



# **Università politecnica delle Marche**

Facoltà di Ingegneria

Corso di Laurea Triennale in Ingegneria Edile

## **Digitalizzazione automatica della segnaletica di emergenza tramite l'impiego di reti neurali**

Automatic digitalization of emergency signs using neural networks

Relatore:

Prof. Massimo Lemma

---

Candidato:

Giulio Bizziccheri

---

Correlatore:

Dott. Alessandra Corneli

---

Correlatore:

Dott. Massimo Vaccarini

---

## --- Indice ---

<b>1. Introduzione</b>	1
<b>2. Stato dell'arte</b>	3
2.1 La digitalizzazione	3
2.2 Digitalizzazione nel campo dell'edilizia	
2.3 Le reti neurali, applicazioni	5
2.3.1 Acquisizione automatizzata del danno sismico da immagini	5
2.3.2 Supporto alle operazioni di manutenzione	8
2.3.3 Utilizzo delle reti neurali per il censimento del verde urbano	9
2.4 Reti neurali	12
2.4.1 YOLO	14
2.4.2 YOLOv1	16
2.4.3 YOLOv2	19
<b>3. Metodologia</b>	23
3.1 Rete e software di allenamento	23
3.2 Formazione del dataset	23
3.2.1 Raccolta delle immagini	24
3.2.2 Aumento delle immagini	26
3.2.3 Creazione delle Bounding boxes	28
3.3 Allenamento di una rete	30
3.4 Validazione di una rete	33
<b>4. Risultati</b>	39
4.1 Dataset utilizzati	39
4.2 Allenamenti e validazioni	42
4.3 Risultati degli allenamenti	49
<b>5. Conclusione</b>	50
<b>6. Sitografia</b>	52
<b>7. Ringraziamenti</b>	53

# 1. Introduzione

Il mondo della tecnologia negli ultimi decenni ha portato numerose innovazioni, aumentando le proprie potenzialità e cambiando il modo di operare in molti settori. A partire dalla vita quotidiana di ogni singolo cittadino, passando per il rinnovamento di alcuni processi industriali fino alla guida autonoma delle auto. Nel settore dell'edilizia tuttavia si può notare come ancora non ci sia stato un vero e proprio inserimento deciso della tecnologia all'interno dei propri processi. Infatti, ad oggi è evidente come il mondo dell'edilizia sia legato principalmente alle abilità del professionista e come i software abbiano solamente in parte facilitato alcune procedure lasciando in ogni caso all'operatore una massiccia mole di lavoro in termini di supervisione e controllo.

Le operazioni di rilievo e di inventario essendo operazioni che richiedono lunghi tempi e alti costi hanno catalizzato l'attenzione di numerosi studi e innovazioni che hanno portato però solo in parte ad una automatizzazione di queste. Infatti, un certo numero di tecnologie ora in uso, a partire dal GPS fino ai laser scanner, sono state sviluppate negli ultimi anni. Queste tecnologie se da una parte forniscono una cattura automatica dei dati, dall'altra lasciano all'operatore il grosso compito di gestione e controllo degli stessi. Inoltre, utilizzando laser scanner, vengono raccolti sia dati mirati che non mirati lasciando quindi al professionista una complessa procedura di revisione e un lungo lavoro di post-elaborazione con l'obiettivo di tradurre le nuvole di punti in un database organizzato di oggetti. Quindi, è evidente come l'automazione di alcuni processi e la gestione dei dati direttamente in sito potrebbe ridurre, o addirittura cancellare, il tempo di post-elaborazione facilitando notevolmente le operazioni di rilievo e di inventario.

Un altro aspetto da tenere in considerazione è il modo in cui le informazioni vengono archiviate dopo essere state raccolte. Poiché il BIM sta diventando uno standard di progetto, in una prospettiva di digitalizzazione dei processi edili, l'automatico rilevamento del tipo di oggetto, la conseguente creazione automatica dell'elemento digitale e la possibilità di arricchire automaticamente il modello BIM dell'edificio porterebbe a vantaggi in termini di tempo e affidabilità dei dati di notevole importanza. L'obiettivo di questa ricerca nasce proprio seguendo questa nuova prospettiva e con l'intenzione di rendere questi processi automatizzati. Al fine di perseguire questo obiettivo è stato scelto l'utilizzo dell'intelligenza artificiale e nello specifico delle reti neurali per il riconoscimento di oggetti.

Reti neurali appunto che hanno spinto numerosi esperti, specializzati in diversi settori, ad investire risorse e studi per un loro sviluppo. Dalla guida autonoma, agli assistenti vocali, dai nostri smartphone alla domotica di casa nostra, l'applicazione di queste nuove tecnologie sta cambiando le nostre abitudini e i nostri modi di operare. Se questo scenario in svariati ambienti è già ben sviluppato, nel campo dell'edilizia invece sta muovendo i suoi primi passi. Ad oggi infatti grazie all'addestramento con specifici dataset di reti neurali, si è già in grado di riconoscere attraverso la computer vision oggetti di uso comune a partire dalle auto fino ad arrivare agli esseri umani. Sempre di più nel mondo della ricerca la si sta utilizzando come metodo di riconoscimento di oggetti, allenandola per scopi specifici.

Questo lavoro di tesi è volto proprio allo sfruttamento di questa capacità delle reti neurali all'interno di processi relativi al mondo delle costruzioni. Il riconoscimento automatico di oggetti potrebbe essere infatti sfruttato in maniera duplice: da un lato in un primo momento per l'arricchimento informativo di modelli BIM, in un secondo momento per il riconoscimento di oggetti sui quali è necessario operare in sede di manutenzione degli edifici. Infatti, se da un lato la fase di inventario, come già descritto precedentemente, è una delle

procedure più impegnative in termini di tempo e costi, dall'altro la fase di supporto alla manutenzione è un altro scenario interessante sul quale si sta ponendo l'attenzione. Lo scenario futuro in cui questa tesi si colloca sarebbe quello di poter dotare un addetto alla manutenzione di uno strumento per la visualizzazione della realtà mixata (es. Microsoft Hololens) che è in grado di sovrapporre una rappresentazione digitale al mondo reale. Strumenti come questo abbinati con il riconoscimento automatico di oggetti saranno in grado di portare direttamente in sito tutta la documentazione necessaria alle operazioni (es. procedure, check-list, manuali e dati di utilizzo) abbinandola direttamente all'oggetto a cui fa riferimento.

L'obiettivo di questa tesi è quindi quello di addestrare la rete al riconoscimento di tutti quegli oggetti facenti parte del sistema di segnalazione delle uscite di emergenza in modo sicuro ed affidabile. Questa rete potrà poi essere utilizzata per il riconoscimento di oggetti durante la fase di rilievo o per la visualizzazione di informazioni durante operazioni di manutenzione. Al fine del raggiungimento dei nostri obiettivi si è utilizzata la rete YOLO tiny v2 allenandola con darknet seguendo gli step elencati:

1. Raccolta di immagini per la formazione del dataset
2. Incremento delle immagini
3. Creazione delle bounding boxes
4. Allenamento della rete
5. Validazione della rete

Lo scopo che si vuole raggiungere è dare la possibilità alla rete, una volta ultimato l'allenamento, di identificare all'interno dell'immagine l'oggetto desiderato, fare in modo che gli abbini l'etichetta giusta e che lo riquadri in modo corretto. Per ottenere questo, sono stati creati diversi dataset di immagini con i quali è stata allenata la rete in modo da poter confrontare i risultati ottenuti e scegliere quella più adatta al nostro scopo. I valori su cui si sono basate le nostre valutazioni in termini di affidabilità e precisione sono stati mAP, precision, recall e F1. Una volta terminato il processo la rete è risultata in grado di elaborare l'immagine di input e restituire come output la stessa immagine arricchita delle informazioni a noi necessarie (Fig. 1).

Figura 1

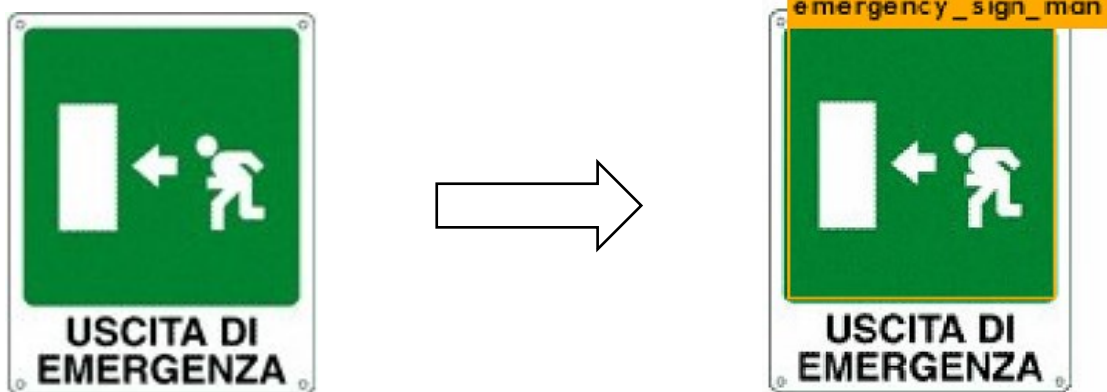


Immagine originale, utilizzata come input per l'elaborazione.

Immagine restituita dalla rete dopo l'analisi.

## 2. Stato dell'arte

### 2.1 La digitalizzazione

La digitalizzazione è il processo di conversione che, applicato alla misurazione di un fenomeno fisico, ne determina il passaggio dal campo dei valori continui a quello dei valori discreti. Tale processo viene oggi comunemente sintetizzato nei termini di passaggio dall'analogico al digitale nell'audio, video, immagini e testo. L'obiettivo di fondo, identificato da alcune avanguardie della ricerca fin dagli anni trenta del secolo trascorso, è quello di riorganizzare la conoscenza in modo sempre più efficiente, semplificando la selezione delle notizie in un mondo sommerso dalle informazioni. In una estrema opera di semplificazione del processo, si potrebbe affermare che quell'obiettivo utopistico ha generato gli ipertesti, il PC, Internet.

In qualsiasi professione, dal commercialista, al consulente del lavoro, ad un avvocato, ad un medico, fino ad una piccola o media impresa, raccogliere, conservare e proteggere una mole considerevole di dati e di informazioni risulta l'aspetto più importante e più difficile da gestire. Se si considera inoltre il tempo perso a reperire informazioni di cui non si conosce o non si ricorda l'ubicazione, a recuperare dati conservati in archivi personali, a trasmettere per posta contenuti di valore tra colleghi lontani o sottoposti a time zone differenti, a rispondere a richieste di clienti immediate e urgenti, a cui si possono poi aggiungere i costi di magazzino, manutenzione e gestione, le difficoltà logistiche in caso di trasloco, i rischi di furto, usura o distruzione nel tempo, si può facilmente notare come sia necessaria la possibilità di attivare meccanismi di archiviazione sostitutiva e backup, attraverso la digitalizzazione dei dati: un nuovo modo di gestire le informazioni in grado di far risparmiare tempo e denaro, aumentando la produttività aziendale.

La dematerializzazione presenta dunque innumerevoli vantaggi concreti, tra cui la riduzione immediata dei quantitativi di carta da produrre, conservare e proteggere, con evidenti benefici anche per l'ambiente: attraverso l'uso di fogli di calcolo e documenti testuali online si comprimono notevolmente i tempi e i costi di stampa, si spreca meno inchiostro, si smaltiscono meno toner, si abbattano meno alberi.

La condivisione dei dati su Cloud consente altresì di accedere ai dati in modo semplice e veloce, semplificando i meccanismi di ricerca (capita molto spesso di dover reperire con urgenza dati immagazzinati in archivi personali o dispositivi locali) e riducendo il rischio di perdita di informazioni, non più legate ad un dispositivo o contenitore materiale, ma conservati in rete e protetti da sistemi di duplica e backup di emergenza.

La ricerca dei dati poi diventa comoda, da qualsiasi dispositivo la si esegua: la digitalizzazione e gestione integrata consente di accedere ai dati ovunque e in qualunque momento, ottimizzando tempi di lavoro e processi decisionali, soprattutto per le aziende che operano in Stati diversi e sottoposte a fusi orari differenti. Per accedervi, basterebbe un cellulare, un tablet o altro dispositivo mobile. E una rete Cloud sicura e protetta.

La digitalizzazione dei dati è dunque necessaria perché le aziende possano restare al passo con i tempi e le moderne tecnologie, ottimizzando i processi di acquisizione e gestione delle informazioni, con evidenti vantaggi competitivi rispetto alla concorrenza. Si pensi, ad esempio, all'automatizzazione delle attività di data entry e alla possibilità di condividere in tempo reale report e statistiche in modo sicuro in tutte le parti del mondo.

## 2.2 Digitalizzazione nel campo dell'edilizia

L'importanza della digitalizzazione è quindi racchiusa nella possibilità di organizzare dati, visualizzarli ed elaborarli con operazioni semplici ed immediate. Nel campo dell'edilizia la digitalizzazione ha aperto lo scenario a tantissime applicazioni che fino a prima erano pura utopia. Essere passati ad un progetto redatto in modo digitale tramite l'utilizzo di computer e software ha stravolto completamente il mestiere del tecnico progettista garantendogli un maggior controllo dei dati ed una maggiore gestione con la possibilità di lavorare in team e condividere in modo quasi istantaneo i propri dati con altri professionisti.

La digitalizzazione del settore delle costruzioni è una realtà inevitabile e sempre più concreta. Tutta la filiera è coinvolta nella radicale trasformazione del "modus operandi", sia in termini di gestione che di operatività.

Sistemi SARP (Sistemi Aeromobili a Pilotaggio Remoto) che volano sul cantiere monitorando lo stato di avanzamento lavori; sensori applicati a oggetti e a persone che collegate a piattaforme di informazione gestiscono la sicurezza dei lavoratori; robot che lavorano in cantiere o la stampa 3D che realizza in scala reale intere abitazioni; smartphone che permettono agli stakeholders di scambiarsi e condividersi informazioni sul cantiere. Questi sono solo alcuni degli elementi che contraddistinguono il Cantiere 4.0.

Sostanzialmente però, prima di tutto c'è il BIM, la metodologia operativa che permette di condividere informazioni tra tutti gli attori coinvolti nel processo: committente, progettista, impresa, fornitori e manutentori. L'acronimo di BIM è Building Information Modeling: ovvero, modello di un edificio con informazioni di progetto. Il BIM è dunque una metodologia che semmai comprende più software adatti alla progettazione ed anche alla costruzione, utilizzati per le sue varie fasi di sviluppo. Tutto ciò che concerne il settore edile e delle costruzioni viene pertanto, proprio per mezzo del BIM, raccolto e combinato in maniera digitale. Il BIM non è un software, ma una metodologia, complessa e innovativa, essenziale per il settore edile, architettonico ed infrastrutturale. Il BIM è inoltre un metodo integrato di progettazione la cui unicità risiede nella capacità di poter raccogliere, unificare e combinare tutti i dati che riguardano la pianificazione della progettazione di un edificio. Inoltre il BIM non serve solo per la pianificazione della costruzione di un edificio, ma è anche un metodo essenziale per il controllo, la verifica e la riduzione degli errori in fase di esecuzione per un ferreo metodo di controllo preventivo del modello. Un progetto può essere totalmente realizzato anche con l'ausilio di questa metodologia che, come specificato, rende possibile l'unione delle "forze" di un elevato numero di categorie professionali tecniche nel settore  
edile.

Nel dettaglio, quindi all'interno del settore edile, il BIM è molto importante perché permette un corretto sviluppo ed una adeguata gestione, senza errori, del progetto multidisciplinare a cui possono quindi partecipare in maniera più agevole tutte le parti coinvolte nel progetto. La multidisciplinarietà è quindi un fattore caratterizzante del BIM: grazie ad esso è possibile la condivisione e il continuo controllo di processo tra i vari soggetti coinvolti dalla committenza alla ditta esecutrice.

## 2.3 Le reti neurali, applicazioni

I settori in cui l'utilizzo delle reti neurali rappresenta una realtà affermata sono numerosi ed ormai è quasi impossibile elencarli tutti. I principali sono:

- finanza, con numerose applicazioni: previsioni sull'andamento dei mercati inclusi quelli valutari, analisi del rischio di credito, analisi del portafoglio, etc;
- **riconoscimento ed elaborazione delle immagini;**
- analisi del parlato e riconoscimento vocale;
- simulazione di sistemi biologici, da quelli intracellulari alle reti neurali;
- diagnosi mediche, inclusi i referti di TAC e risonanze magnetiche;
- robot steering;
- controllo di qualità su scala industriale;
- data mining;

Come abbiamo notato quindi l'utilizzo delle reti neurali potrebbe avere sviluppo in qualsiasi ambito di applicazione ed in qualsiasi settore. Infatti dietro questa nuova tecnologia si celano grandissime potenzialità alcune delle quali sono già state sfruttate in diversi casi d'uso che adesso andremo ad analizzare.

### 2.3.1 Acquisizione automatizzata del danno sismico da immagini

È ampiamente noto quanto siano dispendiose, sia in termini di costi che di risorse umane, le attività necessarie alla restituzione del quadro fessurativo di un edificio esistente danneggiato o semplicemente che ha subito degrado nel corso della sua esistenza. Difatti, prima di ogni intervento, è necessaria una fase di indagine dettagliata basata su rilievi in sito. Tale fase è innanzitutto volta a validare gli elaborati grafici dell'edificio, quando presenti, e in secondo luogo alla raccolta delle informazioni utili alla definizione del quadro fessurativo. Solo a valle di tali attività si può procedere con l'elaborazione manuale di tutti i dati necessari alla definizione e restituzione del livello di danno per un dato edificio.

È proprio in questa circostanza che le tecnologie sviluppate nell'ambito di *Industry 4.0* potrebbero fornire un supporto pratico per il superamento delle suddette criticità. In particolare, in questo articolo, si è proposto il BIM come alternativa alle procedure tradizionali per la gestione informatizzata del danno sismico allo scopo di ridurre i tempi e costi associati alle attività di creazione di un quadro fessurativo e di progettazione di interventi per un edificio esistente. In aggiunta, viene proposta la tecnologia deep learning per l'analisi

delle immagini di cantiere allo scopo di ottenere una maggiore automatizzazione del processo di definizione del quadro fessurativo. In particolare, a partire da immagini fotografiche riprese in sito, si descrive come giungere al riconoscimento automatizzato di lesioni e alla quantificazione delle caratteristiche relative alle stesse, operazioni tutt'oggi manuali e di difficile verifica nella pratica progettuale. La metodologia proposta è applicata a supporto delle procedure di restituzione del livello operativo di un edificio esistente posto in Macerata, danneggiato dal sisma del 2016. Si tratta di un palazzo costruito nel 1860 e dichiarato inagibile a seguito del sisma. Sulla base degli studi e dalle analisi tecniche effettuati in sito, è emersa come probabile strategia l'impiego della metodologia BIM, tecnologia di potenzialità già note nelle operazioni di nuova progettazione. I risultati evidenziano senza dubbio le potenzialità del BIM in merito alla gestione di dati relativi a edifici esistenti danneggiati. La prospettiva di utilizzo del metodo proposto e quella di rendere più affidabili e rapide le valutazioni dello stato di fatto in previsione di interventi di recupero. Tuttavia, la fase iniziale del metodo consiste nella raccolta ed identificazione manuale del danno da inserire nel modello BIM. L'impiego di tecniche di Intelligenza Artificiale può senza dubbio fornire un supporto digitale e rapido in tal senso. Per tale motivo, è stata sviluppata una specifica procedura in ambiente Matlab basata sul Deep Learning per il riconoscimento automatizzato del danno sismico nella forma di fessura.

In particolare, con l'impiego di Matlab e degli annessi Tools, è stato possibile realizzare due reti neurali profonde: l'una allenata per riconoscere lesioni su pareti intonacate e l'altra per riconoscere lesioni e tessitura di muratura in tufo. Si tratta di reti definite come Semantic Segmentation Networks, dove Semantic Segmentation è il termine che descrive il processo con cui si associa ad ogni pixel dell'immagine una ROI (region of interest) Label che rappresenta un'etichetta rispondente ad una specifica classe.

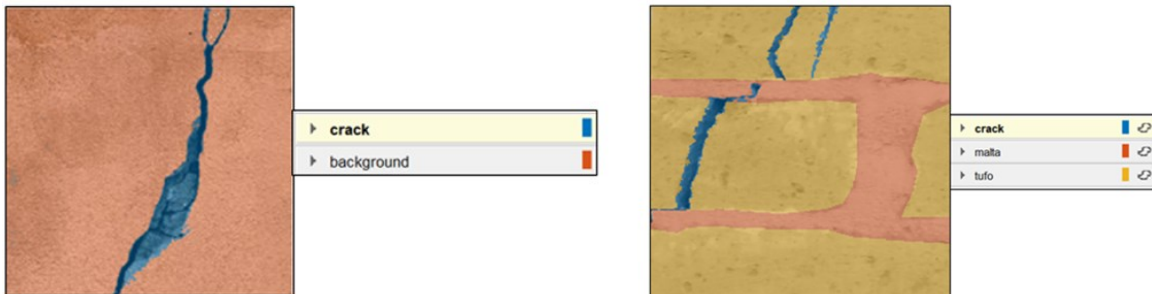
Il flusso di lavoro della procedura automatizzata seguito si articola in step differenti:

1. Scelta delle immagini input: è una delle operazioni principali perché determina la qualità del risultato. Il database, per ciascuna delle due reti, viene creato selezionando accuratamente immagini ad alta definizione che abbiano specifiche caratteristiche. Le stesse sono poi divise in Training Images (necessarie all'allenamento della rete) e Test Images (per la valutazione dell'efficacia della rete).

2. Creazione di ciascuna rete di segmentazione semantica: in questa fase si definisce l'architettura della rete. L'architettura di entrambe le reti considerate è caratterizzata da tre operazioni principali che consentono il rilevamento delle caratteristiche degli oggetti (nel nostro caso, lesioni), dette feature detection layers, quali: Convolution, Pooling, Rectified linear unit (ReLU) cui si aggiungono operazioni finali per la messa a punto ottimale della rete (creazione Full connection layer e Softmax).



3. Addestramento e Test di ciascuna rete: per l'addestramento di ciascuna rete, è necessario considerare delle Training Images e Test Images aventi delle "etichette" con le rispondenti ROI Labels. Nel caso delle pareti interne intonacate sono definite due ROI Labels (Crack, Background); mentre nel caso delle pareti in muratura di tufo sono state definite tre ROI Labels (Crack, Background, Giunto di Malta).



4. La stessa architettura, definita nella fase di creazione della rete, e quindi richiamata per procedere poi all'allenamento della rete stessa, la cui durata è variabile in funzione del numero di immagini di input e della potenza elaborativa del computer a disposizione.

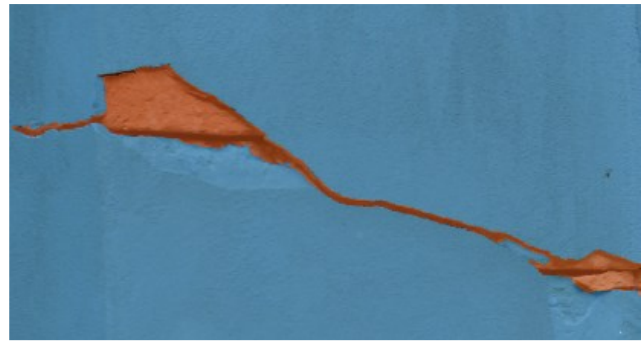
5. Valutazione e controllo dei risultati: si procede con il caricamento della rete addestrata con le immagini di prova. Le etichette stimate sono confrontate con quelle TestLabels, con una successiva valutazione della qualità della previsione. Il controllo dei risultati avviene grazie alla presa visione di specifici parametri quali, Global Accuracy, Mean Accuracy, ecc. Altra importante informazione la si ottiene attraverso la "ConfusionMatrix" capace di restituire in forma tabellare il riscontro tra le classi predette dalla rete e quelle reali derivanti dalla fase 1.

6. Quantificazione dei risultati: specifiche informazioni quantitative inerenti alle superfici lesionate si ricavano con procedimenti completamente automatizzati ricorrendo ad applicativi presenti in Matlab. A partire dalle quantità così ricavate, rispondenti all'unità di misura pixel, sarà sufficiente applicare una semplice conversione per ottenere le rispettive quantità nel sistema di misura desiderato. La successione di step sopra descritta è stata applicata ad entrambe le reti oggetto dello studio. Le iterazioni si considerano concluse quando la segmentazione dell'immagine usata come test (TestImage) e la valutazione della qualità della previsione della rete sono soddisfacenti. Le immagini seguenti illustrano l'applicazione della procedura ad una immagine reale relativa ad una parete intonacata, acquisita presso l'edificio caso studio di Macerata. È possibile apprezzare come la rete riconosca la lesione e ne riesca a valutare le caratteristiche geometriche.

Immagine prima e dopo l'elaborazione della rete.



A sinistra l'immagine della lesione, a destra il riconoscimento automatico del danno.



Il metodo di digitalizzazione del danno sismico proposto nel presente lavoro attesta le significative potenzialità che strumenti digitali avanzati, quali il BIM e Deep Learning, possono avere a supporto degli interventi su edifici esistenti. Infatti, con una dedicata codifica del danno è stato possibile ottenere una completa digitalizzazione del Livello Operativo per un edificio esistente danneggiato dal sisma. Valore aggiunto, inoltre, deriva dall'uso della tecnologia Deep Learning per il riconoscimento automatizzato a partire da immagine. Quest'ultima, insieme con il BIM, ha reso più celeri ed affidabili tutte quelle operazioni necessarie alla corretta progettazione di interventi su edifici esistenti, evitando ritardi in termini di tempo e costo lavoro.

### 2.3.2 SUPPORTO ALLE OPERAZIONI DI MANUTENZIONE

L'architettura proposta trova valide applicazioni anche in un altro caso d'uso: supporto alle operazioni di manutenzione. Gli edifici contengono una serie di strutture che necessitano di manutenzione e alcuni sono sottoposti ad ispezione periodica al fine di verificare la conformità alla normativa e il corretto funzionamento. La localizzazione corretta e immediata degli oggetti richiede tempo e comporta un maggiore possibilità di errore negli edifici complessi. Inoltre, i dati di utilizzo degli oggetti sono spesso non disponibili mentre potrebbero essere assolutamente utili in caso di ispezioni o operazioni che richiedono istruzioni per disattivare componenti o altri tipi di interazioni con le strutture operative.

Con il sistema di riconoscimento degli oggetti di piccole dimensioni, la procedura inizia con l'ispettore che cammina all'interno della struttura con il dispositivo montato sulla testa. Una rete neurale pre addestrata sull'applicazione del dispositivo esegue un'identificazione dell'oggetto semplicemente inquadrandolo all'interno dello schermo e usando lo sguardo come un puntatore. L'applicazione mostra automaticamente l'ologramma dell'oggetto, con tutti i suoi componenti e i documenti a lui associato (procedure, check-list, manuali, dati di utilizzo). Il tecnico ha la possibilità di verificare le informazioni necessarie e compilare i moduli richiesti al fine di completare le sue operazioni.

### 2.3.3 UTILIZZO DELLE RETI NEURALI PER IL CENSIMENTO DEL VERDE URBANO

Il censimento delle aree verdi urbane è uno strumento indispensabile per la loro corretta gestione. A tal fine, l'utilizzo di immagini satellitari ad alta risoluzione è fondamentale per poter conoscere tutto il patrimonio verde delle città, sia questo pubblico o privato. Entrambi, infatti, contribuiscono alla qualità della vita in contesti antropizzati quali quelli urbani. In questo capitolo si presentano i primi risultati ottenuti dalla sperimentazione in atto presso il SINAnet volta al censimento e alla classificazione del verde della città di Roma tramite l'applicazione delle reti neurali artificiali.

Gli obiettivi

Attraverso l'utilizzo delle reti neurali, in questa sperimentazione si è voluto approntare una metodologia innovativa di tipo semiautomatico che faccia uso dell'intelligenza artificiale per ottenere classificazioni di immagini satellitari utili alla conoscenza del patrimonio verde urbano. Il vantaggio di tale metodologia non convenzionale risiede nel fatto che, una volta addestrata la rete per una tipologia d'immagine con determinate caratteristiche (spettrali, temporali, copertura nuvolosa, angolo di ripresa), la stessa rete può essere applicata ad altre immagini con caratteristiche simili, riducendo in tal modo i tempi di elaborazione.

Le reti neurali artificiali, quindi, consentono di ottenere risultati in tempi nettamente più brevi rispetto alle metodologie convenzionali (Sample Angle Mapper, Maximum Likelihood, etc.). Vista l'estensione delle maggiori aree metropolitane italiane, questo vantaggio si tradurrebbe in notevoli risparmi per le amministrazioni.

Le classi utilizzate durante questa sperimentazione sono state:

- 1) Corpi idrici
- 2) Superfici artificiali
- 3) Conifere
- 4) Latifoglie
- 5) Prato
- 6) Suolo nudo

Si sono ottenute quindi 6 differenti classi su cui è stata allenata la rete.

Per ogni classe identificativa sono state selezionate sull'immagine aree di training e di validazione. Tali aree sono state disegnate come ROI (Region Of Interest), rappresentanti un insieme di pixel con risposte spettrali simili per ogni classe. Dopo l'elaborazione di tali ROI è stata lanciata la procedura di addestramento della rete, al termine della quale si è scelta la rete ritenuta più adatta per classificare l'immagine. Per verificare l'attendibilità della classificazione è stata condotta una campagna di rilievi a terra. Sulla scena, avente lati di 7,40x7,64 km, è stata costruita una griglia quadrata con 462 celle di lato 350,8 m, nel sistema di coordinate UTM33N, datum WGS84. È stato selezionato in modo casuale il 10% delle celle, per un totale di 46. In ogni cella sono stati selezionati (sempre in modo casuale) 4 punti, per un totale di 184 punti di controllo. Intorno ad ognuno dei punti sono state create aree circolari di circa 380 m<sup>2</sup> di superficie (raggio di 11 m). Di queste aree ne sono state scelte 12, di cui controllate in campo 10, in quanto 2 sono risultate inaccessibili. Per questo lavoro, è stata addestrata una rete neurale ad operare una classificazione del verde urbano su un'immagine del satellite IKONOS, ricadente nel comune di Roma, avente una superficie di

circa 49 km<sup>2</sup>, acquisita il 12 Luglio 2006, con una risoluzione spaziale di 1 m<sup>2</sup>4 (Figura 2). L'errore medio di classificazione del singolo pixel calcolato sulle 10 aree risulta essere del 6,88 %, ed è quindi più che soddisfacente<sup>25</sup>. La scala è di 1:5.000.



Figura 2 - Immagine IKONOS di partenza su cui è stata addestrata la rete

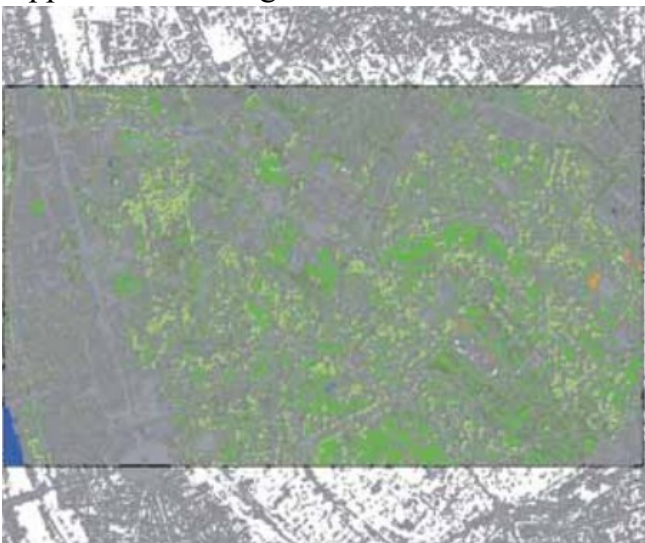


Figura 3 – Riquadro dell'immagine di partenza su cui è stata applicata la rete







Risultati:

La Figura 4 mostra i risultati delle analisi eseguite, con la legenda per l'interpretazione delle relative classi.

Figura 4 – Classificazione dell'immagine rappresentata in Figura 3









Legenda

1	Acqua	
2	Artificiale	
3	Conifere	
4	Latifoglie	
5	Prato	
6	Suolo	

In Figura 5 è riportata la legenda della classificazione e le aree di ogni classe espresse in m<sup>2</sup> ed in percentuale rispetto alla superficie considerata in Figura 3.

Figura 5 – Composizione delle classi di verde e relativa estensione

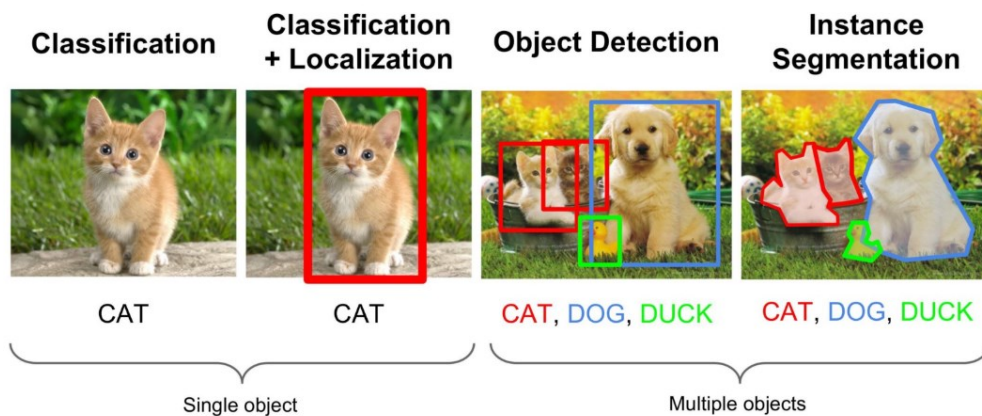
CLASSE		AREA (m <sup>2</sup> )	AREA (%)
Acqua		11.690	0,6
Artificiale		589.872	32
Conifere		354.090	19
Latifoglie		335.441	18
Prato		523.031	28
Suolo		34.878	1,8

La discriminazione tra classe di latifoglie e classe di conifere è stata ottimale, mentre per la classe del prato e del suolo nudo si è dovuto procedere ad un'ulteriore classificazione (Maximum Likelihood), poiché risultanti in classi miste. Il prato, infatti, è stato discriminato tra latifoglie e conifere, mentre il suolo nudo, come previsto, è stato discriminato insieme alle superfici artificiali (essenzialmente tetti), a causa della medesima composizione mineralogica (% in componente argillosa). Dai risultati si evince la presenza di una quota predominante di superfici naturali (65%), diversamente vegetate (conifere, latifoglie, prato). Tale risultato è da imputarsi all'area oggetto di analisi, che comprende una porzione di Villa Borghese. La scelta di quest'area come test di validazione per l'addestramento della rete è stata dettata dall'esigenza di validare il modello delle reti neurali proprio sulla vegetazione e sulle aree verdi.

I risultati ottenuti da questa prima classificazione sono stati ritenuti soddisfacenti e hanno incoraggiato l'utilizzo delle reti neurali artificiale in simili applicazioni. L'affidabilità della metodologia e i vantaggi che essa presenta fanno ritenere auspicabile una sua più larga diffusione a livello nazionale. A tal proposito, APAT ha intenzione di utilizzarla per costituire una banca dati nazionali sul verde urbano delle 24 città italiane capoluoghi di provincia.

## 2.4 Reti neurali

La computer vision è uno dei campi in cui l'Intelligenza Artificiale è in maggiore espansione. Basti pensare alle auto autonome e driverless, dove Tesla ha fatto da apripista, e su cui si stanno ora puntando un quasi tutte le case automobilistiche. Per anni il riconoscimento e la categorizzazione delle immagini sono stati un problema, specialmente considerando la difficoltà di un algoritmo tradizionale a riconoscere lo stesso oggetto in posizioni e angolazioni differenti. Considerando quanto sembra a noi facile e automatico riconoscere e distinguere oggetti da scene visive, rendersi conto dei problemi che si incontrano nel riconoscimento automatico non è così scontato. Innanzitutto dobbiamo distinguere due classi di problemi: la categorizzazione e la localizzazione. La prima, relativamente meno complessa, presenta già difficoltà non banali.



Ci riesce facile per esempio riconoscere una sedia, ma sareste in grado di descriverne una in modo inequivocabile? Potremmo definirlo come un mobile per sedersi con quattro gambe, dei braccioli e uno schienale. Tuttavia, già guardando l'immagine qui sotto si notano diversi problemi: alcune hanno solo tre gambe, alcune addirittura ne hanno solo due, quella rossa di fatto solamente una, quella per ufficio è a rotelle, etc.



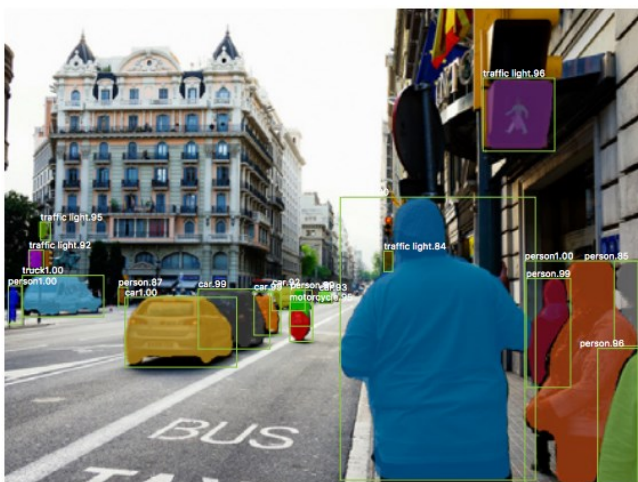
Eppure per noi identificarle tutte come sedie è immediato. Istruire una macchina a riconoscerle presentando tutte le possibili eccezioni è ovviamente impossibile. Di conseguenza un riconoscimento basato su regole è destinato come minimo a produrre risultati insoddisfacenti, pieni di falsi positivi (riconoscimento di sedie dove non ce ne sono) e negativi (sedie non riconosciute per tali).

Senza addentrarci troppo nella storia del riconoscimento automatico degli oggetti, possiamo dire che, prima dell'avvento del deep learning, uno dei tentativi di face recognition di maggiore successo fu quello di Viola-Jones. Questo algoritmo funzionava in modo molto semplice: si generava una sorta di mappa che rappresentava le feature di un viso, tramite migliaia di semplici classificatori binari usando Haar features. Questa mappa veniva poi “cablata” nell'algoritmo usandola per addestrare una SVM come classificatore per individuare la faccia nell'immagine. Era un algoritmo semplice e veloce, tanto che ancora oggi è utilizzato in alcune fotocamere compatte di fascia economica. Queste presentavano però esattamente i problemi sopra descritti, ovvero erano poco flessibili nel riconoscere oggetti presentati con variazioni anche lievi a quelli utilizzati per l'apprendimento.

Più precisi erano algoritmi come quello di Dalal e Triggs, che utilizzava HOG (istogrammi a gradienti orientati) che oltre ai bordi, tiene in conto gli orientamenti dei gradienti in ogni porzione dell'immagine, e SVM per la classificazione. Tuttavia, pur ottenendo risultati molto più precisi del precedente, era decisamente più lento. Inoltre il problema rimaneva la scarsa robustezza e la conseguente difficoltà nel riconoscere immagini con una certa quantità di “rumore” o distrazioni in background. Un altro problema di questi algoritmi era la capacità di riconoscere solo una singola immagine. Inoltre non erano in grado di generalizzare. In altre parole potevano essere “configurati” solo su un genere di immagini (facce, cani, etc), avevano forti difficoltà nei problemi elencati sopra e avevano limitazioni molto stringenti sul formato delle immagini su cui potevano lavorare.

## Deep learning

In effetti per diventare veramente utile, il riconoscimento degli oggetti dovrebbe essere in grado di effettuare riconoscimenti su scene complesse, che sono poi le scene in cui ci imbattiamo nella vita di tutti i giorni. L'esplosione dell'uso delle reti neurali nell'era dei Big Data, e della conseguente popolarità del Deep Learning cambiarono le carte in tavola. In



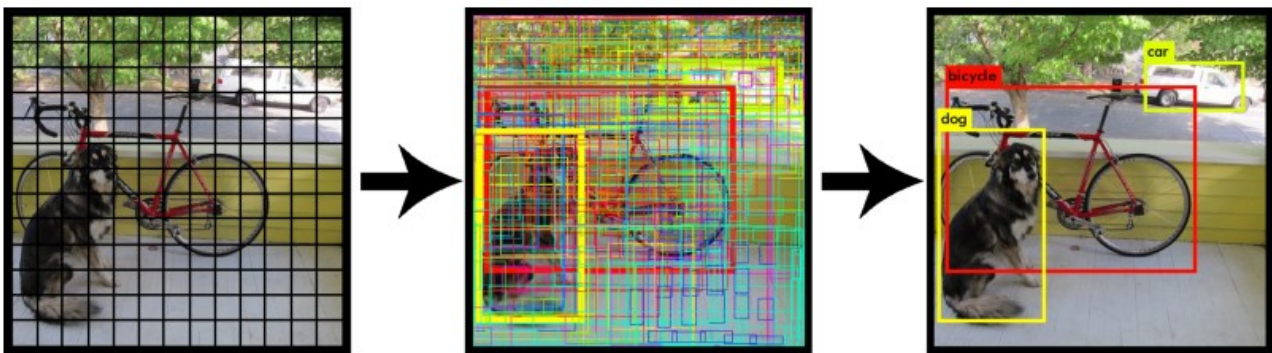
particolare lo sviluppo delle reti convolute diede una forte spinta. Un approccio comune a quasi tutti gli algoritmi (compresi i precedenti) è stato quello della “sliding window”, ovvero scansionare zona per zona tutta l'immagine, analizzandola una porzione (la finestra appunto) alla volta.

Nel caso delle reti convolute (CNN), l'idea è di ripetere il processo con diverse dimensioni della finestra, ottenendo per ciascuna una predizione del contenuto, con grado di confidenza. Alla fine vengono scartate le

predizioni con grado di confidenza più basso.

## 2.4.1 Yolo

Le necessità di oggi vanno molto oltre la semplice classificazione o localizzazione di immagini statiche, oggi la necessità è di avere analisi in tempo reale: nessuno vorrebbe trovarsi a bordo di un'automobile autonoma che impiega diversi minuti (o anche solo secondi) per riconoscere immagini. La soluzione al problema è quella di utilizzare delle reti convolute a passata singola, ovvero che analizzano tutte le parti dell'immagine in parallelo, simultaneamente, consentendo di evitare l'uso della sliding window. Yolo è stato sviluppato da Redmon e Farhadi nel 2015, durante il loro dottorato. Il concetto è di ridimensionare l'immagine in modo da ricavarne una griglia di quadrati. Per ciascuna scala, ogni cella è responsabile della predizione di 3 bounding box, utilizzando 3 anchor box.



Al termine dell'elaborazione vengono mantenute solo le bounding box con la confidenza più elevata, scartando le altre.

La v3 usa come architettura una variante di Darknet, con 106 layer convoluti. Interessante è anche Tiny YOLO, funzionante su Tiny Darknet, e in grado di girare su dispositivi limitati come gli smartphone.

Presentiamo YOLO, un nuovo approccio al rilevamento di oggetti. Il lavoro fatto finora sul rilevamento degli oggetti propone l'esecuzione dei classificatori sull'immagine da rilevare. Al contrario, con YOLO invece inquadrriamo il rilevamento degli oggetti attraverso Bounding boxes e class probabilities. Una singola rete neurale prevede delle Bounding boxes e delle class probabilities direttamente dall'immagine in una unica passata. La nostra architettura unificata è estremamente veloce. La base del nostro modello YOLO elabora le immagini in tempo reale a 45 frame al secondo. Una versione più piccola della rete, Fast YOLO, elabora le immagini fino a 155 fotogrammi al secondo raggiungendo ancora il doppio del mAP di altri rilevatori in tempo reale.

Rispetto ai sistemi di rilevamento all'avanguardia, YOLO commette più errori di localizzazione ma è molto meno probabile che possa prevedere falsi rilevamenti in cui non esiste nulla.

Gli umani danno un'occhiata a un'immagine e sanno immediatamente quali oggetti ci sono nell'immagine, dove sono e come interagiscono. Il sistema visivo umano è veloce e preciso, permettendo a noi di svolgere compiti complessi come guidare. Algoritmi veloci, precisi per il rilevamento di oggetti consentirebbero ai computer di guidare le auto in qualsiasi situazione. Gli attuali sistemi di rilevamento riutilizzano i classificatori per eseguire il rilevamento. Per



rilevare un oggetto, questi sistemi prendono un classificatore per quell'oggetto e lo valutano in varie posizioni e scale in un'immagine di prova. Sistemi come deformable parts models (DPM) utilizzano un approccio a finestra scorrevole in cui il classificatore viene eseguito sull'intera immagine. Approcci più recenti come R-CNN utilizzano metodi diversi per generare prima potenziali riquadri di delimitazione in un'immagine e quindi eseguire un classificatore su questi riquadri. Dopo la classificazione, la post-elaborazione viene utilizzata per perfezionare la BB, eliminare i rilevamenti doppi e reinstallare il box basato su altri oggetti nella scena. Questi processi complessi sono lenti e difficili da ottimizzare perché ogni componente deve essere addestrato separatamente.

YOLO riformula il rilevamento degli oggetti come un singolo problema di regressione, direttamente dall'immagine alle coordinate del riquadro di delimitazione (BB) e class probabilities. Utilizzando il sistema, you only look once (YOLO), su un'immagine, saremo in grado di poter prevedere quali sono gli oggetti presenti e dove sono guardando l'immagine solo una volta.

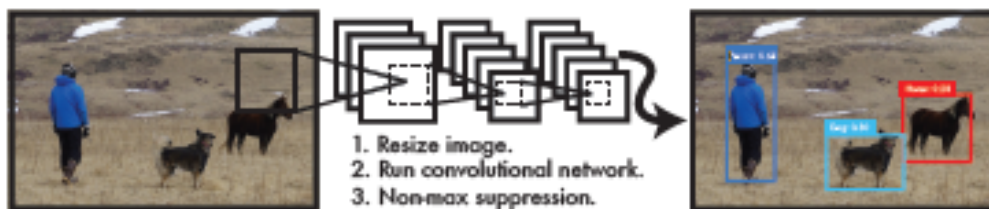


Figura 1

YOLO è notevolmente semplice: vedi Figura 1.

Una singola rete convoluzionale prevede contemporaneamente più scatole di delimitazione (BB) e probabilità di classe per quelle scatole.

YOLO si allena su tutta l'immagine e ottimizza direttamente il rilevamento. Questo modello unificato presenta numerosi vantaggi rispetto ai metodi tradizionali di rilevamento degli oggetti. Innanzitutto, YOLO è estremamente veloce. La rete di base funziona a 45 fotogrammi al secondo senza elaborazione su Titan X GPU e la versione veloce funziona a più di 150 fps. Questo significa che si possono elaborare video in streaming in tempo reale con meno di 25 millisecondi di latenza. Inoltre, YOLO raggiunge più del doppio della precisione media di altri sistemi in tempo reale.

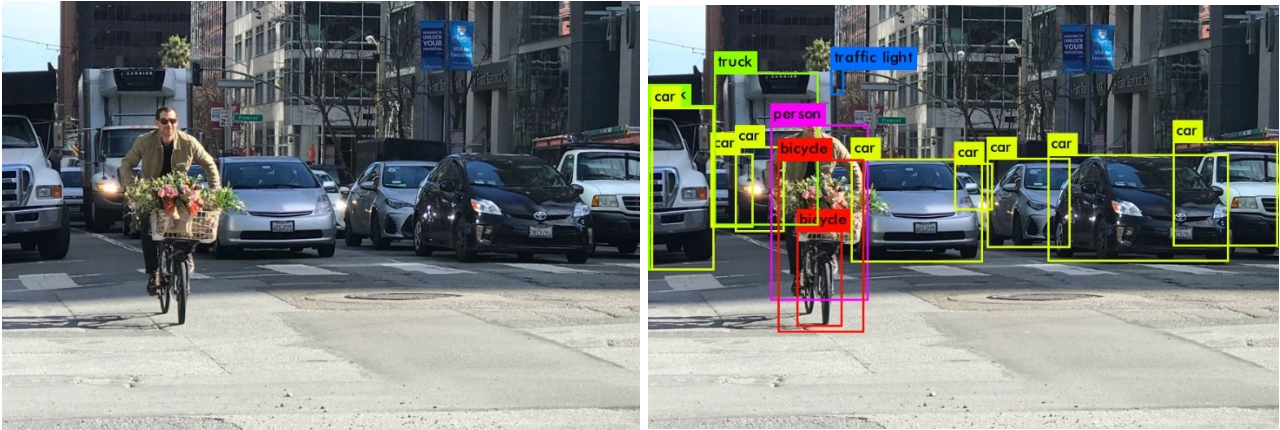
In secondo luogo, YOLO ragiona globalmente sull'immagine quando fa previsioni. A differenza di altre tecniche con finestra scorrevole, YOLO vede l'intera immagine durante l'allenamento e il test di prova in modo da codificare informazioni sulle classi e sul loro aspetto. Fast R-CNN, presenta errori di background in un'immagine perché non riesce a vedere il file in un contesto più ampio. YOLO commette meno della metà del numero di errori di fondo rispetto a Fast R-CNN.

In terzo luogo, YOLO apprende rappresentazioni generalizzabili di oggetti.

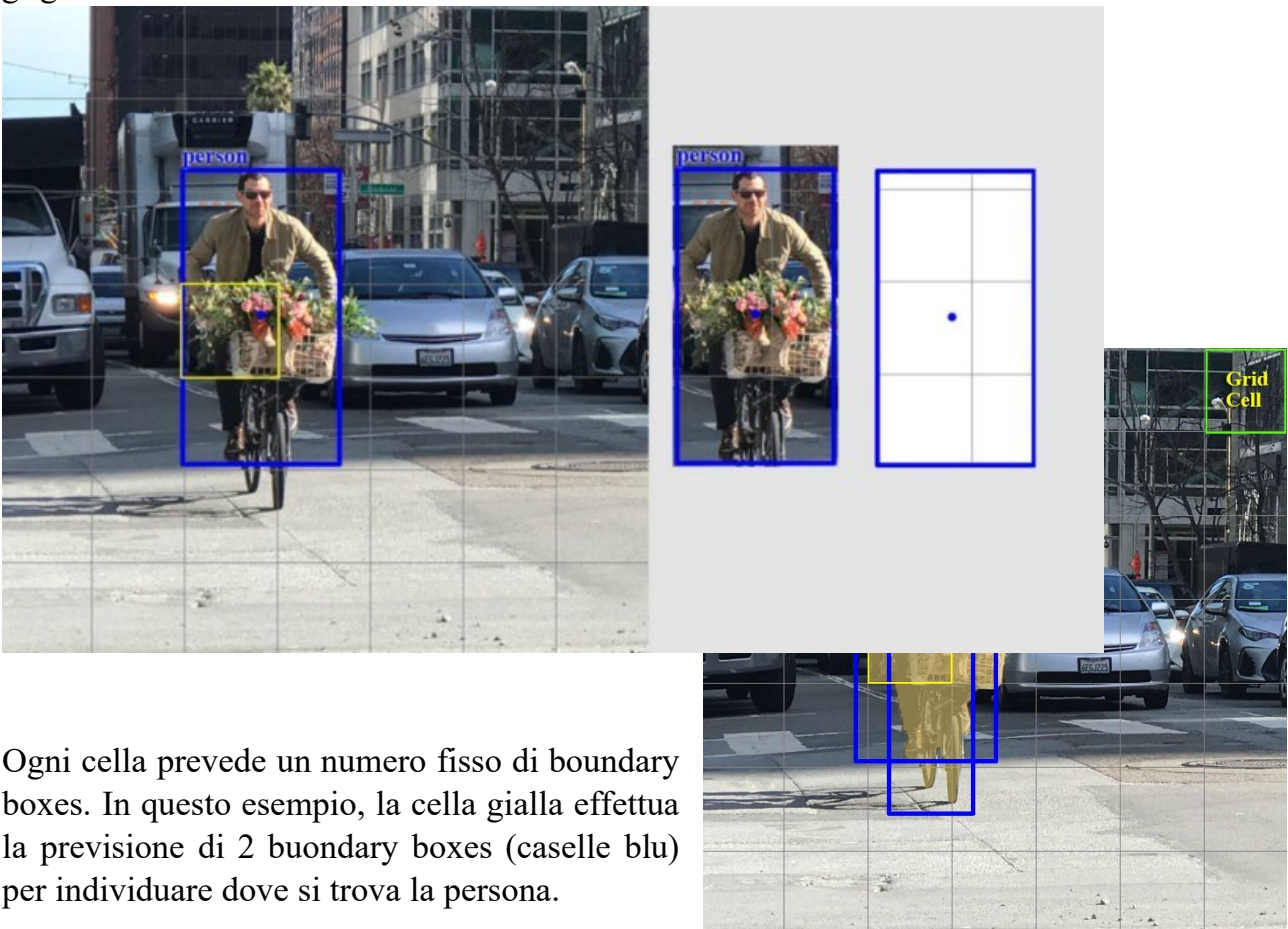
Se addestrato su immagini naturali e testato su opere d'arte, YOLO supera i migliori metodi di rilevamento come DPM e R-CNN con un ampio margine. Poiché YOLO è altamente generalizzabile è meno probabile che si rompa quando applicato a nuovi domini o input imprevisti. Tutto il nostro codice di formazione e test è open source e disponibile online.

## 2.4.2 YOLOv1

Come funziona YOLO ? Cominciamo con la nostra immagine di prova qui sotto a sinistra e a destra la stessa foto dopo essere stata processata da YOLO con indicati gli oggetti rilevati.



Per capire meglio il meccanismo prendiamo questa porzione di foto. YOLO divide l'immagine in una griglia  $S \times S$ . Ogni cella della griglia prevede un solo oggetto. Ad esempio, la cella gialla tenta di prevedere l'oggetto "persona" il cui centro (il punto blu) rientra nella cella della griglia.



Ogni cella prevede un numero fisso di boundary boxes. In questo esempio, la cella gialla effettua la previsione di 2 boundary boxes (caselle blu) per individuare dove si trova la persona.



Tuttavia, questa metodologia limita quanto possono essere vicini gli oggetti rilevati. Per questo, YOLO ha alcune limitazioni su quanto possono essere vicini gli oggetti. Per l'immagine qui sotto, ci sono 9 Babbo Natale nell'angolo in basso a sinistra ma YOLO può rilevarne solo 5.

Quindi YOLO per ogni cella della griglia,

- prevede delle “boundary boxes” BB e ad ognuna di esse viene attribuito un punteggio di confidenza (box confidence score),
- rileva un solo oggetto indipendentemente dal numero di caselle BB,
- prevede un punteggio di “conditional class probabilities” C.

Entriamo in maggiori dettagli. Ogni boundary boxes contiene 5 elementi: (x, y, w, h) e un punteggio di confidenza (“Box confidence score”). Il punteggio di confidenza riflette la probabilità che la casella contenga un oggetto e quanto è precisa la boundary box. Indichiamo con W la larghezza della bounding box e con H l'altezza. X e Y sono le coordinate del centro della bounding box. Quindi, x, y, w e h sono tutti compresi tra 0 e 1. Ogni cella ha 20 conditional class probabilities. Conditional class probability è la probabilità che l'oggetto rilevato appartenga a una particolare classe (una probabilità per categoria per ogni cella). Quindi, la previsione di YOLO ha una forma di  $(S, S, B \times 5 + C) = (7, 7, 2 \times 5 + 20) = (7, 7, 30)$ .

Una volta individuati i parametri precedenti, il sistema calcola un punteggio di confidenza (class confidence score) per ciascuna casella di previsione. Questo parametro viene calcolato come:

$$\text{class confidence score} = \text{box confidence score} \times \text{conditional class probability}$$

Questo misura la precisione sia nella classificazione che nella localizzazione (dove si trova un oggetto).

Per evitare di confondere facilmente i termini di punteggio e probabilità possiamo elencare le definizioni matematiche per ogni singola definizione.

Box confidence score = riflette la probabilità che la casella contenga un oggetto e quanto sia precisa la boundary box

Conditional class probability= è la probabilità che l'oggetto rilevato appartenga a una particolare classe

Punteggio di confidenza della classe= (probabilità che la bb contenga un oggetto) x (la probabilità che l'oggetto appartenga ad una classe).

$$\begin{aligned}\text{box confidence score} &\equiv P_r(\text{object}) \cdot \text{IoU} \\ \text{conditional class probability} &\equiv P_r(\text{class}_i | \text{object}) \\ \text{class confidence score} &\equiv P_r(\text{class}_i) \cdot \text{IoU} \\ &= \text{box confidence score} \times \text{conditional class probability}\end{aligned}$$

where

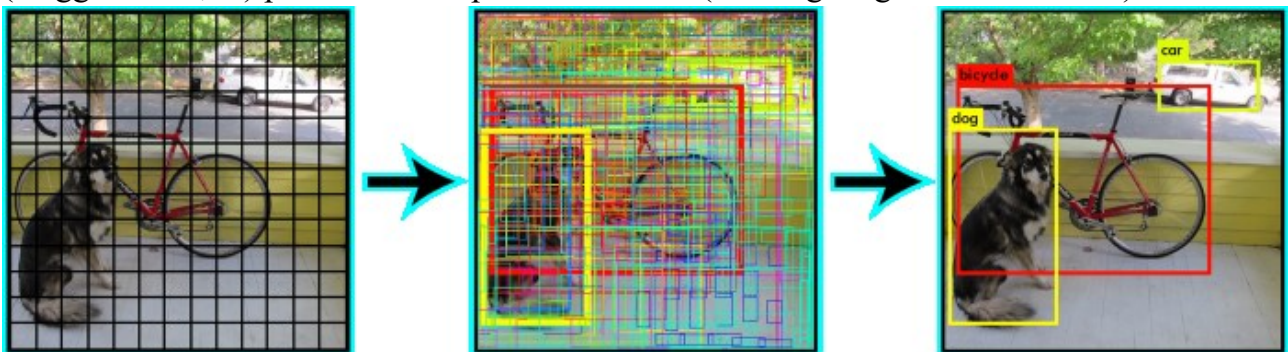
$P_r(\text{object})$  is the probability the box contains an object.

$\text{IoU}$  is the IoU (intersection over union) between the predicted box and the ground truth.

$P_r(\text{class}_i | \text{object})$  is the probability the object belongs to  $\text{class}_i$  given an object is presence.

$P_r(\text{class}_i)$  is the probability the object belongs to  $\text{class}_i$

Per fare una previsione finale, YOLO mantiene solo quelli con punteggi di confidenza elevati (maggiori di 0,25) per ottenere le previsioni finali (l'immagine giusta sulla destra).



## Vantaggi di YOLO

- Veloce. Buono per l'elaborazione in tempo reale.
- Le previsioni (posizioni e classi di oggetti) sono fatte da un'unica rete. Può essere addestrato end-to-end per migliorare la precisione.
- YOLO è più generalizzato. Supera altri metodi quando si generalizza da immagini naturali ad altri domini come le opere d'arte.

## Limitazioni di YOLO

YOLO impone forti vincoli spaziali alla previsione delle “bounding boxes” poiché ogni cella della griglia prevede solo due box e può avere solo una classe. Questo limite spaziale limita il numero di oggetti vicini che il nostro modello può prevedere.

Il modello lotta con piccoli oggetti che compaiono in gruppo, come stormi di uccelli.

Dal momento che il modello impara a prevedere le caselle di delimitazione da dati, fa fatica a generalizzare agli oggetti in nuovi o insoliti proporzioni o configurazioni.

### 2.4.3 YOLOv2

Con YOLOv2 sono stati introdotti vari miglioramenti al metodo di rilevamento YOLO. Il modello migliorato, YOLOv2, è all'avanguardia per quanto riguarda il rilevamento degli obbiettivi standard come PASCAL VOC e COCO. Utilizzando un nuovo metodo di allenamento multi-scala lo stesso YOLOv2 può funzionare a varie dimensioni, offrendo un facile compromesso tra velocità e precisione. A 67 FPS, YOLOv2 ottiene 76,8 mAP su VOC 2007. A 40 FPS, YOLOv2 ottiene 78,6 mAP, superando i metodi all'avanguardia come Faster R-CNN con ResNet e SSD. Infine, è stato introdotto un nuovo metodo di allenamento. Usando questo metodo è stato possibile allenare YOLO9000 contemporaneamente sia sul set di dati di rilevamento COCO sia sul set di dati di classificazione ImageNet.

Il rilevamento di oggetti dovrebbe essere rapido, accurato, e in grado di riconoscere un'ampia varietà di oggetti. Dall'introduzione delle reti neurali, il rilevamento di oggetti è diventato sempre più veloce e preciso. Tuttavia, la maggior parte dei metodi di rilevamento sono ancora vincolati ad una piccola quantità di oggetti.

I set di dati di rilevamento degli oggetti correnti sono limitati rispetto a set di dati per altre attività come la classificazione e la codifica. I set di dati di rilevamento più comuni ne contengono centinaia di migliaia di immagini con dozzine o centinaia di tag. I set di dati di classificazione hanno milioni di immagini con decine o centinaia di migliaia di categorie. Con YOLO si vorrebbe che il rilevamento si ridimensionasse al livello della classificazione degli oggetti. Tuttavia, l'etichettatura delle immagini per il rilevamento è molto più costosa dell'etichettatura per la classificazione (i tag sono spesso forniti dall'utente gratuitamente). Pertanto, è improbabile che nel futuro prossimo vengano visualizzati set di dati di rilevamento sulla stessa scala della classificazione.

Quindi YOLO per ovviare a questo, ha proposto un nuovo metodo per sfruttare la grande quantità dei dati di classificazione che possiedono e che saranno usati per espandere la portata degli attuali sistemi di rilevamento. Questo metodo utilizza una visione gerarchica della classificazione degli oggetti che consente di combinare insieme set di dati distinti, studiando anche un algoritmo di addestramento congiunto che consente di addestrare i rilevatori di oggetti sia sul rilevamento che sulla classificazione dati. Il metodo sfrutta le immagini di rilevamento etichettate e impara a localizzare con precisione gli oggetti mentre usa la classificazione delle immagini per aumentarne il vocabolario e la robustezza della rete. Usando questo metodo è stato addestrato YOLO9000 un rilevatore di oggetti in tempo reale in grado di rilevare oltre 9000 diverse categorie di oggetti.

Innanzitutto è stata migliorata la base di YOLO per produrre YOLOv2. Quindi è stato utilizzato il nuovo metodo di combinazione del set di dati e l’algoritmo di addestramento congiunto per formare un modello su più di 9000 classi da ImageNet e dati di rilevamento da COCO.

Seppur raggiunto un buon livello prestazionale, si è potuto notare come YOLO soffrisse di una varietà di carenze relative al sistema di rilevamento. Gli errori di analisi di YOLO rispetto a Fast R-CNN mostra che YOLO fa un significativo numero di errori di localizzazione. Quindi sono stati effettuati sforzi principalmente sul miglioramento della precisione e localizzazione mantenendo l'accuratezza della classificazione. Con YOLOv2 si è puntato ad un più preciso rilevatore mantenendo comunque inalterata la velocità. Invece di ampliare la rete, è stata semplificata e quindi è stata realizzata una rappresentazione più facile da poter insegnare al computer. Per migliorare YOLO sono state messe in comune una varietà di idee dal lavoro passato con nuovi concetti.

Un riepilogo dei risultati è riassunto in tabella:

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	78.6

Le novità apportate in questa nuova versione si possono elencare in:

Batch Normalization, High Resolution Classifier, Convolutional with anchor boxes, dimension cluster, direct location prediction, fine-grained features, multi scale training.

### Batch normalization

Aggiungendo la “batch normalization” (divisione in riquadri) nei livelli di convoluzione è stata eliminata la necessità di interruzioni e aumentiamo il 2% di mAP.

### High Resolution Classifier

L'allenamento YOLO è composto da 2 fasi. Innanzitutto, è stata allenata una rete di classificatori come VGG16. Quindi sono stati sostituiti i livelli completamente collegati con uno strato di convoluzione e riqualificato end-to-end per il rilevamento degli oggetti. YOLO addestra il classificatore con immagini  $224 \times 224$  seguite da immagini  $448 \times 448$  per il rilevamento di oggetti. YOLOv2 inizia con immagini  $224 \times 224$  per l'addestramento del classificatore, ma risintonizza nuovamente il classificatore con immagini  $448 \times 448$ . Ciò semplifica l'addestramento del rivelatore e aumenta il mAP del 4%.

## Convolutional with anchor boxes

Inizialmente, YOLO fa ipotesi arbitrarie sulle bounding boxes. Queste ipotesi possono funzionare bene per alcuni oggetti, ma male per altri. All'inizio dell'allenamento, le previsioni combattono tra loro su quali forme specializzarsi.

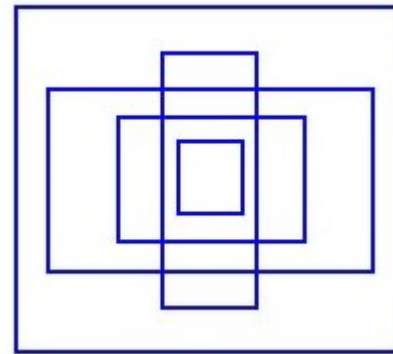
Nella vita reale, le caselle di confine non sono arbitrarie.

Le auto hanno forme molto simili e i pedoni hanno un rapporto di aspetto approssimativo di 0,41. Dal momento che YOLO ha bisogno solo di una supposizione,

l'allenamento iniziale sarà più stabile se partirà con ipotesi diverse che sono comuni per gli oggetti della vita reale.

Ad esempio, si possono creare 5 scatole di ancoraggio con diverse forme.

Invece di prevedere 5 bounding boxes arbitrarie, si prevede l'offset di ciascuna delle anchor boxes. Se si vincolano i valori di offset, si potrà mantenere la diversità delle previsioni e ogni previsione si concentra su una forma specifica. Quindi l'allenamento iniziale sarà più stabile.

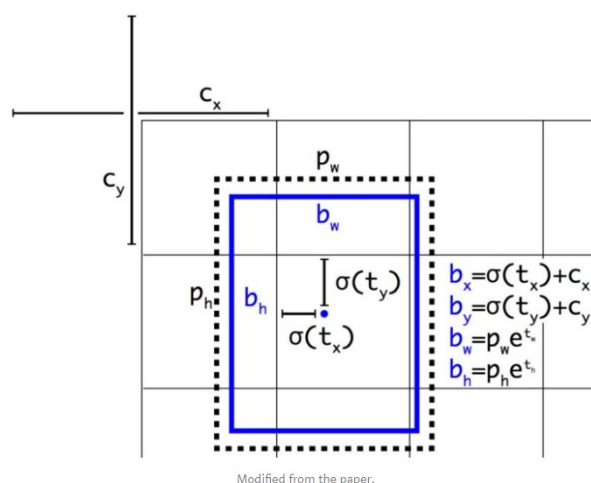


## Dimension cluster

Facciamo previsioni sugli offset delle anchor box. Tuttavia, se non vincolate, le nostre ipotesi saranno nuovamente casuali. YOLO prevede 5 parametri ( $t_x$ ,  $t_y$ ,  $t_w$ ,  $t_h$  e  $t_o$ ) e applica la funzione sigma per limitare il suo possibile intervallo di offset.

Ecco l'esempio. La casella blu è la boundary box prevista e il rettangolo punteggiato è l'anchor box.

Con l'uso del clustering k-mean (cluster di dimensioni) e il miglioramento menzionato in questa sezione, il mAP aumenta del 5%.



## Classificazione

I set di dati per il rilevamento di oggetti hanno molte meno classi rispetto a quelli per la classificazione. Per espandere le classi che YOLO è in grado di rilevare, YOLO propone un metodo per mescolare le immagini dai set di dati di rilevamento e classificazione durante l'allenamento.

Questo approccio incontra alcune sfide:

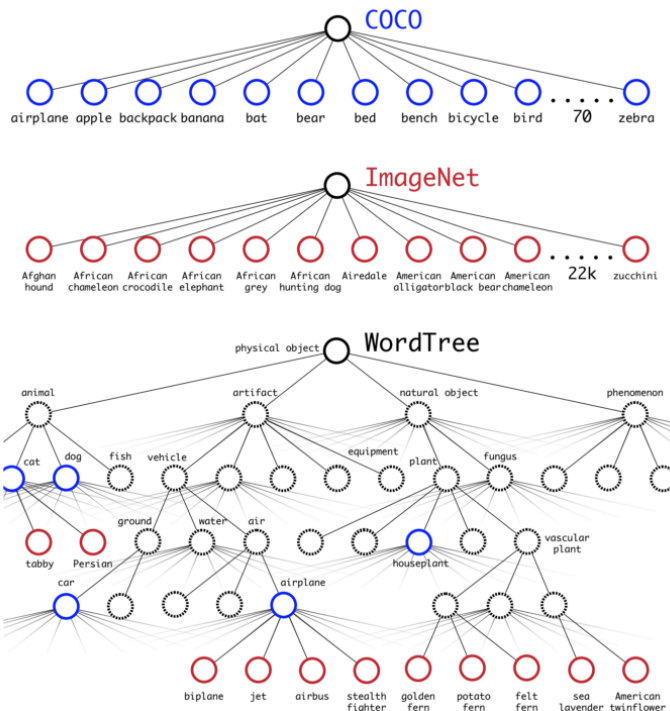
Come si può unire etichette di classe da diversi set di dati? In particolare, i set di dati di rilevamento degli oggetti e i diversi set di dati di classificazione utilizzano etichette diverse.

Eventuali etichette unite potrebbero non escludersi a vicenda, ad esempio "Norfolk terrier" (una razza di cane) in ImageNet e "cane" in COCO. Poiché non si escludono a vicenda, non è possibile utilizzare softmax per calcolare la probabilità.

## Classificazione gerarchica

Per ovviare a questo problema, YOLO combina etichette in diversi set di dati per formare una struttura ad albero “WordTree”.

Semplifichiamo la discussione utilizzando ImageNet con 1000 classi. Invece di prevedere 1000 etichette in una struttura piatta, è stato creato il WordTree corrispondente a 1000 nodi di partenza per le etichette originali e 369 nodi per le loro classi principali. Inizialmente, YOLO prevede il punteggio di classe per il biplano. Ma con WordTree, ora prevede il punteggio per il biplano dato che è un aeroplano.



In conclusione quindi si è ottenuto YOLOv2 e YOLO9000, due sistemi di rilevamento in tempo reale. YOLOv2 è all'avanguardia e più veloce rispetto ad altri sistemi di rilevamento attraverso una varietà di set di dati di rilevamento. Inoltre, può essere eseguito su diverse dimensioni di immagini per fornire un compromesso tra velocità e precisione.

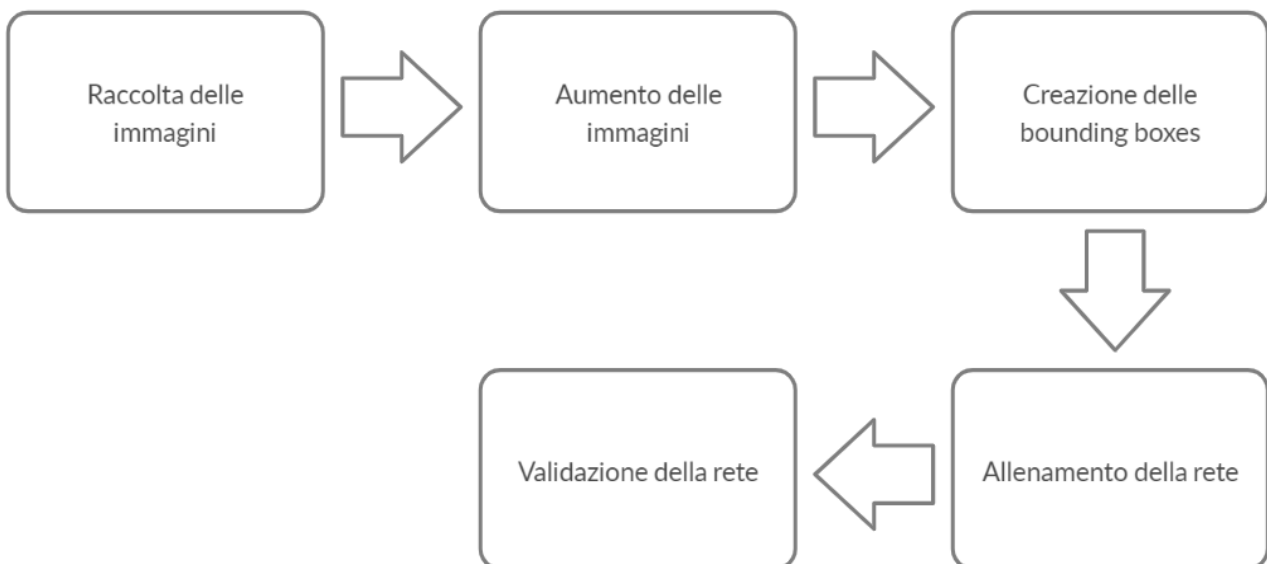
YOLO9000 è una struttura in tempo reale per il rilevamento di più di 9000 categorie di oggetti ottimizzando congiuntamente il rilevamento e classificazione. È stato introdotto inoltre WordTree per combinare i dati da varie fonti e tecnica di ottimizzazione congiunta per la formazione contemporaneamente su ImageNet e COCO. YOLO9000 è un forte passo in avanti per colmare il divario dimensionale del set di dati tra rilevamento e classificazione.



### 3 Metodologia

#### 3.1 Rete e software di allenamento

L'obiettivo quindi è quello di creare una rete stabile e affidabile per il rilevamento delle uscite di emergenza, una rete che poi potrà essere in futuro utilizzata per ulteriori studi. Per raggiungere i nostri obiettivi e per eseguire i nostri allenamenti abbiamo fatto uso della rete neurale YOLOv2 in versione tiny, una versione più leggera utilizzata per le applicazioni desktop e adeguata per i nostri scopi. Come sistema di allenamento è stato utilizzato il "Darknet" un software sviluppato dallo stesso sviluppatore della rete in grado di allenare la rete stessa per il riconoscimento di oggetti. Per ottenere un buon allenamento è stato utilizzato un iter procedurale seguendo i seguenti step.

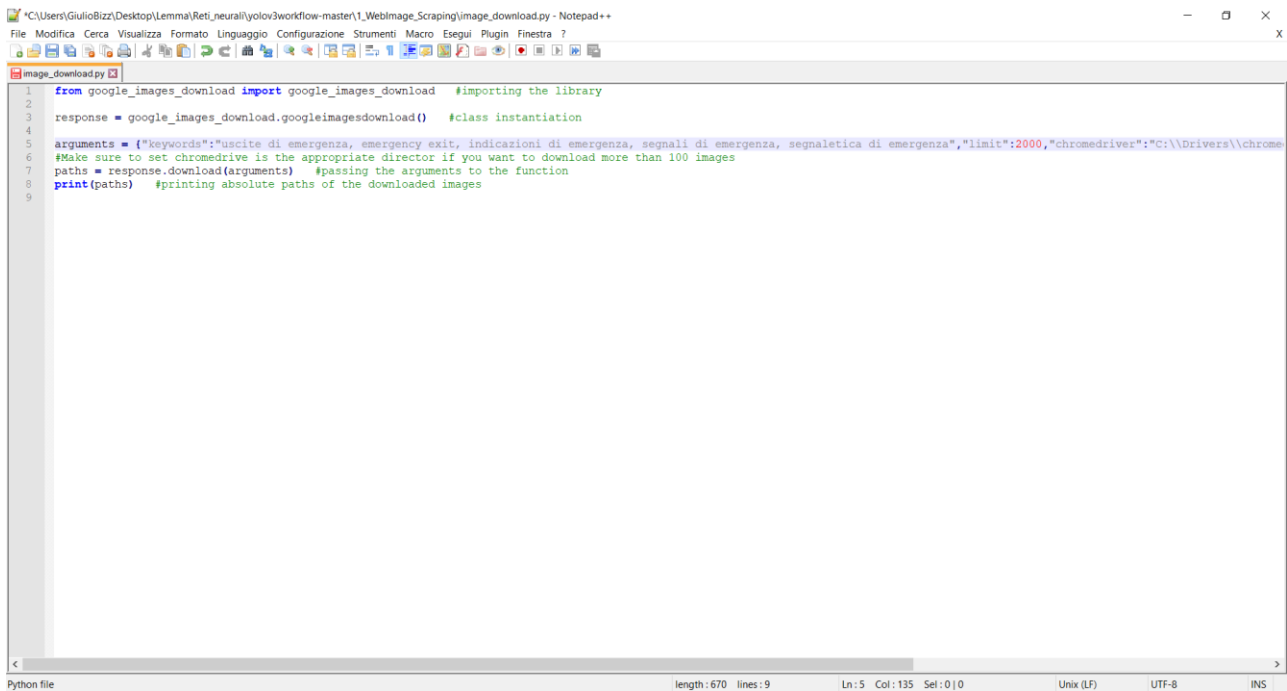


## 3.2 Formazione del dataset

### 3.2.1 Raccolta delle immagini

La prima operazione, nonché quella fondamentale per ottenere un buon allenamento, è stata partire da una buona base di immagini. Per questo il dataset è stato creato utilizzando sia una serie di immagini scattate attraverso l'uso della fotocamera di uno smartphone ma soprattutto da immagini scaricate direttamente dal web. Per fare ciò si è diviso il lavoro in due step, come primo step si è cercato di scattare il più possibile foto di segnali di emergenza reali cercando di trovarne di più svariati e nelle condizioni più disparate. Partendo dalla facoltà di ingegneria presente in Ancona e muniti di smartphone si è percorsa l'intera facoltà, tra corridoi, stanze e locali per poter acquisire quante più possibili immagini. Inoltre per arricchire la quantità di immagini scattate si è cercato, per quanto possibile, di fotografare il maggior numero di segnali nei locali, negozi e sul posto di lavoro. Terminata questa attività e avendo già una discreta ma non sufficiente base da cui partire, come secondo step si è cercato sul web e scaricato quante più possibili immagini. Operazione questa resa possibile utilizzando anaconda e `image_download`. Per poter utilizzare questo strumento è stato necessario installare anaconda sul pc seguendo una procedura online ed abilitare "Webimage\_Scraping". Per una corretta procedura si illustrano ora i vari passaggi.

Come prima operazione sarà necessario specificare con quale parola chiave il nostro software dovrà cercare sul motore di ricerca e fino a che numero massimo di file potrà scaricare. Per il nostro allenamento si sono utilizzate queste parole chiavi (uscite di emergenza, emergency exit, indicazioni di emergenza, segnali di emergenza, segnaletica di emergenza) e imposto come limite 2000 immagini (il limite di 2000 immagini è un limite teorico per dire al computer di scaricare più immagini possibili. Infatti non succederà mai o quantomeno sarà raro che il software riesca a trovare un numero così elevato di immagini in rete). Per fare questo quindi bisognerà aprire `image_download` (tasto destro, apri con notepad) cambiare le variabili (parola chiave e n. max. di file) e salvare il tutto.



```
1 from google_images_download import google_images_download #importing the library
2
3 response = google_images_download.googleimagesdownload() #class instantiation
4
5 arguments = {"keywords": "uscite di emergenza, emergency exit, indicazioni di emergenza, segnali di emergenza, segnaletica di emergenza", "limit": 2000, "chromedriver": "C:\\Drivers\\chrome
6 #Make sure to set chromedriver is the appropriate director if you want to download more than 100 images
7 paths = response.download(arguments) #passing the arguments to the function
8 print(paths) #printing absolute paths of the downloaded images
9
```

Una volta completato sarà necessario aprire anaconda e sul comando dei prompt scrivere il percorso file per arrivare ad WebImage\_Scaping: cdC:\Users\GiulioBizz\Desktop\Lemma\Reti\_neurali\yolov3workflow-master\1\_WebImage\_Scraping

Poi sulla stessa riga del comando scrivere: python image\_download.py dopodiché partirà il download.

Una volta completato il download il programma creerà automaticamente una cartella per ogni parola chiave inserita per la ricerca e salverà all'interno di ognuna di esse tutte le foto scaricate. Quest il percorso per arrivare alle cartelle:

yolov3workflow-master\1\_WebImage\_Scraping\downloads

Una volta scaricate le immagini si è reso necessario a quel punto controllarle per eliminare eventuali immagini non conformi per l'allenamento come ad esempio immagini non coerenti con la parola chiave cercata, immagini troppo piccole o immagini che secondo noi potrebbero non essere completamente idonee al fine dell'allenamento. Quindi seppur partendo da una base abbastanza ricca si è ottenuto un numero più ristretto di immagini utili. A questo punto è stato possibile unire tutte le immagini in un'unica cartella e continuare l'iter per l'allenamento.

### 3.2.2 Aumento delle immagini

Una volta quindi aver raggruppato le nostre foto scaricate e quelle fotografate in un'unica cartella, il passo successivo è stato quello dell'aumento per arricchire ulteriormente il nostro dataset. Per aumento si intende la duplicazione delle nostre foto di base cambiando alcuni parametri come ad esempio esposizione, orientamento, etc. Durante questa fase si è cercato il più possibile di mantenere il rapporto 1 ad 1 cioè per ogni foto originale crearne una aumentata in modo così da avere a operazione terminata il doppio delle immagini di partenza. Si è scelto di aumentarle solamente di una volta perché da alcuni esperimenti svolti al di fuori di questa tesi si è notato come 1 ad 1 fosse il rapporto ideale per raggiungere risultati migliori, infatti aumentando questo rapporto la rete allenandosi avrebbe visto sempre la stessa foto e anche se in condizioni diverse peggiorava il risultato finale. Per aumentare le immagini si è fatto ricorso ad un servizio online gratuito disponibile al seguente link: <https://app.supervise.ly/teams/my> sul quale sarà necessario solamente effettuare una registrazione.

Per aumentare le immagini si è utilizzato il seguente iter.

Per prima cosa bisognerà aprire la pagina: <https://app.supervise.ly/teams/my>. Una volta aperto il sito e dopo essersi registrati, bisognerà cliccare su import, trascinare la cartella contenente le immagini da aumentare nell'apposito settore e dopo aver inserito il nome del progetto cliccare su start import. A questo punto cliccando nella barra di sinistra sulla voce "DTL" ci apparirà una schermata in cui si dovrà scrivere il codice per fare l'aumento. Il codice standard da poter utilizzare lo si può trovare nella pagina: [https://github.com/reigngt09/yolov3workflow/blob/master/5.Data\\_Augmentation/DTL.txt](https://github.com/reigngt09/yolov3workflow/blob/master/5.Data_Augmentation/DTL.txt) (con questo codice avremo però un aumento eccessivo delle immagini ma per ovviare al problema lascio in fondo il codice utilizzato da me per ottenere un aumento di 1 ad 1). Nel codice cambiare la riga: "Console Dataset/\*" con "nome della cartella caricata/\*" e la riga: "Console Dataset\_Aug" con "nome della cartella caricata\_Aug". A questo punto non resta che cliccare su start e partirà il procedimento. Una volta terminata l'operazione per scaricare le immagini aumentate, cliccare su DTL, cercare la cartella denominata con "nome della cartella caricata\_Aug", cliccare sui 3 pallini affianco alla cartella e poi su download as .json+images. A questo punto basterà tornare su task, far partire il download ed aprire il file scaricato con zip. All'interno ci sarà la cartella immagini con tutte le foto.

Il codice utilizzato e creato appositamente ottenere soltanto un'immagine aumentata è stato il seguente:

```

[
  {
    "dst": "$data",
    "src": [
      "Console Dataset/*"
    ],
    "action": "data",
    "settings": {
      "classes_mapping": "default"
    }
  },
  {
    "dst": "$flip_vert",
    "src": [
      "$data"
    ],
    "action": "flip",
    "settings": {
      "axis": "vertical"
    }
  },
  {
    "dst": "Console Dataset_Aug",
    "src": [
      "$data",
      "$flip_vert"
    ],
    "action": "supervisely",
    "settings": {}
  }
]

```

Terminata questa operazione si è ottenuto un incremento in numero di immagini per passare poi allo step successivo.

### 3.2.3 Creazione delle bounding boxes

A questo punto ai fini dell'allenamento è stato necessario indicare per ogni singola foto un quadro di delimitazione il cosiddetto bounding box. Questo bounding Box essenzialmente serve al sistema di allenamento per indicare alla rete in che posizione si trova l'oggetto che la rete deve riconoscere e di che tipo di oggetto si tratta. Attraverso questo procedimento quindi è stato possibile indicare all'interno di un'immagine sia la tipologia di oggetto da rilevare chiamata classe e indicata con un numero progressivo da 0 in poi, sia la dimensione dell'oggetto e la sua posizione indicando le coordinate assiali del suo baricentro. Per il nostro caso di utilizzo è stato necessario predisporre 3 diverse classi di oggetto:

Classe 0 > emergencysigndoor

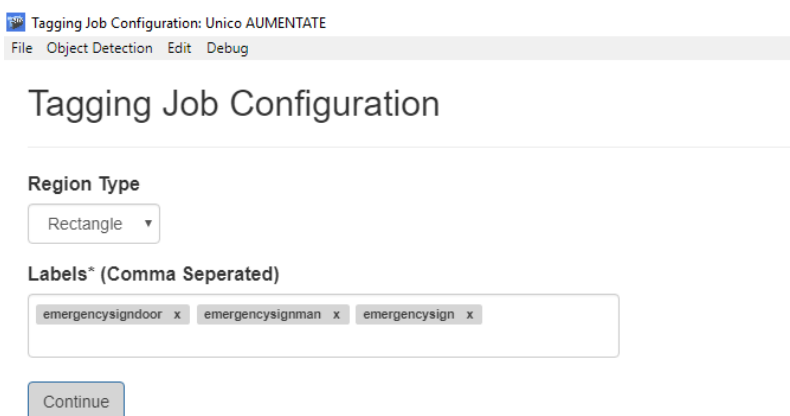
Classe 1 > emergencysignman

Classe 2 > emergencysign

Ognuna di esse riferite rispettivamente a 3 diverse tipologie di segnali di emergenza.

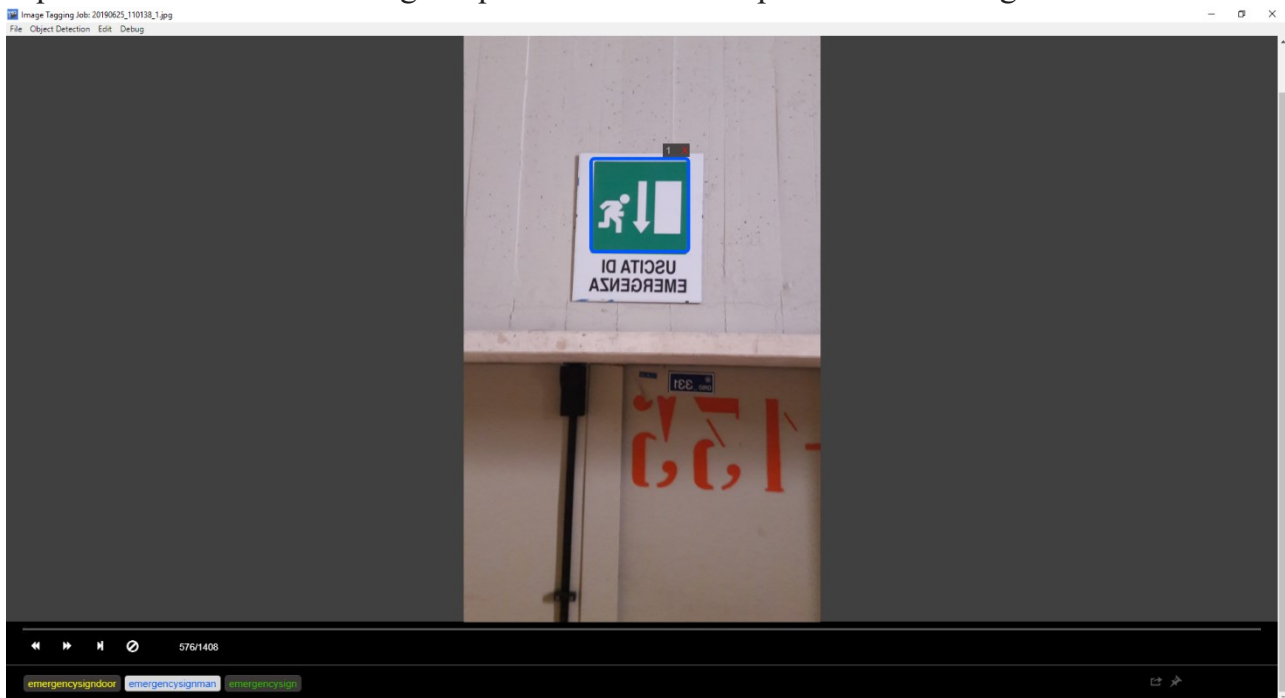
Per la creazione delle bounding boxes si è fatto ricorso al programma VoTT scaricabile gratuitamente da internet. Questa operazione ha richiesto molta attenzione e pazienza perché di estrema importanza al fine di un corretto allenamento e a tal proposito si illustrano i passaggi fondamentali.

Per prima cosa sarà necessario aprire il programma VoTT e nella schermata iniziale selezionare il riquadro a sinistra che permetterà di importare le immagini da una cartella creata appositamente con tutte le immagini da elaborare. A questo punto nella schermata successiva verrà richiesto di indicare la forma della bounding box (rettangolare) e di specificare il nome per ogni singola classe.

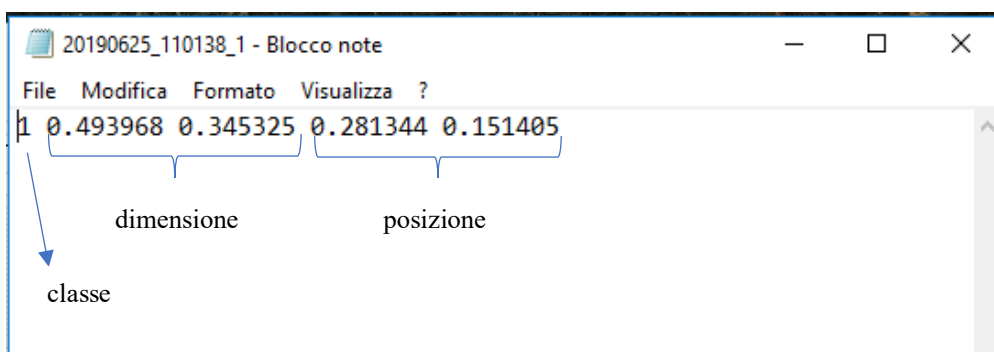


Ora cliccando su continua si potrà procedere con la creazione delle bounding boxes. A questo punto comparirà una alla volta tutte le foto facenti parte della cartella inserita all'inizio. Qui per ogni immagine, con il puntatore del mouse sarà necessario disegnare il riquadro contenente l'oggetto di interesse al fine dell'allenamento. Inoltre se per caso si avessero

diverse classi sarà necessario indicare per ogni rettangolo anche la classe selezionando dai riquadri sottostanti all'immagine quello contenente la parola chiave adeguata.



Terminata questa procedura per ogni singola immagine, non resterà che esportare il tutto cliccando su Object Detection, Export Tags, selezionare su Export Format: Yolo, indicare la cartella di esportazione ed infine cliccare su export. Una volta terminata l'esportazione, si avrà una cartella al cui interno si potranno trovare sia le immagini sia il loro corrispondente file di testo in cui saranno presenti le indicazioni di classe, dimensione e posizione dell'oggetto individuato.



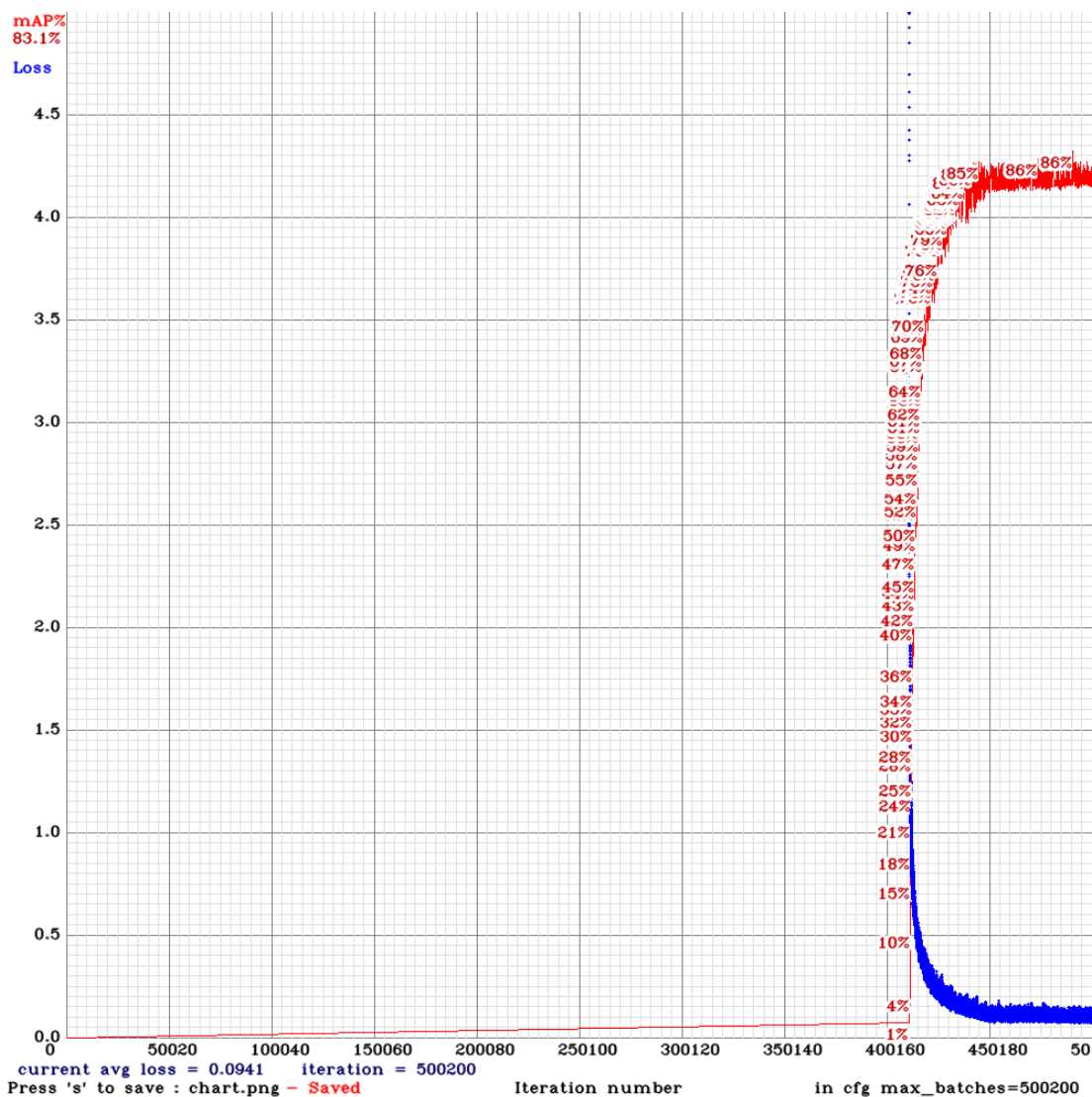
Una volta terminata questa operazione e quindi dopo aver completato uno degli step fondamentali per l'allenamento, è stato possibile creare i vari dataset utili per l'allenamento. Il dataset non è altro che una cartella all'interno della quale sono state inserite tutte le immagini con le relative bounding boxes e tutte le informazioni necessarie.

A questo punto per poter tenere traccia dei dataset e degli allenamenti si è deciso di creare un file word o excel nel quale si è andato ad inserire e a descrivere per filo e per segno tutti i vari dataset e allenamenti che effettuati nel corso della sperimentazione in modo da tenere organizzato il lavoro.

### 3.3 Allenamento di una rete

Una volta completate tutte le operazioni preliminari si è passato alla fase di allenamento vero e proprio. In questa fase è stato possibile, utilizzando il Darknet, allenare la nostra rete per i nostri scopi. Infatti partendo da diversi dataset si sono eseguiti più di un allenamento potendo testare così la precisione della rete e valutare il diverso comportamento in funzione del dataset di partenza. Si sono utilizzate sempre reti pre-allenate impostando determinati pesi iniziali utilizzando quindi come base di partenza questi due file : `cfg/yolov2-tiny-ale.cfg` e `yolov2-tiny-ale.weights`, l'uno per la configurazione della rete e l'altro per i pesi iniziali. Una volta quindi avviato l'allenamento il nostro software ha iniziato la fase di apprendimento dove visualizzando le immagini, le loro relative bounding boxes e modificando i pesi dell'allenamento ha restituito in tempo reale un diagramma chiamato chart in cui si ha lungo l'asse delle ascisse il numero delle iterazioni, sull'asse delle ordinate la funzione di perdita e in rosso il mAP ossia la precisione della nostra rete. Gli allenamenti sono stati ritenuti conclusi nel momento in cui sia il segno del chart (rosso) che del loss (blu) presentavano un andamento asintotico sul grafico.

(esempio di un grafico durante l'allenamento)





Concluso l'allenamento, la rete ha restituito 3 differenti file. Il primo, il file backup, è una cartella contenente diversi file .weights che altro non sono dei file contenenti informazioni sui pesi utilizzati dalla rete durante l'allenamento in funzione delle diverse iterazioni e che saranno importanti poi alla fine per valutare con quali serie di pesi l'allenamento ha ottenuto i migliori risultati. Il secondo file restituito dalla rete è invece il chart che è il grafico completo restituito dalla rete una volta ultimato l'allenamento sul quale si nota (dal segno rosso) la percentuale massima di mAP che ha raggiunto la nostra rete. Infine come terzo file la rete ci restituisce il file log, un file contenente tutte le operazioni effettuate dalla rete durante l'allenamento che è stato utilizzato per determinare con precisione il mAP della rete, durante quale iterazione si è raggiunta quella precisione e con quali pesi è stato possibile raggiungere tale valore. Ma ora vediamo passo a passo le varie fasi per configurare al meglio l'allenamento.

Come operazione preliminare sarà necessario inserire all'interno della cartella darknet un'ulteriore cartella (con il nome dell'allenamento) in cui verrà creata la cartella obj al cui interno bisognerà inserire le immagini del dataset con le rispettive bounding boxes e due ulteriori file creati con notepad e chiamati "test" e "train". Il file "train" dovrà contenere i percorsi unicamente delle immagini per permettere alla rete di trovare all'interno della cartella i file per l'allenamento, mentre il file "test" dovrà contenere i percorsi unicamente delle immagini per permettere alla rete di trovare all'interno della cartella i file per la validazione (solitamente circa il 20% del totale delle immagini). Inoltre sarà opportuno creare all'interno della cartella Darknet una cartella chiamata backup per poter permettere alla rete di salvare i file .weights.

```

Train - Blocco note
File Modifica Formato Visualizza ?
data\obj\20190625_104602_1.jpg
data\obj\20190625_104629.jpg
data\obj\20190625_104629_1.jpg
data\obj\20190625_104637.jpg
data\obj\20190625_104637_1.jpg
data\obj\20190625_104643.jpg
data\obj\20190625_104643_1.jpg
data\obj\20190625_104732(0).jpg
data\obj\20190625_104732(0)_1.jpg
data\obj\20190625_105450.jpg
data\obj\20190625_105450_1.jpg
data\obj\20190625_105454.jpg
data\obj\20190625_105454_1.jpg
data\obj\20190625_105637.jpg
data\obj\20190625_105637_1.jpg
data\obj\20190625_105655.jpg
data\obj\20190625_105655_1.jpg
data\obj\20190625_105700.jpg
data\obj\20190625_105700_1.jpg
data\obj\20190625_105735.jpg
data\obj\20190625_105735_1.jpg
data\obj\20190625_105741.jpg
data\obj\20190625_105741_1.jpg
data\obj\20190625_105753.jpg
data\obj\20190625_105753_1.jpg
data\obj\20190625_105802.jpg
data\obj\20190625_105802_1.jpg

```

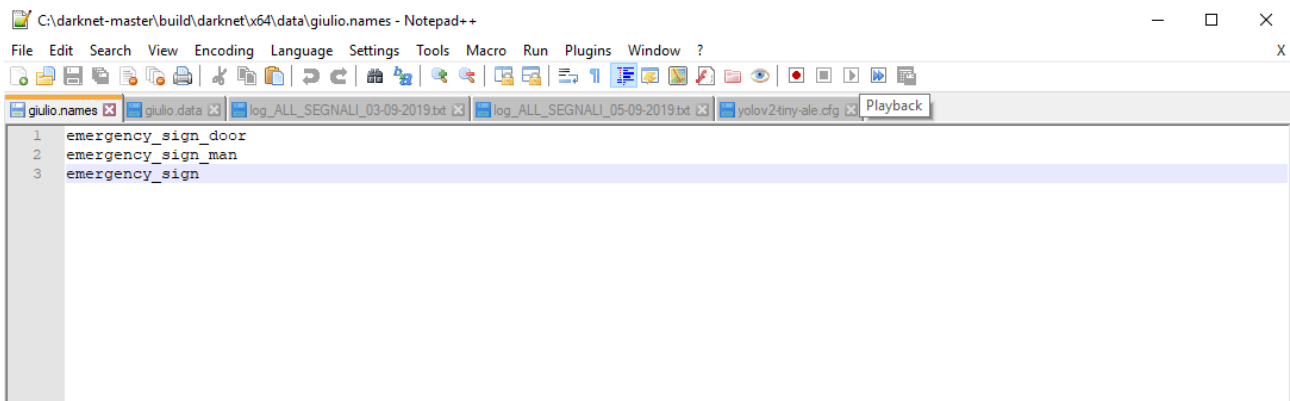
```

Test - Blocco note
File Modifica Formato Visualizza ?
data\obj\321_emergency-exit-1.jpg
data\obj\321_emergency-exit-1_kIWaxv5pX3.jpg
data\obj\322_tempus-exc.jpg
data\obj\322_tempus-exc_wfoswH8nMk.jpg
data\obj\326_uscita-di-emergenza-con-freccia-116026.jpg
data\obj\326_uscita-di-emergenza-con-freccia-116026_Dn9eyuT9LQ.jpg
data\obj\326_uscita-di-emergenza-con-freccia-116026_EOGTqINc1K.jpg
data\obj\326_uscita-di-emergenza-con-freccia-116026_M5WPLq95kL.jpg
data\obj\326_uscita-di-emergenza-con-freccia-116026_xnmJsM1nR5.jpg
data\obj\327_t_u3701500.jpg
data\obj\327_t_u3701500_dffG5QHc1g.jpg
data\obj\329_closeup-neon-exit-sign.jpg
data\obj\329_closeup-neon-exit-sign_TkDY09xGBr.jpg
data\obj\332_img.NjM0NzM2MTc.jpeg
data\obj\332_img.NjM0NzM2MTc_nK6yiwjR91.jpeg
data\obj\333_Emergency-Exit-Sign-HB-27M-3-8H-AT-MiCh-Side.jpg
data\obj\333_Emergency-Exit-Sign-HB-27M-3-8H-AT-MiCh-Side_JAPy3yxISX.jpg
data\obj\334_eds.jpg
data\obj\334_eds_pUaL0SWBnp.jpg
data\obj\335_6654874.jpg
data\obj\335_6654874_0Pb0hONsVw.jpg
data\obj\336_F7959744-01.jpg
data\obj\336_F7959744-01_DMD3dE8rh.n.jpg

```

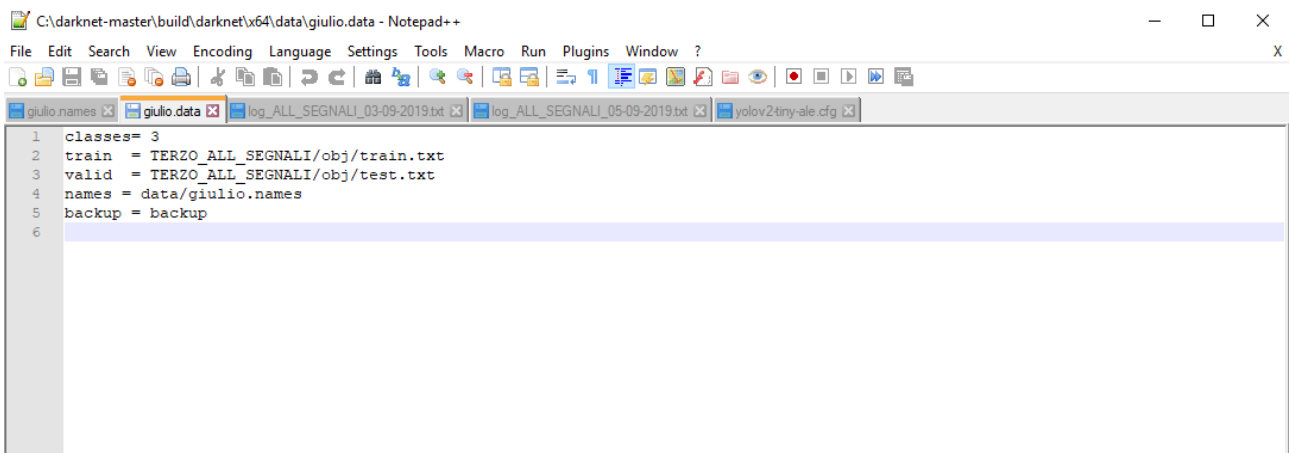
Come primo step sarà fondamentale configurare la rete, quindi una volta entrati all'interno della cartella "darknet", e una volta essersi spostati nella cartella "data" sarà necessario copiare i file .names e .data per poi poterli modificare. Una volta creati questa due file sarà

necessario aprire con notepad++ il file .names in cui all'interno bisognerà indicare su righe differenti i tag delle varie classi.



```
C:\darknet-master\build\darknet\x64\data\giulio.names - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
giulio.names | giulio.data | log_ALL_SEGNALI_03-09-2019.txt | log_ALL_SEGNALI_05-09-2019.txt | yolov2tiny-ale.cfg | Playback
1 emergency_sign_door
2 emergency_sign_man
3 emergency_sign
```

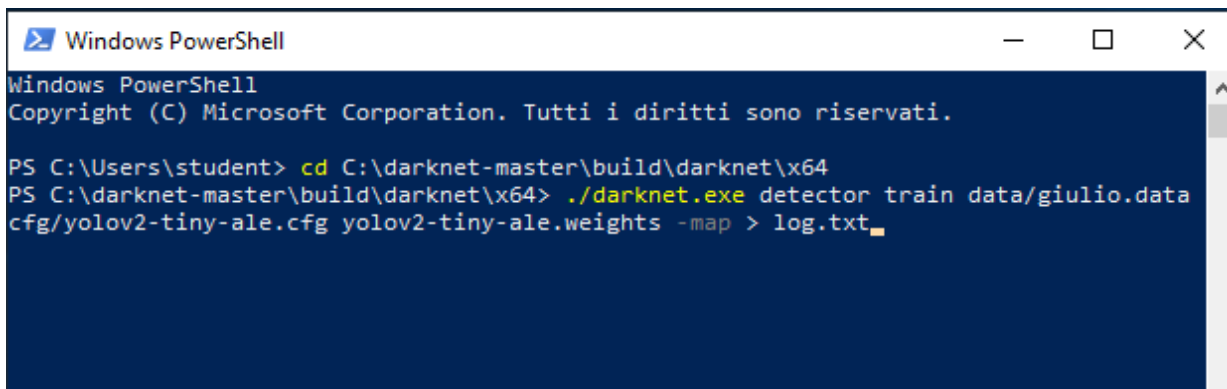
Una volta effettuata questa operazione e salvato il tutto bisognerà spostarsi sul file .data, aprirlo con notepad++ e modificare alcuni parametri. Per la voce “classes” sarà necessario indicare il numero di classi con la quale la nostra rete sarà allenata, per “train” dovrà essere indicato il percorso per arrivare al file “train.txt” così come per valid il percorso per arrivare al file “test.txt”. Per l’indicazione names sarà necessario specificare il percorso per il file .names e per backup il percorso per la cartella backup.



```
C:\darknet-master\build\darknet\x64\data\giulio.data - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
giulio.names | giulio.data | log_ALL_SEGNALI_03-09-2019.txt | log_ALL_SEGNALI_05-09-2019.txt | yolov2tiny-ale.cfg
1 classes= 3
2 train = TERZO_ALL_SEGNALI/obj/train.txt
3 valid = TERZO_ALL_SEGNALI/obj/test.txt
4 names = data/giulio.names
5 backup = backup
6
```

Una volta completate queste operazioni, il secondo step sarà quello di avvio dell’allenamento. Per fare questo bisognerà aprire windows powershell e nella riga dei comandi scrivere il percorso file per arrivare alla cartella darknet: **cd C:\darknet-master\build\darknet\x64** e premere invio.

Per far partire l’allenamento invece nella riga seguente bisognerà incollare questa stringa: **/darknet.exe detector train data/giulio.data cfg/yolov2-tiny-ale.cfg yolov2-tiny-ale.weights -map > log.txt** avendo precedentemente modificato i valori sottolineati in giallo.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tutti i diritti sono riservati.





PS C:\Users\student> cd C:\darknet-master\build\darknet\x64
PS C:\darknet-master\build\darknet\x64> ./darknet.exe detector train data/giulio.data
cfg/yolov2-tiny-ale.cfg yolov2-tiny-ale.weights -map > log.txt
```

Una volta premuto invio la rete inizierà il proprio allenamento ed una volta concluso basterà premere sulla riga dei comandi la combinazione di tasti Ctrl+c per interrompere l'allenamento. Una volta completato l'allenamento, come terzo step, sarà necessario salvare i 3 file di output rilasciati dalla rete. Per fare questo occorrerà rinominare i 3 file in modo da non essere sovrascritti dalla rete una volta fatto partire un nuovo allenamento.

Terminato l'allenamento quindi è stato possibile valutarne a primo impatto le prestazioni ricorrendo all'analisi di valori di map e di avg loss, e avere indicazioni sul risultato della rete. Risultato che si è considerato soddisfacente con un valore di mAP superiore all'80% mentre valori dal 60 al 70 % sono stati considerati non adeguati. Ultimata questa fase si è potuto procedere con la validazione.

### 3.4 Validazione di una rete

Dopo aver effettuato l'allenamento, la fase conclusiva è stata quella della validazione. Per effettuare la validazione è stato necessario studiare sia il chart che il file di log per poter determinare la precisione (mAP) massima e scoprire con quali pesi fosse stata raggiunta. Una volta determinata l'iterazione con il valore di mAP più alto è stato necessario eseguire la validazione utilizzando i pesi riferiti a quella iterazione e valutarne l'andamento. In questa fase utilizzando le immagini predisposte per la fase di test si è dovuto, per ogni immagine, farla elaborare alla rete e verificare per ognuna di esse diverse caratteristiche. Prima di tutto si è dovuto verificare che la rete fosse stata in grado di riconoscere tutti gli elementi da rilevare all'interno della foto, se alcuni di essi non fossero stati rilevati, se fossero stati rilevati oggetti in una posizione in cui oggetti da rilevare non c'erano o se fossero stati rilevati in modo non corretto. Per valutarne le prestazioni a questo punto si è dovuto creare una tabella excel nella quale per ogni foto è stato indicato se il rilevamento aveva prodotto un rilevamento positivo o negativo. Nello specifico per ogni immagine è stato indicato con l'acronimo TP "true positive" se l'oggetto era stato rilevato correttamente all'interno dell'immagine, FP "false positive" se era stato rilevato un oggetto che però non era presente nell'immagine, FN "false negative" se l'oggetto presente nell'immagine non era stato rilevato. Inoltre per i TP e i FP è stata indicata anche la percentuale di precisione restituita dal test mentre per i FN una precisione pari allo 0%

	A	B	C	D	E	F	G	H	I	J	K	L
1	image	bounding box	confidence	TP/FP/FN								
	TERZO_ALL_SEGNAL\obj\20190625_103424.jpg											
2												
3			83,00%	TP					VALIDATION THRESHOLD 60%		TRUE POSITIVE	200
4	TERZO_ALL_SEGNAL\obj\20190625_103424_1.jpg		99,00%	TP					TOTAL NUMBER OF FE 277		FALSE POSITIVE	44
5											FALSE NEGATIVE	24
6	TERZO_ALL_SEGNAL\obj\20190625_103528.jpg											
7			99,00%	TP							PRECISION	0,81967
8	TERZO_ALL_SEGNAL\obj\20190625_103528_1.jpg		100,00%	TP							RECALL	0,89286
											F1	0,8547

Finita questa operazione e ripetuta per ogni foto, grazie al sommario posto in alto a destra è stato possibile verificare l'andamento del test e controllare la precisione della validazione.

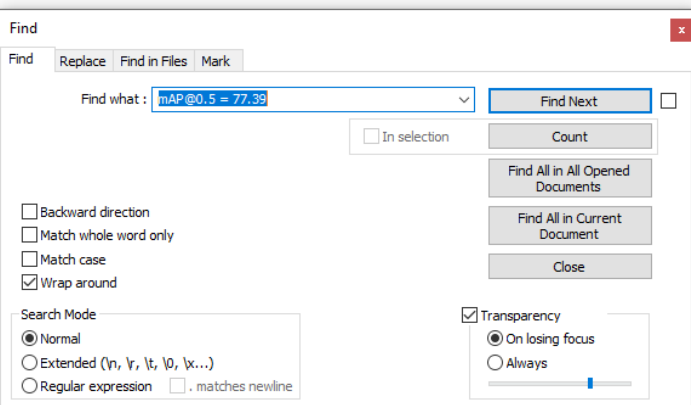
Per eseguire la validazione è stato necessario seguire una serie di accortezze ed operazioni di seguito sintetizzate.

Come operazione preliminare sarà necessario andare ad individuare quali pesi utilizzare per la validazione. Per fare questo una volta completato l'allenamento sarà necessario aprire il file chart per visualizzare a schermo il valore approssimato di mAP massimo raggiunto dalla rete. Una volta individuato questo valore si dovrà aprire il file log al cui interno, per ogni iterazione, è indicato un valore di mAP. Essendo però il valore di mAP visualizzato sul chart un dato non troppo preciso ma approssimato, dal file log bisognerà controllare questo dato per valutare con precisione il massimo valore raggiunto. Per fare questo, dal file log aperto con notepad++, bisognerà effettuare una ricerca. Premendo la combinazione di tasti Ctrl+f si aprirà la schermata di ricerca al cui interno si andrà a cercare la parola "mAP@0.5 = \*" dove al posto di \* bisognerà inserire il valore massimo di mAP letto sul chart. Se la ricerca non dovesse portare ad alcun risultato bisognerà provare con risultati più bassi o altrimenti in caso contrario con valori più alti. Bisognerà poi ripetere l'operazione fino a quando non si troverà il valore massimo di mAP.

```

406934
406935 (next mAP calculation at 440700 iterations)
406936 Last accuracy mAP@0.5 = 77.39 %, best = 80.23 %
406937 440678: 0.115599, 0.128175 avg loss, 0.000100 rate, 1.124000 seconds, 28203392 images
406938 Loaded: 0.000000 seconds
406939 Region Avg IOU: 0.799478, Class: 0.998654, Obj: 0.972926, No Obj: 0.007134, Avg Recall: 1.000000, count: 9
406940 Region Avg IOU: 0.827846, Class: 0.998687, Obj: 0.965992, No Obj: 0.007739, Avg Recall: 1.000000, count: 10
406941 Region Avg IOU: 0.696281, Class: 0.996413, Obj: 0.913049, No Obj: 0.006856, Avg Recall: 0.833333, count: 12
406942 Region Avg IOU: 0.800921, Class: 0.937886, Obj: 0.988380, No Obj: 0.008703, Avg Recall: 1.000000, count: 11
406943 Region Avg IOU: 0.826530, Class: 0.999348, Obj: 0.989276, No Obj: 0.006622, Avg Recall: 1.000000, count: 9
406944 Region Avg IOU: 0.728053, Class: 0.999253, Obj: 0.982988, No Obj: 0.007154, Avg Recall: 0.900000, count: 10
406945 Region Avg IOU: 0.731780, Class: 0.988591, Obj: 0.897708, No Obj: 0.010267, Avg Recall: 0.875000, count: 16
406946 Region Avg IOU: 0.808966, Class: 0.998441, Obj: 0.789576, No Obj: 0.006486, Avg Recall: 1.000000, count: 10
406947
406948 (next mAP calculation at 440700 iterations)
406949 Last accuracy mAP@0.5 = 77.39 %, best = 80.23 %
406950 440679: 0.117429, 0.127100 avg loss, 0.000100 rate, 1.124000 seconds, 28203456 images
406951 Loaded: 0.000000 seconds
406952 Region Avg IOU: 0.747063, Class: 0.989613, Obj: 0.823959, No Obj: 0.007866, Avg Recall: 0.900000, count: 10
406953 Region Avg IOU: 0.777384, Class: 0.975542, Obj: 0.911851, No Obj: 0.010095, Avg Recall: 0.937500, count: 16
406954 Region Avg IOU: 0.793217, Class: 0.997349, Obj: 0.991245, No Obj: 0.006032, Avg Recall: 1.000000, count: 10
406955 Region Avg IOU: 0.761792, Class: 0.988208, Obj: 0.913878, No Obj: 0.008042, Avg Recall: 0.866667, count: 15
406956 Region Avg IOU: 0.767354, Class: 0.998280, Obj: 0.959904, No Obj: 0.007833, Avg Recall: 1.000000, count: 16
406957 Region Avg IOU: 0.776140, Class: 0.995342, Obj: 0.919506, No Obj: 0.005813, Avg Recall: 1.000000, count: 8
406958 Region Avg IOU: 0.802867, Class: 0.971687, Obj: 0.970485, No Obj: 0.006588, Avg Recall: 1.000000, count: 8
406959 Region Avg IOU: 0.828973, Class: 0.989360, Obj: 0.994962, No Obj: 0.007314, Avg Recall: 1.000000, count: 10
406960
406961 (next mAP calculation at 440700 iterations)
406962 Last accuracy mAP@0.5 = 77.39 %, best = 80.23 %
406963 440680: 0.117429, 0.127100 avg loss, 0.000100 rate, 1.124000 seconds, 28203456 images
406964 Resizing
406965 384 x 384
406966 try to allocate
406967 CUDA allocation
406968 Loaded: 0.000000 seconds
406969 Region Avg IOU: 0.747063, Class: 0.989613, Obj: 0.823959, No Obj: 0.007866, Avg Recall: 0.900000, count: 8
406970 Region Avg IOU: 0.777384, Class: 0.975542, Obj: 0.911851, No Obj: 0.010095, Avg Recall: 1.000000, count: 9
406971 Region Avg IOU: 0.793217, Class: 0.997349, Obj: 0.991245, No Obj: 0.006032, Avg Recall: 0.937500, count: 16
406972 Region Avg IOU: 0.761792, Class: 0.988208, Obj: 0.913878, No Obj: 0.008042, Avg Recall: 1.000000, count: 8
406973 Region Avg IOU: 0.767354, Class: 0.998280, Obj: 0.959904, No Obj: 0.007833, Avg Recall: 1.000000, count: 18
406974 Region Avg IOU: 0.776140, Class: 0.995342, Obj: 0.919506, No Obj: 0.005813, Avg Recall: 1.000000, count: 13
406975 Region Avg IOU: 0.802867, Class: 0.971687, Obj: 0.970485, No Obj: 0.006588, Avg Recall: 0.947368, count: 19
406976 Region Avg IOU: 0.828973, Class: 0.989360, Obj: 0.994962, No Obj: 0.007314, Avg Recall: 0.666667, count: 12
406977
406978 (next mAP calculation at 440700 iterations)
406979 Last accuracy mAP@0.5 = 77.39 %, best = 80.23 %
406980 440681: 0.117429, 0.127100 avg loss, 0.000100 rate, 1.124000 seconds, 28203456 images
406981 Loaded: 0.000000 seconds
406982 Region Avg IOU: 0.747063, Class: 0.989613, Obj: 0.823959, No Obj: 0.007866, Avg Recall: 0.608696, count: 23
406983 Region Avg IOU: 0.737409, Class: 0.998695, Obj: 0.871223, No Obj: 0.006131, Avg Recall: 0.875000, count: 8

```



Una volta trovato il valore massimo di mAP, si dovrà andare a identificare a quale iterazione è riferito. Per fare questo basterà controllare la prima serie di numeri della riga sottostante a quella in cui è indicato il valore del map.

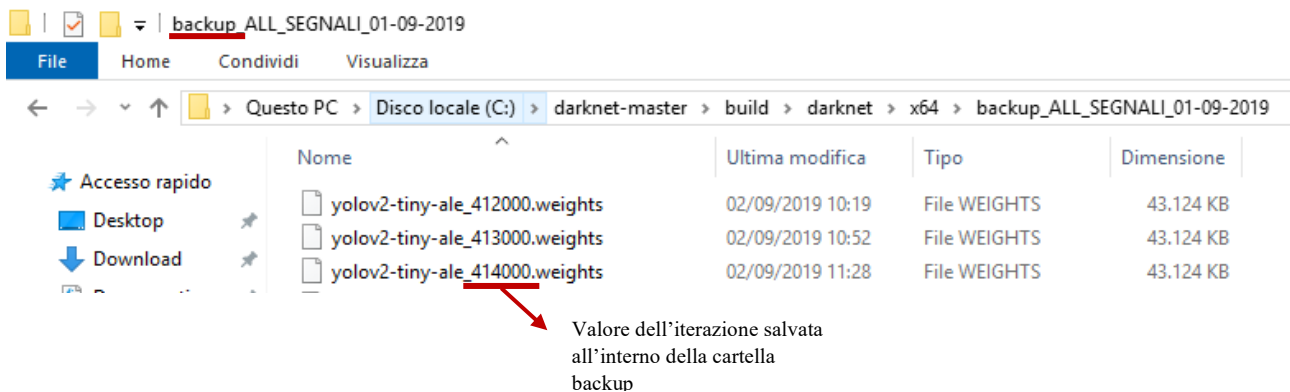
```

(next mAP calculation at 440700 iterations)
Last accuracy mAP@0.5 = 77.39 %, best = 80.23 %
440679: 0.117429, 0.127100 avg loss, 0.000100 rate, 1.124000 seconds, 28203456 images

```

Identificato quindi questo valore bisognerà confrontarlo con i valori presenti all'interno della cartella backup dove viene salvato ogni 1000 iterazioni un file .weights necessario per la validazione. Una volta entrati in questa cartella saranno presenti una serie di file identificati da un numero, quel numero identifica il numero di iterazione. A questo punto sarà necessario confrontare il valore di iterazione letto precedentemente sul log con i valori più vicini presenti della cartella backup ed annotarsi questi numeri (in generale è sufficiente annotarsi il valore superiore e quello inferiore). Individuati poi questi valori, bisognerà tornare sul file log, premere Ctrl+f e cercare uno alla volta i due numeri di iterazione seguiti dal segno “ : ” es 410000: . Attraverso questa ricerca si dovrà andare a visualizzare e a confrontare il valore del mAP dell'iterazione inserita con il massimo valore del mAP ottenuto e bisognerà scegliere

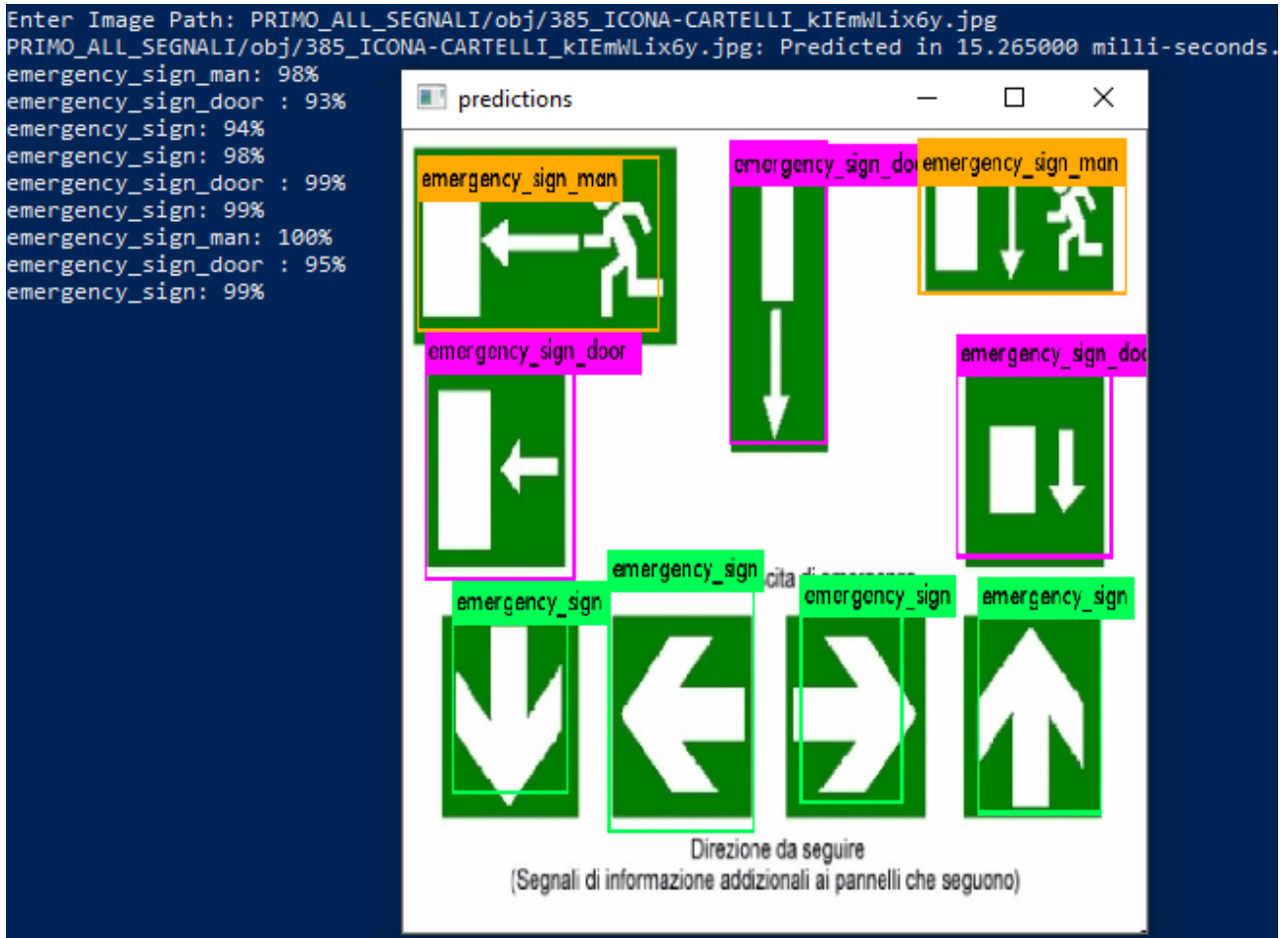
l'iterazione avente mAP più alto e utilizzare i pesi riferiti a quella iterazione per la validazione.



Una volta quindi identificato il file .weights si potrà far partire la validazione. Per fare questo sarà necessario aprire windows powershell e copiare queste 2 righe:

```
cd C:\darknet-master\build\darknet\x64 ;  
./darknet.exe detector test data/giulio.data cfg/yolov2-tiny-ale.cfg  
backup_PRIMO_ALL_SEGNALI/yolov2-tiny-ale_491000.weights -i 0 -thresh 0.25
```

La prima riga servirà per aprire la cartella darknet la seconda invece per configurare la validazione. Nella seconda riga si dovrà andare ad inserire il percorso per il file .data, per il file .cfg e per il file .weights che sarà quello riferito all'iterazione con mAP più alto precedentemente identificato. Premendo invio partirà la validazione. Ora sarà necessario aprire il file test presente all'interno della cartella contenente il dataset e incollare volta per volta ogni percorso immagine. Una volta incollato il percorso e dopo aver premuto invio il sistema restituirà una serie di valori percentuali che rappresentano la precisione del rilevamento con affiancato il tag del rilievo in cui è indicato il nome della classe rilevata.



Questi valori poi insieme all'immagine e al percorso file dovranno essere riportati su una tabella excel per valutarne la precisione. Nel foglio excel infine si dovrà valutare con 3 criteri:

- TP (TRUE POSITIVE)
- FP (FOLSE POSITIVE)
- FN (FOLSE NEGATIVE)

Il comportamento della rete. Dunque bisognerà valutare con TP se la rete è stata in grado di riconoscere l'oggetto in modo corretto, FP se è stato rilevato un oggetto non presente nell'immagine, FN se non è stato rilevato un oggetto invece presente nell'immagine.

PRIMO_ALL_SEGNALI/obj/360_uscita-di-emergenza-a-sinistra-20105_6pxdKLGhR3.jpg  253		100,00% TP
--	--	------------

Completata la fase di validazione, il foglio excelle, oltre a contare il numero di TP, FP, FN, restituirà dei valori di Precision, Recall e F1. Questi valori derivano da semplici calcoli matematici, esposti qui di seguito.

$$\text{Precision} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Positive}}$$

Il valore Precision indica quanti oggetti sono stati etichettati correttamente in confronto a tutti quelli etichettati.

$$\text{Recall} = \frac{\textit{True Positive}}{\textit{True Positive} + \textit{False Negative}}$$

Il valore di Recall invece indica quanti oggetti sono stati etichettati correttamente in confronto a tutti quelli presenti.

$$\text{F1} = 2 \times \frac{\textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

Il valore di F1 infine considera sia la precision sia il recall ed è una sorta di media dei due. Il punteggio F1 è migliore se esiste una sorta di equilibrio tra precision e recall nel sistema. Al contrario, il punteggio F1 non è così elevato se una misura viene migliorata a spese dell'altra. Ad esempio, se P è 1 e R è 0, il punteggio F1 è 0. Questi valori sono importanti perché serviranno per valutare il comportamento della rete e per poter confrontare i vari allenamenti tra di loro.



## 4 Risultati

### 4.1 Dataset utilizzati

Durante il lavoro svolto per ottenere sufficienti dati e per cercare di ottenere dei buoni risultati si è sottoposta la rete a diversi allenamenti con dataset differenti.

Dataset utilizzati:

- PRIMO\_ALL\_SEGNALI
- SECONDO\_ALL\_SEGNALI
- TERZO\_ALL\_SEGNALI
- TRAINING\_Dataset\_SIGN\_FE

Ognuno di questi dataset è stato creato appositamente per raggiungere determinati scopi e per valutare il comportamento della rete a seconda delle immagini a lei fornite.

Come primo dataset è stato utilizzato il dataset PRIMO\_ALL\_SEGNALI in cui all'interno sono state inserite tutte le immagini in possesso, utilizzando sia immagini originali (539) sia immagini aumentate (834). In questo dataset si è cercato di incrementare il numero delle immagini aumentate per le classi emergencysign ed emergencysingdoor poiché avendo come immagini originali un numero inferiore rispetto alla classe emergencysingman si è cercato di limare il quanto più possibile questa differenza.

Nel secondo allenamento effettuato invece utilizzando il dataset SECONDO\_ALL\_SEGNALI si sono utilizzate solamente foto originali senza fare ricorso ad immagini aumentate. Questo allenamento quindi si è svolto con 539 immagini, volendo appositamente cercare di ridurre il numero di immagini per avere un riscontro sul comportamento della rete con un numero inferiore di informazioni.

Il terzo dataset TERZO\_ALL\_SEGNALI invece è stato creato cercando di mettere in difficoltà la rete. Infatti sono state scelte per lo più immagini sia originali sia aumentate con all'interno più di un oggetto in modo da incentivare la rete ad allenarsi con immagini ricche di oggetti, campo in cui YOLO presenta qualche criticità.

Il dataset TRAINING\_Dataset\_SIGN\_FE è stato l'ultimo dataset creato con immagini solamente originali. Questo dataset presenta la particolarità di essere stato creato unendo due diversi campi di oggetti. Infatti esso comprende sia immagini con uscite di emergenza sia immagini di estintori ed è stato creato come esperimento per capire anche in questo caso come si sarebbe comportata la rete

Le immagini utilizzate per creare i dataset sono state prese da due differenti fonti, sia da immagini scattate personalmente da smartphone sia da immagini scaricate.

Tabella riassuntiva:

Solo originali	Classe emergencysigndoor	Classe emergencysignman	Classe emergencysign	Classe fireextinguisher
Immagini scattate	11	23	11	243
Immagini scaricate	65	429	42	382
Totale	76	452	53	500

All'interno dei dataset sono stati inseriti 4 diverse classi di oggetto:


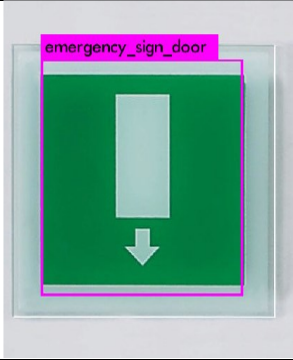
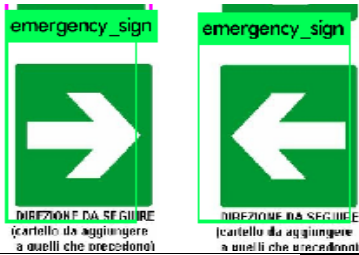

emergencysignman	
emergencysigndoor	
emergencysign	
fireextinguisher	

Tabella riassuntiva sui vari dataset utilizzati per gli allenamenti:

Dataset summary	N. immagini totali	N. immagini originali	N. immagini aumentate
PRIMO_ALL_SEGNALI	1373	539	834
SECONDO_ALL_SEGNALI	539	539	0
TERZO_ALL_SEGNALI	310	155	155
TRAINING_Dataset_SIGN_FE	1039	1039	0

Dataset summary	N. emergencysigndoor*	N. emergencysignman*	N. emergencysign*
PRIMO_ALL_SEGNALI	542	1389	381
SECONDO_ALL_SEGNALI	113	562	84
TERZO_ALL_SEGNALI	142	458	106
TRAINING_Dataset_SIGN_FE	113	562	84 + 625 fireextinguisher

Dataset summary	N. immagini per test	N. foto scattate totali	N. foto scaricate totali
PRIMO_ALL_SEGNALI	251	86	1287
SECONDO_ALL_SEGNALI	101	43	496
TERZO_ALL_SEGNALI	60	86	224
TRAINING_Dataset_SIGN_FE	201	-	-

\*indica il numero degli oggetti presenti all'interno di ogni immagine

## 4.2 Allenamenti e validazioni

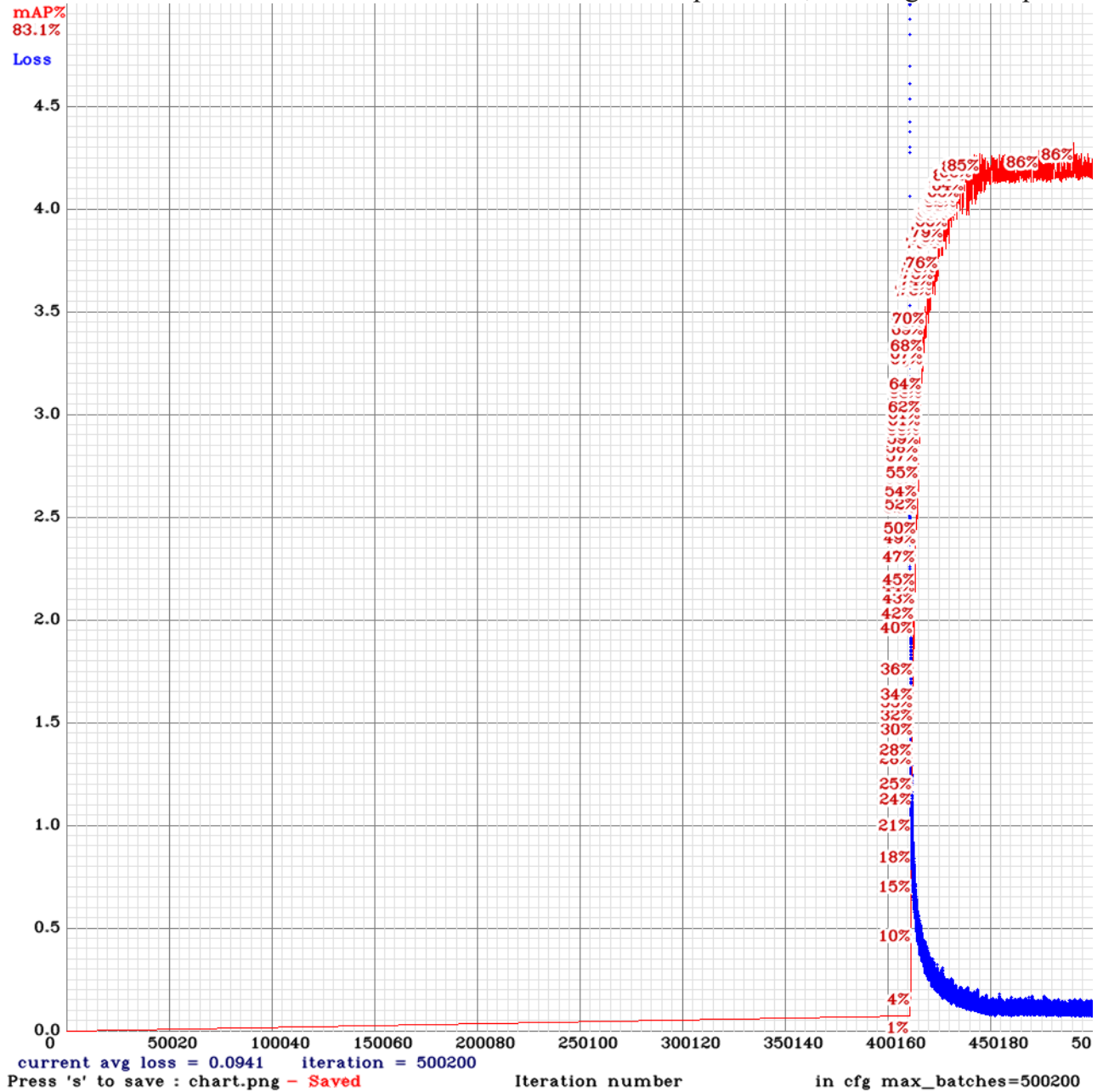
Utilizzando i dataset precedenti sono stati effettuati svariati allenamenti per capire al meglio il funzionamento della rete e per trovare il massimo valore di mAP ottenibile per formare una rete performante e sicura per il rilevamento di oggetti. Utilizzando quindi dataset contenenti quantità di informazioni diverse le une dalle altre sono stati ottenuti questi risultati.

Data Allenamento	Dataset utilizzato	Risultati
19/08/2019	PRIMO_ALL_SEGNALI	mAP=86,4%. Iterazione 491000
01/09/2019	SECONDO_ALL_SEGNALI	mAP=80,98%. Iterazione 441000
03/09/2019	PRIMO_ALL_SEGNALI	mAP=85,02%. Iterazione 482000
05/09/2019	TERZO_ALL_SEGNALI	mAP=84,46%. Iterazione 457000. mAP massimo raggiunto 84,87% con l'iterazione 457602 di cui però non abbiamo i pesi.
11/09/2019	TRAINING_Dataset_SIGN_FE	mAP = 70,76%. Iterazione 434000

I valori massimi di mAP raggiunti da ogni singolo allenamento sono stati recuperati consultando i rispettivi log e chart di seguito riportati.

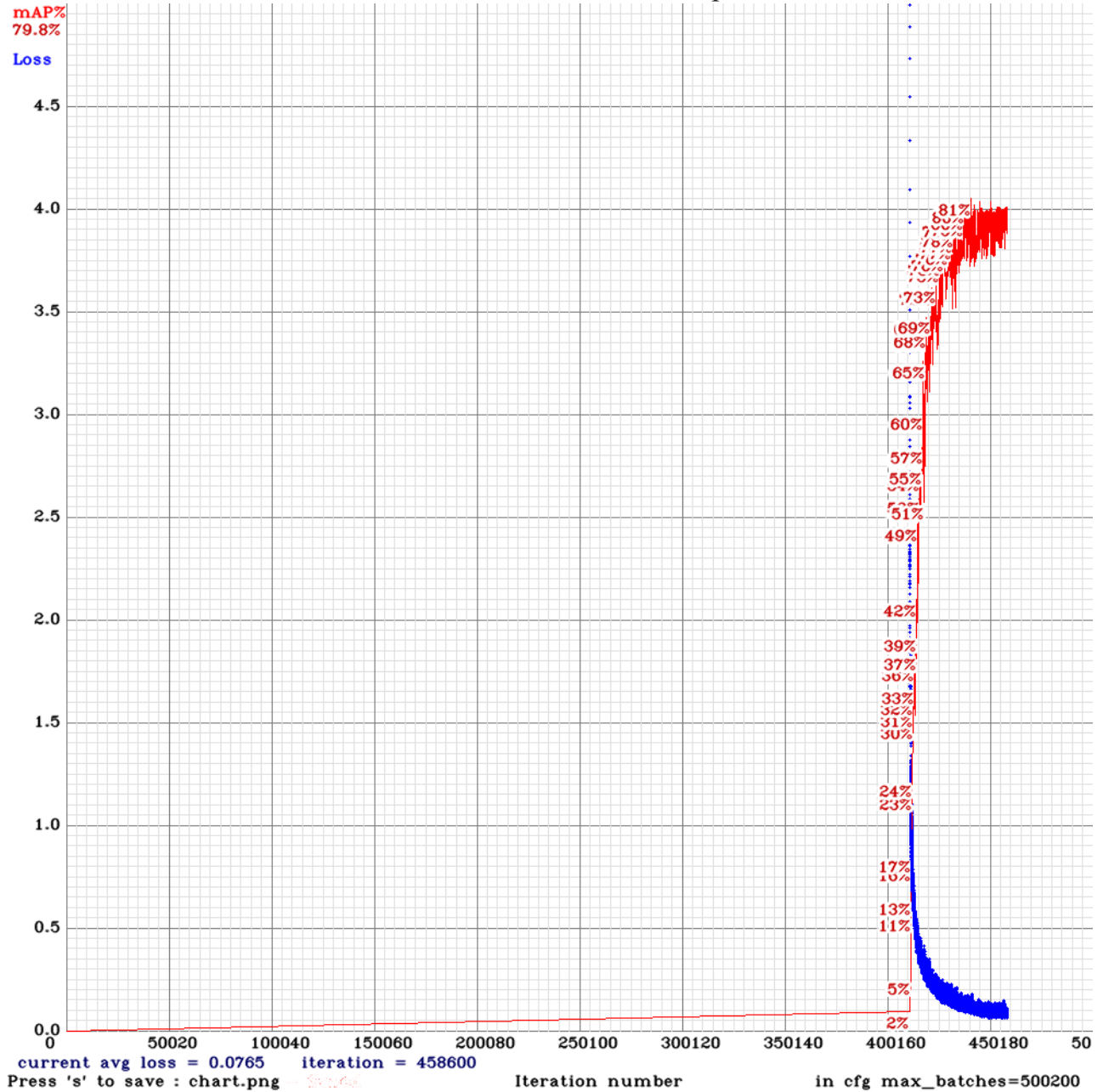
## Allenamento del 19/08/2019

Il primo allenamento è stato effettuato utilizzando come dataset PRIMO\_ALL\_SEGNALI, un dataset contenente 1373 immagini totali, 20% (251 immagini) utilizzato per il test. Utilizzate sia immagini originali sia immagini aumentate. È stato effettuato un allenamento unico per le 3 differenti classi. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign. L'allenamento ha restituito un valore di mAP massimo pari a 86,4%. Log andato perso.



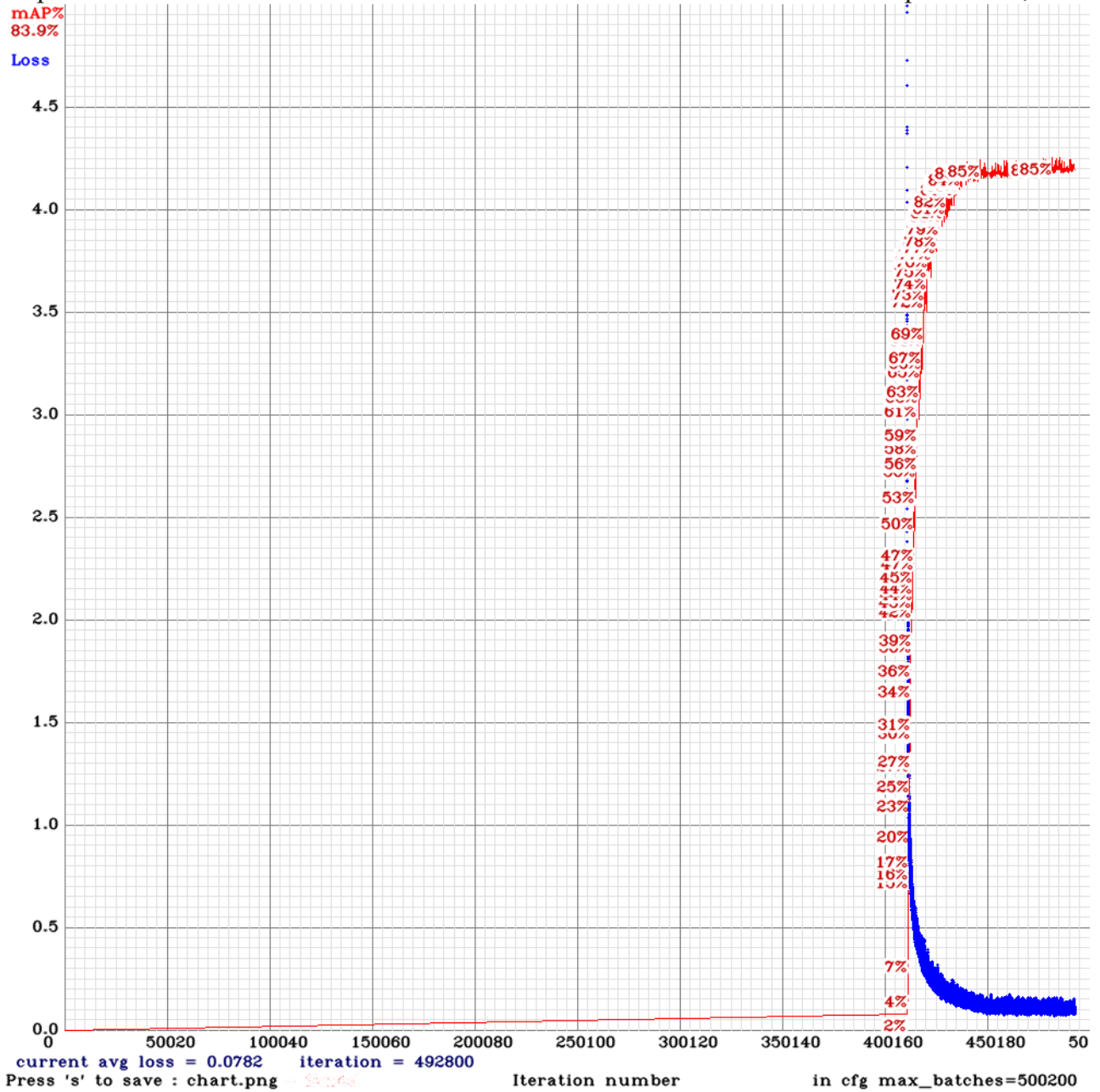
## Allenamento del 1/09/2019

Il secondo allenamento è stato effettuato utilizzando come dataset SECONDO\_ALL\_SEGNALI, un dataset contenente 539 immagini totali non aumentate, 20% (101) utilizzate per il test. UTILIZZATE SOLO IMMAGINI ORIGINALI E NON QUELLE AUMENTATE.. È stato effettuato un allenamento unico per le 3 differenti classi. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign. L'allenamento ha restituito un valore di mAP massimo pari a 80,98%.



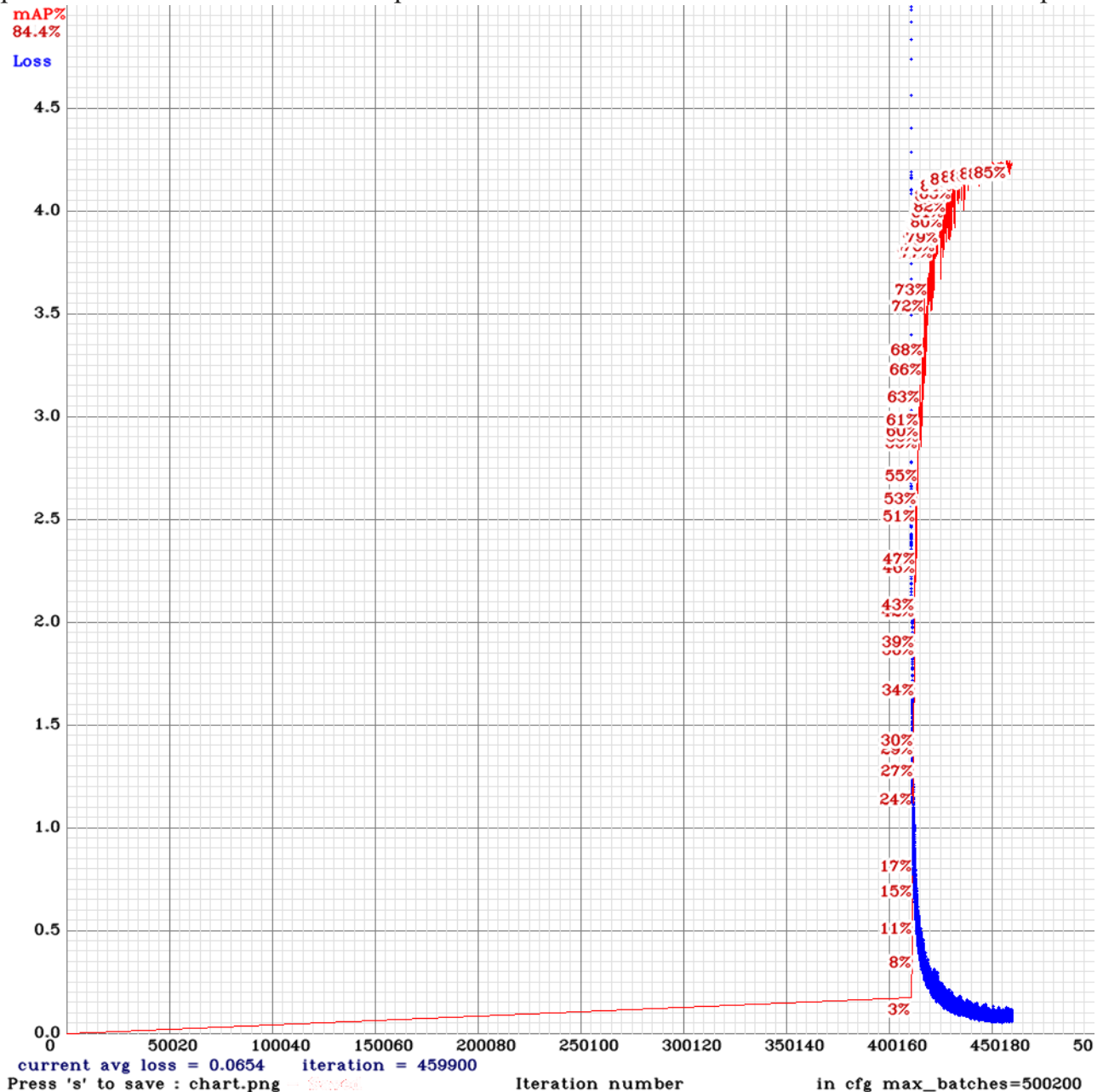
## Allenamento del 03/09/2019

Il terzo allenamento è stato effettuato per recuperare il log andato perso durante il primo allenamento ed è stato utilizzando come dataset PRIMO\_ALL\_SEGNALI, lo stesso dataset utilizzato in precedenza. L'allenamento ha restituito un valore di mAP massimo pari a 85,02%.



## Allenamento del 05/09/2019

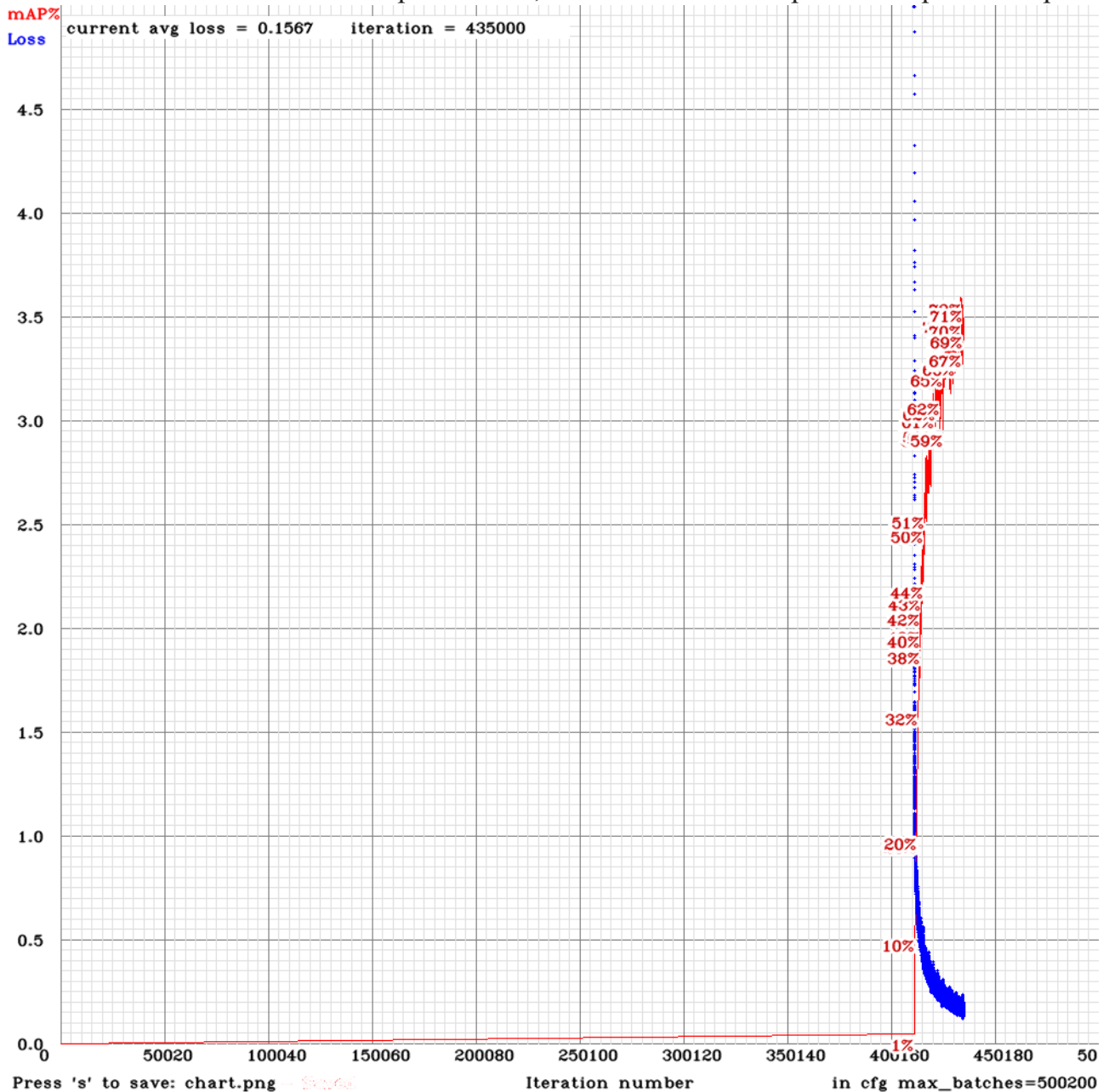
Il quarto allenamento è stato effettuato utilizzando come dataset TERZO\_ALL\_SEGNALI, un dataset contenente 310 immagini totali (155 originali), 20% (60) utilizzate per il test. Utilizzate sia immagini originali sia immagini aumentate. È stato effettuato un allenamento unico per le 3 differenti classi. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign. L'allenamento ha restituito un valore di mAP massimo pari a 84,87% di cui però non abbiamo i pesi quindi al fine della validazione è stato considerato un mAP pari al 84,46% in modo da poter recuperare i pesi.





## Allenamento del 11/09/2019

L'ultimo allenamento è stato effettuato utilizzando come dataset TRAINING\_Dataset\_SIGN\_FE, un dataset contenente 1039 immagini totali, 201 utilizzate per il test. UTILIZZATE SOLO IMMAGINI ORIGINALI E NON QUELLE AUMENTATE. Questo allenamento a differenza degli altri è stato effettuato utilizzando un dataset unico contenente sia le 3 classi di uscite di emergenza sia la classe degli estintori. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign, fireextinguisher. L'allenamento ha restituito un valore di mAP massimo pari a 71,98% di cui però non abbiamo i pesi quindi al fine della validazione è stato considerato un mAP pari al 70,76% in modo da poter recuperare i pesi.



In seguito ad ogni singolo allenamento poi è stata effettuata la validazione di cui si riportano qui di seguito i risultati.

#### Allenamento del 19/08/2019

True Positive	406
False Positive	63
False Negative	26
Precision	86.57%
Recall	93.98%
<b>F1</b>	<b>90.12%</b>

#### Allenamento del 1/09/2019

True Positive	145
False Positive	26
False Negative	13
Precision	84.8%
Recall	91.77%
<b>F1</b>	<b>88.15%</b>

#### Allenamento del 05/09/2019

True Positive	200
False Positive	44
False Negative	24
Precision	81.97%
Recall	89.29%
<b>F1</b>	<b>85.47%</b>

#### Allenamento del 11/09/2019

True Positive	376
False Positive	73
False Negative	45
Precision	83.74%
Recall	89.31%
<b>F1</b>	<b>86.44%</b>

### 4.3 Risultati degli allenamenti

Data Allenamento	Dataset utilizzato	Descrizione dataset	Descrizione allenamento	Risultati	Note	Configurazione allenamento
19/08/2019	PRIMO_ALL_SEGNALI	1373 immagini totali, 20% (251 immagini) utilizzato per il test. Utilizzate sia immagini originali sia immagini aumentate	Utilizzato allenamento unico per le 3 categorie differenti di immagini. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign.	mAP=86,4%. Iterazione 491000. Precision=83.57%, Recall=93.98%, F1=90.12%.	log andato perso, si proverà ad utilizzare il log dell'allenamento del 03/09/2019	The original .cfg file was modified in its following features: batch=64, this means we will be using 64 images for every training step; subdivision=8, the batch will be divided by 8; classes=3, the number of categories we want to detect; filters=40, from the previous formula; learning rate=0.001, advised by the developer of YOLO in order to avoid false minimum point..Allenamento effettuato con yolov2-tiny-ale.weights, yolov2-tiny-ale.cfg.
01/09/2019	SECONDO_ALL_SEGNALI	539 immagini totali non aumentate, 20% (101) utilizzate per il test. UTILIZZATE SOLO IMMAGINI ORIGINALI E NON QUELLE AUMENTATE.	Utilizzato allenamento unico per le 3 categorie differenti di immagini. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign.	mAP=80,98%. Iterazione 441000. Precision=84.80%, Recall=91.77%, F1=88,15%.		The original .cfg file was modified in its following features: batch=64, this means we will be using 64 images for every training step; subdivision=8, the batch will be divided by 8; classes=3, the number of categories we want to detect; filters=40, from the previous formula; learning rate=0.001, advised by the developer of YOLO in order to avoid false minimum point..Allenamento effettuato con yolov2-tiny-ale.weights, yolov2-tiny-ale.cfg.
03/09/2019	PRIMO_ALL_SEGNALI	1373 immagini totali, 20% (251 immagini) utilizzato per il test. Utilizzate sia immagini originali sia immagini aumentate	Utilizzato allenamento unico per le 3 categorie differenti di immagini. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign.	mAP=85,02%. Iterazione 482000. Precision=81,97%, Recall=89,29%, F1=85,47%.	Eseguito allenamento uguale a quello del 19/08 per recuperare il log. Validazione non effettuata.	The original .cfg file was modified in its following features: batch=64, this means we will be using 64 images for every training step; subdivision=8, the batch will be divided by 8; classes=3, the number of categories we want to detect; filters=40, from the previous formula; learning rate=0.001, advised by the developer of YOLO in order to avoid false minimum point..Allenamento effettuato con yolov2-tiny-ale.weights, yolov2-tiny-ale.cfg.
05/09/2019	TERZO_ALL_SEGNALI	310 immagini totali (155 originali), 20% (60) utilizzate per il test. Utilizzate sia immagini originali sia immagini aumentate.	Utilizzato allenamento unico per le 3 categorie differenti di immagini. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign.	mAP=84,46%. Iterazione 457000. Precision=81,97%, Recall=89,29%, F1=85,47%.	UTILIZZATE PER LO PIÙ IMMAGINI CONTENENTI PIÙ DI UN SEGNALE IN MODO DA METTERE IN DIFFICOLTÀ LA RETE.	The original .cfg file was modified in its following features: batch=64, this means we will be using 64 images for every training step; subdivision=8, the batch will be divided by 8; classes=3, the number of categories we want to detect; filters=40, from the previous formula; learning rate=0.001, advised by the developer of YOLO in order to avoid false minimum point..Allenamento effettuato con yolov2-tiny-ale.weights, yolov2-tiny-ale.cfg. mAP massimo raggiunto 84,87%, 457602 iterations
11/09/2019	TRAINING_Dataset_SIGN_FE	1039 immagini totali, 201 utilizzate per il test. UTILIZZATE SOLO IMMAGINI ORIGINALI E NON QUELLE AUMENTATE.	Utilizzato allenamento unico per le 3 categorie di uscite di emergenza e per gli estintori. Classi utilizzate : emergencysigndoor, emergencysignman, emergencysign, fireextinguisher.	mAP = 70,76%. Iterazione 434000. Precision=83,74%, Recall=89,31%, F1=86,44%.	Utilizzato allenamento unico includendo anche estintori.	Network chosen Yolo tiny V2 with pretrained weights.The original .cfg file was modified in its following features: batch=64, this means we will be using 64 images for every training step; subdivision=8, the batch will be divided by 8; classes=4, the number of categories we want to detect; filters=45, from the previous formula; learning rate=0.001, advised by the developer of YOLO in order to avoid false minimum point. >mAP massimo raggiunto 71,98%, 23867 iterations.

## 5 Conclusioni

Durante questa fase di sperimentazione e allenamento, si è potuto sottoporre la rete neurale a diversi allenamenti utilizzando diversi dataset. Dataset creati con l'elaborazione di immagini acquisite attraverso la fotocamera dello smartphone e attraverso l'utilizzo del web scaricando da internet immagini raffiguranti uscite di emergenza. Sottoponendo poi la rete ai vari allenamenti è stato possibile, una volta effettuata la validazione di quest'ultimi, estrarre i risultati in termini di mAP ed F1 per ciascun allenamento. Infine, con l'obiettivo di ottenere la più accurata e precisa rete in grado di riconoscere tutti quegli oggetti facenti parte del sistema di segnalazione delle uscite di emergenza in modo sicuro ed affidabile si è potuto studiare e confrontare i dati e i risultati raccolti durante questo periodo di sperimentazione e trarre alcune conclusioni. Dagli esiti riportati si è riscontrato che i migliori risultati in termini di mAP e F1 e quindi di precisione generale della rete sono stati raggiunti con il primo allenamento (del 19/08/2019) utilizzando un dataset di discrete dimensioni con circa 1373 immagini totali. Inoltre, considerando il dataset "secondo\_all\_segnali" formato da 539 immagini e confrontandolo con il dataset "primo\_all\_segnali" contenente 1373 immagini si è potuto accertare come allenando la rete con un dataset decisamente inferiore rispetto al primo di circa la metà delle informazioni sono stati ottenuti risultati del tutto soddisfacenti con un mAP e un F1 non di molto inferiori. Infatti, il secondo allenamento effettuato il 01/09/2019 ha restituito un mAP del 80,98% ed un F1 del 88% mentre il primo allenamento del 19/08/2019 un mAP del 86,4% ed un F1 del 90% così da poter dedurre che un aumento considerevole di informazioni all'interno del dataset non corrisponde necessariamente ad un incremento proporzionale in termini di precisione e affidabilità della rete. Il ragionamento è rafforzato ancora di più se si prende in analisi l'allenamento del 05/09/2019 effettuato con il dataset "terzo\_all\_segnali" contenente 310 immagini di circa 4 volte inferiore rispetto al primo e che ha restituito un valore di mAP del 84,46% e di F1 del 85%. Confrontando i risultati di questi primi 3 allenamenti e esaminando la composizione di ogni dataset si è potuto affermare inoltre che ai fini dell'allenamento, considerando il dataset "terzo\_all\_segnali" formato da 310 immagini di cui soltanto 155 originali, non conti tanto la quantità di immagini presenti all'interno del dataset ma bensì la qualità di quest'ultime. Per ottenere un efficace allenamento si dovrà cercare di utilizzare immagini veritiere, immagini chiare e che rispettino la realtà dell'oggetto, evitando immagini doppie, immagini con oggetti poco chiari o troppo piccoli.

Inoltre, dall'esperienza maturata durante questo periodo, si è verificato come siano effettivamente riscontrabili i limiti di Yolo tiny v2 circa il rilevamento di oggetti ravvicinati, come dichiarato nella letteratura. Infatti, sottoponendo la rete a immagini con all'interno più di un oggetto da rilevare, la rete ha presentato alcune difficoltà e alcune lacune nell'individuare con esattezza e precisione gli oggetti, fallendo nel riconoscimento di alcuni di essi.

In conclusione quindi si può affermare che la rete allenata in data 19/08/2019 con dataset PRIMO\_ALL\_SEGNALI che ha restituito un mAP pari al 86.40% e che durante la validazioni ha raggiunto valori di precision pari a 86%, di recall 94% e di F1 90% risulta affidabile per gli scopi da noi ricercati e che potrà essere utilizzata in futuro per altre

applicazioni come l'implementazione all'interno di visori per realtà mixata (es. Microsoft Hololens) in modo da poter permettere ad un operatore di individuare in tempo reale i vari segnali di uscita di emergenza e raccogliere i dati relativi ai fini dell'arricchimento informativo di un modello BIM abbattendo notevolmente il tempo e i costi di post-elaborazione. si ritiene conclusa questa prima parte di sperimentazione per quanto riguarda l'addestramento, il rilevamento e la classificazione dei vari segnali di uscita di emergenza, e si lascia il materiale fruibile all'Università Politecnica delle Marche per poter completare gli studi e utilizzare la rete addestrata per scopi futuri.

## 6 Sitografia

<https://it.wikipedia.org/wiki/Digitalizzazione>

<https://www.unocloudbackup.it/digitalizzare-i-dati-benefici-e-vantaggi/>

<https://www.a-sapiens.it/bim/news/cosa-e-bim-ne-un-software-ne-un-programma-ma-una-metodologia/>

<https://www.ingenio-web.it/21786-digitalizzazione-del-danno-sismico-di-edifici-su-piattaforma-bim-attraverso-tecniche-di-intelligenza-artificiale>

[http://www.isprambiente.gov.it/files/pubblicazioni/statoambiente/Focus\\_La\\_natura\\_in\\_citta\\_ultimo.pdf#page=41](http://www.isprambiente.gov.it/files/pubblicazioni/statoambiente/Focus_La_natura_in_citta_ultimo.pdf#page=41)

<https://www.spindox.it/it/blog/yolo-riconoscimento-oggetti-real-time/>

<https://pjreddie.com/publications/>

[https://medium.com/@jonathan\\_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088](https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088)

<http://pjreddie.com/yolo9000/>

## **7 Ringraziamenti**

Si ringraziano tutti coloro che mi hanno accompagnato e sostenuto durante questo mio percorso universitario ed in particolare la mia famiglia che mi ha seguito durante questa esperienza formativa motivandomi e credendo costantemente in me.

Si ringraziano inoltre tutti i professori dell'Università politecnica delle Marche ed in particolare il mio relatore Massimo Lemma ed i correlatori Alessandra Corneli e Massimo Vaccarini che mi hanno aiutato nella stesura di questa tesi. Grazie a tutti!