche di seguito descritte.

3.4.1 mean Average Precision

La mean Average Precision (mAP) si basa sul calcolo dell'intersection over union (IoU), dato un box di groundtruth e un box predetto viene calcolata l'area dell'intersezione fratto l'area dell'unione (vedi Figura 3.4).

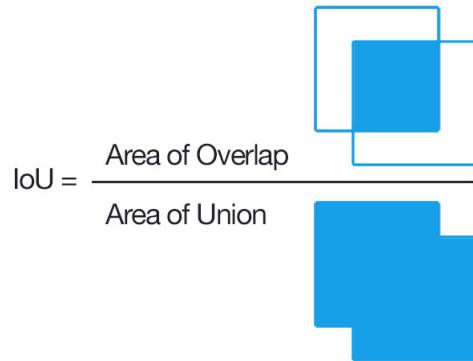
La IoU è utilizzata per classificare il box predetto come: True Positive se è maggiore di una certa threshold (solitamente 0.5); False Positive se è minore della threshold o l'etichetta predetta è diversa dall'etichetta reale; False Negative quando è presente un oggetto ma il modello non riesce ad identificarlo; True Negative non è considerato.

Successivamente è possibile calcolare per ogni classe la Precision come :

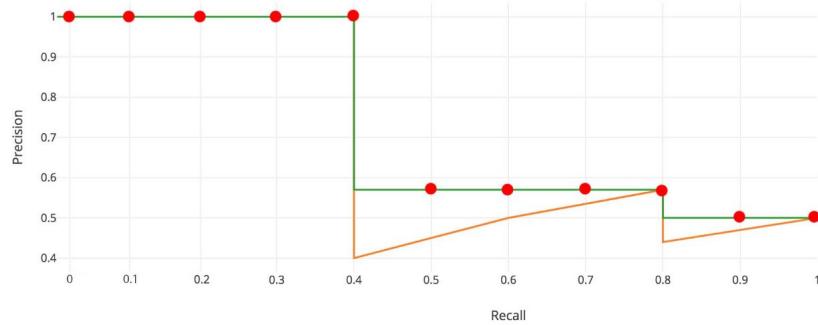
$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

e la Recall come :

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

**Figura 3.4:** Calcolo della IoU.

Calcolando Precision e Recall al variare della threshold dell'IoU è possibile effettuarne il plot, l'area sotto la curva venutasi a creare viene interpolata utilizzando la tecnica degli 11 punti, l'area sotto la curva interpolata è definita come Average Precision (Vedi Figura 3.5). La media dell'Average Precision calcolata per ogni classe è la mAP.

**Figura 3.5:** In figura viene mostrato una curva di Precision Recall e l'interpolazione ottenuta con la tecnica degli 11 punti.

Capitolo 4

Training e Risultati

4.1 Training e ambiente di sviluppo

Di seguito vengono descritte alcune delle informazioni e settings utilizzati per la fase di training come l'utilizzo di callbacks e ulteriori parametri sulla configurazione del training del modello.

4.1.1 Settings

- Numero di epoche: Il modello è stato addestrato per un totale di **26** epoche.
- Dimensione del Batch: Durante il training, è stata utilizzata una dimensione del batch pari a **4**.
- Funzione di Loss: Per l'addestramento del modello, è stata utilizzata una **Multi-Task Loss**, ovvero una combinazione ponderata delle loss di classificazione e di regressione, essa tiene conto contemporaneamente delle prestazioni di classificazione e di regressione del modello.
- Ottimizzatore: Per l'ottimizzazione del modello, è stato impiegato l'algoritmo di ottimizzazione **Stochastic Gradient Descent (SGD)**, con un *learning rate* di : **0.005** e un *momentum* di **0.9**. Questo algoritmo è ampiamente utilizzato nell'addestramento di reti neurali ed è basato sulla discesa del gradiente, il cui obiettivo è individuare i pesi del modello che minimizzano una funzione di costo rispetto ai dati di addestramento.

4.1.2 Ambiente di sviluppo

Per lo svolgimento del progetto, è stato utilizzato l'ambiente di sviluppo messo a disposizione da Kaggle, una piattaforma online che offre risorse e strumenti per lo sviluppo e l'esecuzione di progetti di data science e machine learning. Kaggle offre l'accesso a potenti risorse di calcolo, tra cui le GPU, sebbene con alcune limitazioni. In questo specifico progetto, il modello è stato addestrato e utilizzato sfruttando una **GPU P100**.

Durante il training del modello è stato inoltre utilizzato **WandB** al fine di monitorare e ottimizzare il processo di addestramento. **WandB**, abbreviazione di "Weights and Biases" è una piattaforma di gestione sperimentale per il machine learning. Offre la capacità di registrare in modo efficiente i log durante l'addestramento dei modelli, consentendo di tenere traccia dei dettagli dei test. Inoltre, WandB mette a disposizione potenti strumenti per visualizzare i grafici e le metriche dei modelli, rendendo più facile analizzare i risultati dei test effettuati.

4.2 Training

Di seguito, nella figura 4.1, viene visualizzata la curva di Loss relativa al Training Set. Come si può osservare dal grafico, la curva di Loss mostra un andamento decrescente. È evidente la presenza di numerose fluttuazioni che causano picchi locali. Queste fluttuazioni potrebbero essere attribuibili alle dimensioni ridotte dei batch, poiché in ogni batch potrebbero esserci dati abbastanza diversi rispetto al batch precedente. È importante notare che l'aumento delle dimensioni del batch non è possibile a causa di limitazioni hardware.



Figura 4.1: In figura viene mostrato una curva di Loss sul Training Set

4.3 Risultati

I risultati in tabella 4.1 evidenziano le prestazioni del modello allenato su due differenti set di dati: il training set utilizzato per addestrare il modello e il test set per valutarne l'efficacia. La metrica di "Average Precision" (AP) a differenti livelli di sovrapposizione (IoU) mostra una precisione media del 47,36% nel training set e del 30,35% nel test set a un IoU del 50%, mentre a un IoU del 75% si registra una precisione del 43,25% nel training set e del 27,12% nel test set.

La metrica di "Average Recall" (AR) a diversi intervalli di sovrapposizione (0.50:0.95) evidenzia una Recall media del 41,44% nel training set e del 25,56% nel test set per il caso di massimo rilevamento uguale a 1, mentre per un massimo di rilevamenti pari a 100, si registra una Recall del 42,09% nel training set e del 26,76% test set.

Metrica	Training Set	Test Set
AP@IoU0.50	47,36%	30,35%
AP@IoU0.75	43,25%	27,12%
AR@IoU0.50:0.95@max_dets=1	41,44%	25,56%
AR@IoU0.50:0.95@max_dets=100	42,09%	26,76%

Tabella 4.1: Tabella riassuntiva delle varie metriche sul test set e sul training set

Questi risultati indicano una discreta capacità del modello di rilevare correttamente gli oggetti nel training set, ma una performance inferiore nel test set. Le differenze tra le performance nei due set evidenziano l'importanza della valutazione su dati separati per valutare l'effettiva capacità del modello di generalizzazione.

Conclusione

Nel seguente elaborato è stato analizzato il problema della Hand Detection. Sono stati presentate le reti neurali e le metriche utilizzate, spiegandone l'architettura e l'idea alla base. Tutti gli esperimenti sono stati eseguiti usando una Faster R-CNN ResNet50 FPN pretrainata su ImageNet. Come dataset di riferimento si è deciso di usare il dataset Meccano che mette a disposizione un vasto numero di annotazioni in un ambiente industriale. Dagli esperimenti effettuati si è ottenuto una percentuale di Average Precision di x% sul Training Set e dell'x% sul Test Set. I problemi riscontrati risultano essere molteplici uno tra quali la poca potenza computazionale e la vasta dimensione del dataset. Sviluppo futuri potrebbero prevedere l'utilizzo di altri modelli e hardware apposito, in aggiunta si potrebbe rendere il sistema robusto con l'aggiunta meccanismi di tracking successivamente alla prima detection di mani.

Bibliografia

- [1] Francesco Ragusa, Antonino Furnari, and Giovanni Maria Farinella. Mecano: A multimodal egocentric dataset for humans behavior understanding in the industrial-like domain. *Computer Vision and Image Understanding (CVIU)*, 2023.
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
- [3] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.
- [4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.