**OBJECT DETECTION**

**Descrizione**:

Il nostro progetto si concentra sul rilevamento di oggetti, in particolare veicoli, utilizzando il modello pre-addestrato YOLOv5. Il codice integra vari passaggi chiave, tra cui la gestione dei dataset, la configurazione dell'ambiente e la valutazione delle prestazioni.

L'obiettivo principale è valutare l'efficacia di YOLOv5 su più dataset di immagini, calcolare le metriche di precisione e visualizzare i risultati annotati. Inoltre, il progetto utilizza un dataset creato manualmente, permettendo di testare immagini personalizzate.

**Obiettivi**:

1. Automatizzare il processo di configurazione e addestramento del modello YOLOv5.
2. Valutare l'efficacia di YOLOv5 su dataset di diverse dimensioni e complessità.
3. Calcolare le metriche di precisione e recall per valutare la performance del modello.
4. Visualizzare e salvare i risultati annotati per una revisione visiva.
5. Utilizzare il modello per la rilevazione dei veivoli.

**Strumenti e Tecnologie**:

**- Ambiente:** configurato con Miniconda e varie librerie Python elencate in un file requirements.txt.

**- Framework:** YOLOv5, con configurazioni personalizzate tramite il file data.yaml.

**- Dataset:** automatizzazione del download da Roboflow e gestione di dataset personalizzati.

**- Metriche:** calcolo di precisione, recall e IoU (Intersection over Union).

## **Codice e Integrazione:**

Il codice è progettato per essere modulare, consentendo:  
 - Seed casuale o fisso per esperimenti riproducibili.  
 - Configurazione YAML personalizzabile per specificare percorsi e parametri.  
 - Supporto per dataset personalizzati, incluso il download automatico e l’annotazione.

Questo approccio consente una gestione completa del ciclo di vita del modello, dalla preparazione dei dati all'analisi delle prestazioni.

**Processo**:

Il flusso di lavoro del progetto include i seguenti passaggi:

**1) Preparazione dell'ambiente:** il codice configura l'ambiente con librerie Python e clona automaticamente il repository YOLOv5.  
**2) Download del dataset:** utilizza strumenti come curl e git per scaricare dataset pubblici e personalizzati.  
**3) Addestramento del modello:** il file train.py viene eseguito per addestrare YOLOv5 utilizzando parametri specificati nel file config.yaml. **4) Valutazione delle prestazioni:** vengono calcolate precisione, recall e mAP per ogni classe e per ogni dataset.  
**5) Visualizzazione dei risultati:** i bounding box rilevati e le immagini annotate vengono salvati per analisi future.

**6) Test di predizione:** Viene utilizzato un dataset per il test delle predizioni in base al nostro modello allenato.

**Descrizione dei datasets:**

I primi tre datasets (train, test, valid) sono molto grandi e raccolgono immagini statiche di vari tipi di veicoli, mentre il quarto si tratta di un piccolo dataset creato ad hoc da noi, e annotato tramite il tool “LabelImg”.

Ogni immagine è annotata con le coordinate dei bounding box che delimitano ciascun veicolo, essenziali per l'addestramento e la valutazione di modelli di rilevamento degli oggetti.

**Definizioni:**

Per ogni immagine, il modello YOLOv5 esegue il rilevamento dei veicoli e restituisce una lista di bounding box predetti con coordinate (x1, y1, x2, y2), un punteggio di confidenza e una classe di oggetto.

Le annotazioni di verità a terra sono caricate da un file CSV e contengono un bounding box con coordinate (xmin, ymin, xmax, ymax) e un attributo per identificare la classe.

Per ogni bounding box rilevato, il codice confronta questi box con quelli annotati di verità a terra per quella specifica immagine.

La funzione “calculate\_iou” calcola la metrica IoU (intersection over union) tra il bounding box rilevato e ciascun bounding box di verità a terra, ovvero il rapporto tra l'area di intersezione dei due bounding box e l'area della loro unione.

1. **True Positives (TP)**: Il numero di istanze correttamente classificate come positive, ovvero con l’IoU >= 0.5.
2. **False Positives (FP)**: Il numero di istanze erroneamente classificate come positive, ovvero se un bounding box rilevato non corrisponde a nessuna annotazione di verità a terra (IoU < 0.5).
3. **False Negatives (FN)**: Il numero di istanze erroneamente classificate come negative, ovvero se esistono annotazioni di verità a terra che non sono state associate a nessun bounding box rilevato (IoU < 0.5).
4. **Precision:** La proporzione di vere istanze positive tra tutte quelle classificate come positive.

È calcolata come: Precision = TP / (FP+TP)​.

1. **Recall**: La proporzione di vere istanze positive tra tutte quelle che realmente sono positive.

È calcolata come: Recall = TP / (TP+FN).

1. **mAP (mean Average Precision):** La **mAP** è la metrica che misura la precisione media su tutte le classi del dataset. Si calcola prendendo l'area sotto la curva Precision-Recall per ciascuna classe (**AP**) e poi calcolandone la media su tutte le classi.

* **mAP50:** Valuta le performance del modello considerando un'IoU (Intersection over Union) fissa pari a 0.5 come soglia per definire un rilevamento corretto.
* **Map50-95:** Calcola la media delle precisioni su 10 soglie di IoU, variando da 0.5 a 0.95 con incrementi di 0.05.

**Statistiche del validation dataset in base ai loro parametri:**

Nelle prime cinque rilevazioni, il nostro modello è stato allenato utilizzando le stesse 878 immagini, con le relative annotazioni, contenute in un dataset scaricato dalla rete.

Successivamente, abbiamo effettuato la validazione su un secondo dataset.

Il dataset di validazione utilizzato per le prime cinque rilevazioni conteneva 250 immagini ed era anch’esso scaricato dalla rete.

Questo dataset poteva includere immagini duplicate, al fine di verificare la costanza del modello nelle rilevazioni.

Per la sesta rilevazione, il modello è stato allenato con lo stesso dataset utilizzato nelle rilevazioni precedenti, ma la validazione è stata eseguita su un dataset creato e annotato appositamente da noi.

Per ogni rilevazione abbiamo utilizzato il seed numero 12345.

**Come cambiare i parametri del training per le rilevazioni:**

Nel file config.yaml, è possibile modificare i parametri di addestramento per ottimizzare il modello YOLOv5 in base alle esigenze specifiche del progetto. I parametri principali da configurare includono:

* **Percorsi dei dataset:**
  + train\_data\_path: Specifica il percorso delle immagini utilizzate per l'addestramento.
  + val\_data\_path: Indica il percorso delle immagini di validazione, utilizzate per monitorare le prestazioni del modello durante l'addestramento.
  + detect\_data\_path: Definisce il percorso delle immagini di test per la rilevazione degli oggetti.
* **Numero di epoche (epochs):**  
  Questo parametro controlla il numero di iterazioni complete sull'intero dataset di training. Ad esempio, con epochs: 2, il modello eseguirà due iterazioni sull'intero set di dati di addestramento.
* **Dimensione delle immagini (img\_size):**  
  Indica la dimensione di input delle immagini utilizzate per l'addestramento. Con img\_size: 128, ogni immagine sarà ridimensionata a 128x128 pixel.
* **Dimensione del batch (batch\_size):**  
  Controlla il numero di immagini processate simultaneamente durante l'addestramento. Con batch\_size: 64, il modello elaborerà 64 immagini per volta.
* **Seed per la riproducibilità:**
  + use\_fixed\_seed: Se impostato su true, il training utilizzerà un seed fisso per garantire la riproducibilità dei risultati. Di default il valore del seed è specificato in seed: 12345.
  + Se use\_fixed\_seed è impostato su false, un seed casuale sarà generato automaticamente a ogni esecuzione.

Questi parametri possono essere adattati modificando il file config.yaml prima dell'avvio dello script di addestramento. Cambiando i percorsi dei dataset, le impostazioni relative alle epoche, alla dimensione delle immagini e al batch size, è possibile ottimizzare il processo di training in base alle risorse hardware disponibili e agli obiettivi del progetto.

Una volta avviato lo script, con le nostre configurazioni personalizzate, il file data.yaml sarà generato o aggiornato, se già presente da una precedente rilevazione, in modo automatico. Questo file conterrà i percorsi per i dataset relativi al training e alla validazione del nostro modello, caricati automaticamente dal file config.yaml precedentemente configurato, e il numero e il tipo di classi prese in considerazione per le nostre rilevazioni, specificate direttamente nel codice script.py e lette grazie alla funzione “prepare\_data\_yaml”.

**Risultati delle rilevazioni:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Statistiche** | **1° Rilevazione** | **2° Rilevazione** | **3° Rilevazione** | **4° Rilevazione** | **5° Rilevazione** | **6° Rilevazione** |
| Total images processed | 250 | 250 | 250 | 250 | 250 | 16 |
| Time of validation | 0,041 h | 0,747 h | 0,031 h | 0,121 h | 1,679 h | 1,618 h |
| Number of epochs | 5 | 5 | 5 | 15 | 15 | 15 |
| image size | 128 | 640 | 128 | 128 | 640 | 640 |
| Batch size | 16 | 16 | 64 | 16 | 8 | 8 |
| Precision | 0,224 | 0,506 | 0,257 | 0,492 | 0,585 | 0,826 |
| Recall | 0,369 | 0,582 | 0,150 | 0,557 | 0,582 | 0,782 |
| mAP50 | 0,273 | 0,526 | 0,115 | 0,542 | 0,592 | 0,900 |
| mAP50-95 | 0,158 | 0,300 | 0,0466 | 0,361 | 0,409 | 0,424 |

### **Analisi delle metriche di performance:**

1. **Miglior Precision (0,585):** Si osserva nella 5° rilevazione, che utilizza:
   * Image size: 640.
   * Epochs: 15.
   * Batch size: 8.
2. **Miglior Recall (0,582):** Ottenuto nella 5° e 2° rilevazione.
3. **Peggior Precision (0,224):** Nella 1° rilevazione:
   * Image size: 128.
   * Epochs: 5.
   * Batch size: 16.
4. **Peggior Recall (0,150):** Nella 3° rilevazione, inoltre possiamo osservare che in questa rilevazione anche la precision è molto bassa.
5. **Migliore mAP50 (0,592):** Ottenuta nella 5° rilevazione.
6. **Peggior mAP50 (0,115):** Riscontrata nella 3° rilevazione.
7. **Migliore mAP50-95 (0,409):** Nella 5° rilevazione coerentemente con la mAP50.
8. **Peggior mAP50-95 (0,0466):** Nella 3° rilevazione, anche qui coerentemente con la mAP50

### **Considerazioni sui parametri:**

1. **Risoluzione delle immagini (Image size):**
   * **Alta risoluzione (640):** Nella 2° e 5° rilevazione, contribuisce a performance migliori.
   * **Bassa risoluzione (128):** Le 1°, 3° e 4° rilevazioni con image size basso mostrano valori di precision e recall inferiori.
2. **Batch size:**
   * **Batch size elevato (64):** La 3° rilevazione mostra chiaramente una penalizzazione delle performance (peggior precision e recall).
   * **Batch size ridotto (8):** Nelle 5° rilevazione, consente le migliori performance (alta precision e recall), anche se richiede tempi di validazione maggiori.
3. **Numero di epoche (Number of epochs):**
   * Le prime tre rilevazioni utilizzano solo 5 epoche e presentano performance generalmente inferiori rispetto alle ultime due rilevazioni, che sfruttano 15 epoche.
   * Un numero maggiore di epoche migliora l'apprendimento del modello.
4. **Tempo di validazione:**
   * Il tempo varia significativamente (da 0,031 h nella 3° rilevazione a 1,679 h nella 5° rilevazione).
   * La risoluzione delle immagini (640x640) e batch size più piccoli contribuiscono a un aumento dei tempi di validazione.

### **Conclusioni e raccomandazioni:**

* **Miglior configurazione complessiva:**La 5° rilevazione dimostra che un compromesso tra image size alta (640x640), un batch size ridotto (8), e un numero di epoche adeguato (15) porta alle migliori performance complessive (precision: 0,585, recall: 0,582).

Sebbene la 6° rilevazione abbia mostrato le migliori performance, questi risultati devono essere interpretati con cautela a causa del **numero ridotto di immagini processate**.

I risultati della 6° rilevazione non possono essere considerati pienamente rappresentativi a causa del dataset ridotto. Il confronto tra rilevazioni deve tenere conto di questa disparità per evitare deduzioni fuorvianti.

* **Trade-off performance/tempi:**Un'immagine con risoluzione di 640 migliora precision e recall, ma aumenta il tempo di validazione.  
  Per ridurre i tempi senza compromettere troppo la qualità, si potrebbe adottare:
  + **Image size:** 640.
  + **Batch size:** 16.
  + **Number of epochs:** 10.
* **Batch size e memoria:**È importante bilanciare il batch size con la memoria disponibile. Valori troppo alti (es. 64) riducono la qualità del modello, mentre valori troppo bassi (es. 8) aumentano i tempi.

In sintesi, la configurazione ottimale per il modello dovrebbe bilanciare la risoluzione delle immagini, il numero di epoche e il batch size in base alle risorse disponibili e all'obiettivo desiderato (massimizzare precision e recall o minimizzare i tempi di validazione).