



UNIVERSITÀ  
DEGLI STUDI  
**FIRENZE**

Scuola di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea Magistrale in Informatica  
Curriculum: *Data Science*

TECNICHE DI APPRENDIMENTO  
PROFONDO PER RICONOSCIMENTO DI  
OGGETTI IN VIDEO TERMICI

DEEP LEARNING TECHNIQUES FOR  
OBJECT DETECTION IN THERMAL VIDEOS

FEDERICO SCHIPANI

Relatore: Prof. *Marco Bertini*

Anno Accademico 2018-2019

Federico Schipani: *Tecniche di apprendimento profondo per riconoscimento di oggetti in video termici*, Corso di Laurea Magistrale in Informatica, © Anno Accademico 2018-2019

---

## INDICE

---

Introduzione	6
1 Object Detection	9
1.1 Storia della object detection	9
1.1.1 Evoluzione delle tecniche	10
1.1.2 Dataset	18
1.1.3 Metriche	24
1.2 Detector basati su metodi tradizionali	24
1.3 Detector basati su Deep learning	24
1.3.1 Estrattori di feature	24
1.3.2 One Stage Detector	24
1.3.3 Two Stage Detector	25
1.4 RetinaNet	25
1.4.1 Focal Loss	25
2 SISTEMA	27
2.1 Dataset	27
2.1.1 KAIST Multispectral Pedestrian Dataset	27
2.1.2 FLIR Thermal Starter Dataset	30
2.1.3 Video di RFI	32
2.2 Addestramento iniziale di RetinaNet	32
2.2.1 Transfer Learning	32
2.2.2 Addestramento sulle immagini RGB	35
2.2.3 Passaggio alle immagini termiche su KAIST	38
2.2.4 ALTRA ROBA DA SCRIVERE	40
2.3 Data Augmentation	40
2.3.1 Auto Augment	40
2.3.2 Rand Augment	42
3 Esperimenti	45
3.1 Organizzazione dei dataset	45
3.2 Esperimenti iniziali	45
3.3 Data Augmentation	45
4 Conclusioni	47
A Codice di IoU over time	49
B Implementazione di RandAugment su RetinaNet	51
Acronimi	52



---

## ELENCO DELLE TABELLE

---

Tabella 1	Schema riassuntivo delle categorie di Transfer Learning	36
Tabella 2	Test complessivo delle performance dopo l'addestramento	37
Tabella 3	Risultati della valutazione separata	37
Tabella 4	Test complessivo delle performance dopo l'addestramento	39
Tabella 5	Risultati della valutazione separata	39



---

## ELENCO DELLE FIGURE

---

Figura 1	Storia della Object Detection [1]	10
Figura 2	Evoluzione della detection multiscala [1]	12
Figura 3	Evoluzione della regressione basata su Bounding Box (BB) [1]	13
Figura 4	Evoluzione del context priming [1]	15
Figura 5	Evoluzione della Non Maximum Suppression [1]	16
Figura 6	Evoluzione di Hard Negative Mining [1]	17
Figura 7	Statistiche riassuntive di Microsoft - Common Object in COntext (MS-COCO) [2]	23
Figura 8	Heatmap riguardante la posizione dei pedoni	30
Figura 9	Differenze tra apprendimento tradizionale e transfer learning	33
Figura 10	Transfer learning da COCO a KAIST	36
Figura 11	Esempio di predizioni, in verde la <i>ground truth</i> in rosso le predizioni.	37
Figura 12	Transfer learning da KAIST Visibile a KAIST Termico	38
Figura 13	Schema di funzionamento di AutoAugment	40
Figura 14	Esempio di policy con 5 sub-policy, immagine tratta da [3]	41



---

## INTRODUZIONE

---



# 1

---

## OBJECT DETECTION

---

L'Object Detection è un *task* legato al mondo della *computer vision* che consiste nel rilevare e classificare istanze di oggetti in immagini o video.

Negli ultimi anni, grazie soprattutto all'avvento delle Graphics Processing Unit (GPU), c'è stato un incremento notevole del potere computazionale. Questo ha portato a sviluppare tecniche sempre più raffinate allo scopo di raggiungere prestazioni sempre migliori.

Sempre grazie allo sviluppo di hardware sempre più potente l'interesse sta sempre più virando verso il mondo del *Deep Learning*. In questo capitolo cercheremo di classificare le varie metodologie con cui si porta a compimento la *Object Detection*. La letteratura sui detector è molto disomogenea e variegata, prenderemo quindi come riferimento i lavori di *Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng and Rong Qu [4]* e *Zhengxia Zou, Zhenwei Shi, Yuhong Guo and Jieping Ye [1]*.

### 1.1 STORIA DELLA OBJECT DETECTION

Una prima, ma importante distinzione va fatta tra il periodo pre e post *deep learning*. Il primo periodo va dagli inizi degli anni 2000 fino al 2014. Il secondo periodo, dove hanno preso il sopravvento tecniche basate sul *deep learning*, va dal 2014 fino ai nostri giorni. Quest'ultime tecniche possono essere a loro volta divise in altre due categorie, *One Stage Detector* e *Two Stage Detector* (sarà il caso di tradurre in italiano?) il cui sviluppo procede in maniera parallela. In Figura 1 è presente uno schema riassuntivo con tutte le pietre miliari raggiunte durante lo sviluppo di tecniche per il rilevamento di oggetti.

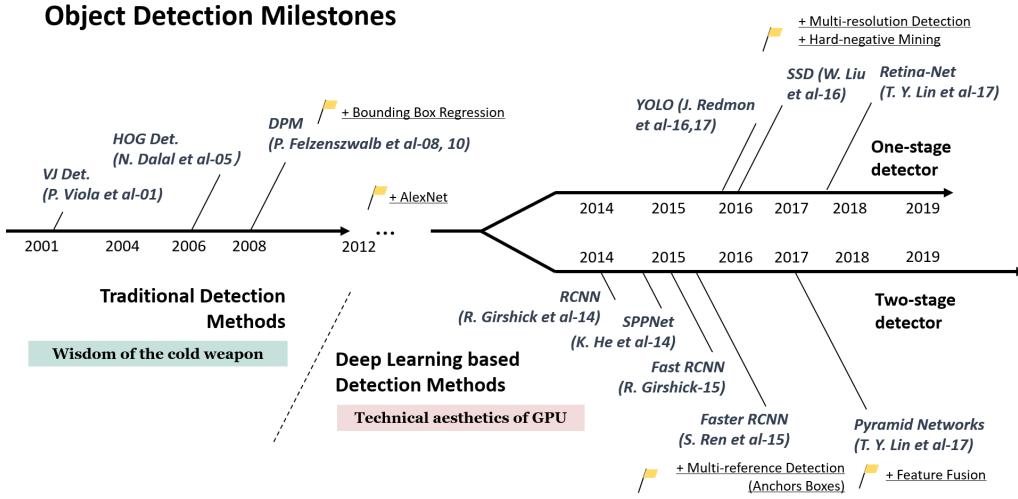


Figura 1.: Storia della Object Detection [1]

### 1.1.1 Evoluzione delle tecniche

Durante questo ventennio i detector più famosi sono stati costruiti usando come mattoncini delle tecniche sviluppate ed affinate via via nel tempo. Queste tecniche sono di diverso tipo ed hanno subito evoluzioni di cui faremo una disamina nel proseguo di questa sottosezione.

#### Prime tecniche

Storicamente una delle prime tecniche si basava sul una teoria cognitiva chiamata *Recognition by Components* [5], ed è stata per molto tempo la base di alcuni lavori riguardanti il riconoscimento di immagini e la rilevazione di oggetti [6] [7] [8].

Nel passato alcuni ricercatori hanno formulato soluzioni al problema usando misure di similarità tra le componenti di un oggetto, tra la forma o i contorni, tra cui *Distance Transforms* [9], *Shape Contexts* [10] e *Edgelet* [11].

I risultati iniziali erano molto promettenti, tuttavia quando la rilevazione è diventata più complicata queste tecniche hanno iniziato a mostrare i propri limiti, motivo per cui il passaggio al Machine Learning è stato quasi naturale. Le prime metodologie basate su questo approccio risalgono ad un periodo inquadrabile prima del 1998, in questo caso la detection si basava su modelli statistici costruiti sopra le apparenze degli oggetti da rilevare. Il primo di questi modelli statistici nasce nel 1991, chiamato *Eigenfaces* [12] [13], riesce in laboratorio a riconoscere volti in tempo reale.

Successivamente, fino al 2005, l'evoluzione ha portato a tecniche in cui si cambiava radicalmente la rappresentazione, intesa come feature, delle immagini. Lo scopo era quindi apprendere come trasformare un'immagine da insieme di pixel a insieme di coefficienti *wavelet*. Grazie alla sua efficienza, tra tutte le trasformate, quella a prendere piede fu la *Haar wavelet*. Dal 2005 al 2012 c'è stato un passaggio a rappresentazioni basate sul gradiente.

Intorno al 1990 iniziano a fare capolino le prime Convolutional Neural Network (CNN) [14] le quali però non hanno avuto grandi applicazioni per via dell'elevato costo computazionale rispetto alle risorse disponibili ai tempi. I modelli realizzati con CNN non potevano essere quindi molto profondi, e per questo avevano forti limitazioni. Per ridurre questo elevato costo computazionale sono stati effettuati miglioramenti come *space displacement network* [15]. Tramite questi perfezionamenti si è arrivato ad estendere ogni layer della CNN in maniera tale da riuscire ad estrarre le feature da un'immagine in un solo passaggio. Queste possono essere considerate un po' le antenate di quelle che attualmente chiamiamo Fully Convolutional Network (FCN) [16] [17].

### *Detection multiscala*

Uno degli aspetti più interessanti della ricerca si basa sulla rilevazione di oggetti con diverse misure o diverse proporzioni. Come è possibile bedere in Figura 2 la soluzione a questo problema ha attraversato varie fasi.

**FEATURE PIRAMIDALI E FINESTRE SCORREVOLI** L'idea dietro questa tecnica è abbastanza basilare, infatti dopo aver estratto le feature da un'immagine quello che viene fatto è far scorrere una finestra rettangolare di dimensione generalmente fissa per effettuare il rilevamento e la classificazione di oggetti.

Dal 2004 al 2014 sono stati creati un sacco di detector basati su questa filosofia, il problema è che erano stati disegnati con l'intento specifico di rilevare oggetti con proporzioni fisse. Ricercatori come R. Girshick *et al.* iniziarono a cercare soluzioni a questo problema, arrivando a formulare un modello mistura [18] composto da più modelli addestrati su oggetti con differenti proporzioni. Sono state sviluppate anche altre soluzioni, basate questa volta sull'addestrare modelli separati per ogni istanza di oggetto dell'insieme di addestramento [19] [20]. Le limitazioni di tutte queste tecniche risidono nel fatto che i dataset più moderni sono molto

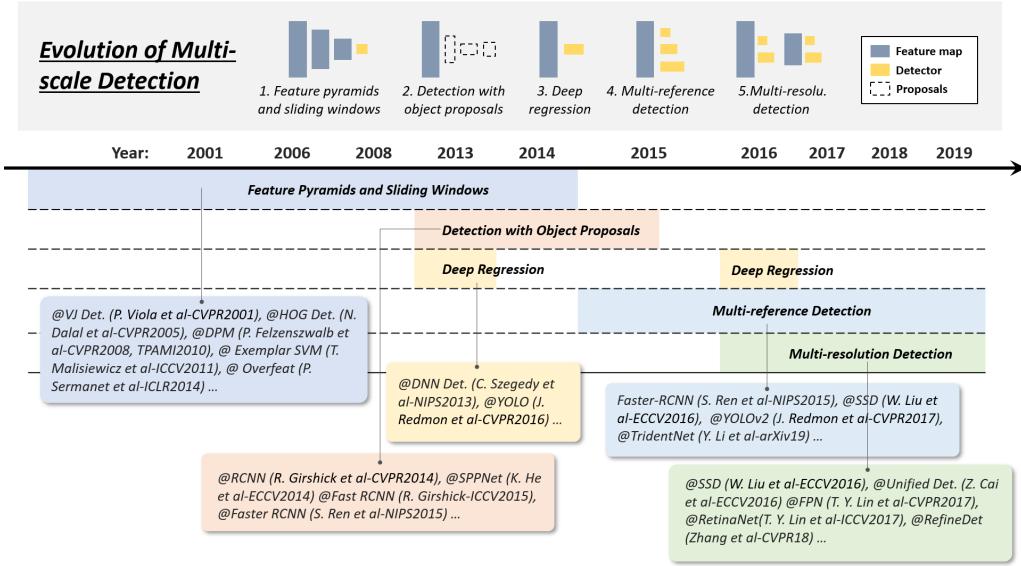


Figura 2.: Evoluzione della detection multiscala [1]

diversificati, quindi nel corso del tempo queste tecniche sono diventate sempre meno precise ed utilizzabili. Ciò ha portato allo sviluppo di *Object Proposal*.

**OBJECT PROPOSAL** Il primo avvistamento di *Object Proposal* risale al 2010 in un task di rilevazione di oggetti [21]. Possiamo definire una regione come un'area di un'immagine contenente pixel che hanno caratteristiche comuni tra di loro. L'idea dietro questa tecnica è creare regioni non etichettate con classi che potenzialmente possono contenere qualunque tipo di oggetto, e lo scopo è riuscire a rilevare oggetti di varie misure e scale pur non dovendo necessariamente svolgere una ricerca esaustiva con finestre scorrevoli.

Per ottenere queste regioni di pixel su un'immagine ci sono vari modi, discussi in parte da J. Hosang *et al.* in [22].

**DEEP REGRESSION** Questa tecnica, sviluppata dal 2013 al 2016 si basa sull'idea di predire direttamente le coordinate della BB contenente l'oggetto usando come feature quelle estratte da un modello di *deep learning* [23]. Il vantaggio fondamentale di questo approccio è l'efficienza e la velocità di implementazione, mentre uno svantaggio è la bassa accuratezza di localizzazione specialmente su piccoli oggetti.

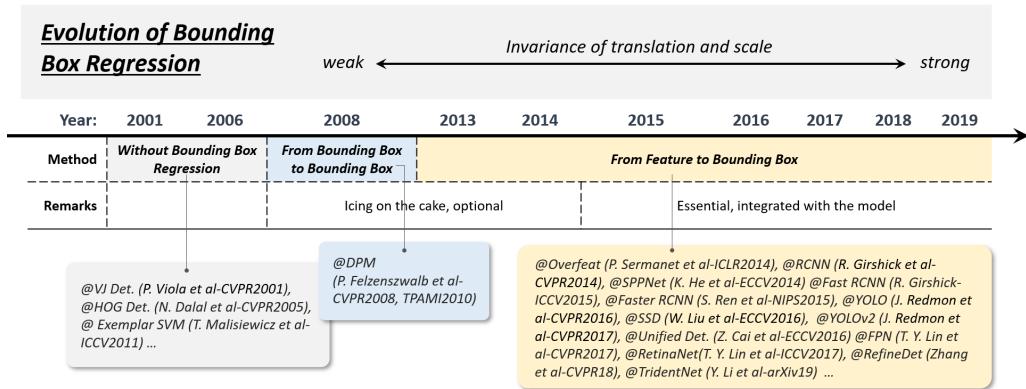


Figura 3.: Evoluzione della regressione basata su BB [1]

**MULTI REFERENCE DETECTION** Questo approccio è il più usato per il rilevamento di oggetti con scale differenti e si basa sull'uso di un insieme di finestre rettangolari, che possono variare in dimensione e proporzioni, applicate sull'immagine, dette anche *Anchor Boxes* [24, 25, 26]. Sulla base di queste regioni rettangolari viene poi effettuata una predizione della BB.

**MULTI RESOLUTION DETECTION** Negli ultimi anni un altro approccio che ha preso piede si basa sul rilevare oggetti di con dimensioni differenti a layer differenti [25, 27, 28, 29]. Basti pensare alle CNN che nel corso della propagazione dell'immagine in input, grazie alla loro composizione, formano una serie di feature piramidali. Diventa quindi più facile rilevare oggetti grandi nei layer più profondi e viceversa diventa più facile rilevare oggetti piccoli nei layer meno profondi.

### Regressione basata su Bounding Box

Questo insieme di metodologie ha lo scopo di affinare la posizione delle BB basandosi sulle rilevazioni effettuate tramite *Object Proposal* o *Anchor Boxes*, descritte in 1.1.1. Uno schema riassuntivo dell'evoluzione tecnica di questi approcci è in Figura 3. I primi detector non raffinavano in alcun modo la posizione delle BB, anzi molte volte usavano direttamente l'output derivato da un algoritmo basato su finestre scorrevoli. L'unico modo per ottenere rilevazioni più precise era quindi costruire modelli piramidali molto densi e assicurarsi di far scorrere la finestra lungo tutta l'immagine.

**DA BOUNDING BOX A BOUNDING BOX** I primi ad usare una forma di regressione per aumentare la precisione sulle BB sono stati P. F. Felzenszwalb *et al.* in DPM [30] formulando la soluzione con il metodo dei minimi quadrati. Per scendere più nel dettaglio dobbiamo considerare un modello con feature piramidali. Nel modello proposto in [30] viene usata l'intera configurazione di  $z$  per predire la BB riguardante quell'oggetto. In breve l'implementazione è effettuata tramite una funzione  $g(z)$  che restituisce come output le coordinate  $(x_1, y_1)$  e  $(x_2, y_2)$  della BB. Dopo la fase di addestramento del modello viene usato l'output di  $g(z)$  per un'ulteriore fase di addestramento dove tramite il metodo dei minimi quadrati si impara a predire  $x_1, y_1, x_2$  e  $y_2$  partendo da  $g(z)$ . C'è da specificare che questo tipo di ottimizzazione è stato implementato a livello di post-processing, quindi risulta del tutto opzionale.

**DA FEATURE A BOUNDING BOX** A differenza del tipo di ottimizzazione proposto in precedenza, con l'introduzione delle *Faster RCNN* [24] nel 2015 la regressione è implementata a livello di rilevamento dell'oggetto e quindi viene addestrata insieme al modello. Inoltre, sempre confrontandolo con quanto detto prima, vengono usate anche diverse funzioni di loss da minimizzare che risultano più robuste rispetto a quella usata nei minimi quadrati. Degli esempi possono essere la *smooth-L1* o la *root-square*.

### *Valutazione del contesto*

Generalmente gli oggetti che vengono rilevati dai sistemi di detection sono immersi in un contesto. Il cervello umano durante la fase cognitiva trae vantaggio dal riconoscere un contesto, motivo per cui le tecniche spiegate nel proseguo di questa sottosezione provano a emulare questa capacità degli umani. Una breve storia di come queste tecniche si sono evolute è in Figura 4.

**CONTESTO LOCALE** Per contesto locale si intendono tutte le informazioni visive che fanno parte dell'area prossima ai contorni di un oggetto. Sin dagli anni 2000 Sinha e Torralba in [31] hanno provato che includere del contesto locale migliora le prestazioni ai fini del rilevamento di volti. Dalal e Triggs in [32] hanno dimostrato che introdurre una piccola porzione di sfondo migliora i risultati anche nel rilevamento dei pedoni.

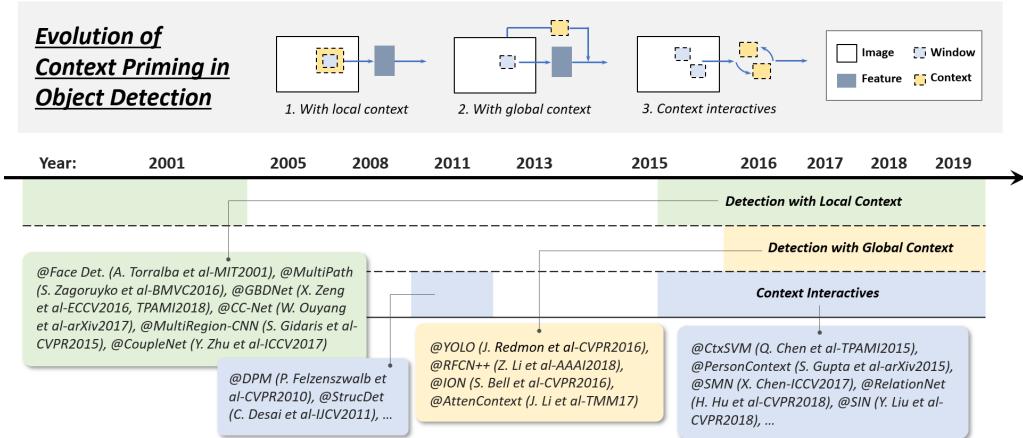


Figura 4.: Evoluzione del context priming [1]

**CONTESTO GLOBALE** Il contesto globale può essere considerato come una fonte di informazione aggiuntiva riguardante la scena in cui l'oggetto da rilevare è immerso. Un metodo usato all'inizio per inglobare nella detection informazioni sul contesto globale era realizzare delle statistiche che riassumevano la totalità degli elementi compresi nella scena [33]. In lavori più moderni catalogabili come modelli di *deep learning* sono state intraprese due strade, la prima è quella di inglobare il contesto tramite campi recettivi sempre più ampi, a volte anche più dell'immagine stessa [23] o usare l'operazione di *pooling* delle CNN [34]. La seconda via per ottenere informazioni dal contesto globale è pensare ad esso come un flusso di informazioni sequenziali ed usare Recurrent Neural Network (RNN) [35, 36].

**CONTESTO DERIVATO DALLE INTERAZIONI** L'ultima contestualizzazione che si può dare ad un oggetto riguarda le sue interazioni con ciò che lo circonda. Per interazioni si possono intendere tutte quei vincoli o dipendenze che può avere l'obbiettivo della rilevazione. Recentemente in alcuni lavori sono state analizzate questo tipo di informazioni contestuali ai fini del miglioramento della detection. Questi miglioramenti possono essere divisi in due macrocategorie, la prima di queste è quella in cui si esplorano le relazioni tra oggetti individuali [18, 37, 38, 39, 40]. Nella seconda categoria fanno parte i lavori nei quali vengono prese in considerazione le relazioni che ci sono tra gli oggetti e la scena che li circonda [41, 42].

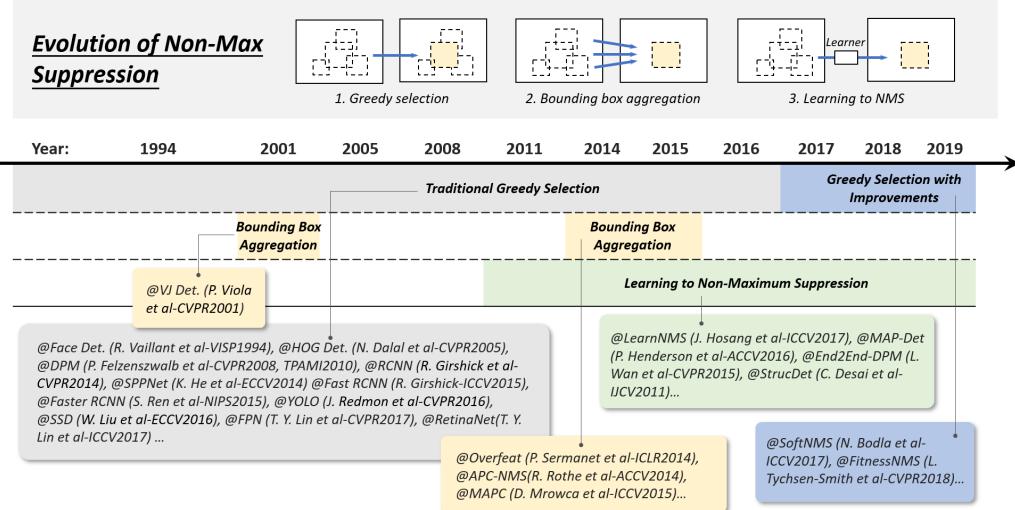


Figura 5.: Evoluzione della Non Maximum Suppression [1]

### Non Maximum Suppression

Per Non-Maximum Suppression (NMS) si fa riferimento a tutte quelle tecniche di post processing per ridurre il fenomeno delle BB duplicate. Questo fenomeno si concretizza quando, per la stessa rilevazione, ci sono più BB che lo circondano con confidenze molto simili tra di loro. L'evoluzione di queste tecniche nel corso del tempo è possibile vederla in Figura 5.

**GREEDY** La maniera più semplice per attuare la NMS è con un algoritmo di tipo *greedy*, infatti per un insieme di BB sovrapposte si considera solamente quella con la confidenza massima, mentre le altre vengono scartate. La sua semplicità, che da un certo punto di vista può anche essere vista come un punto di forza, può anche essere fonte di debolezze in quanto un algoritmo *greedy* non sempre porta all'ottimalità. Si possono infatti verificare casi in cui la BB con massima confidenza non ricopre tutto l'oggetto, o ancora peggio le BB di oggetti vicini tra di loro possono essere scaricate erroneamente.

**AGGREGAZIONE DI BOUNDING BOX** L'aggregazione di BB vicine tra di loro è un altro approccio per attuare la NMS [43, 44, 45, 46]. L'aggregazione può essere fatta sia attraverso algoritmi di *clustering*, sia combinando le BB sovrapposte in un'unica singola detection.

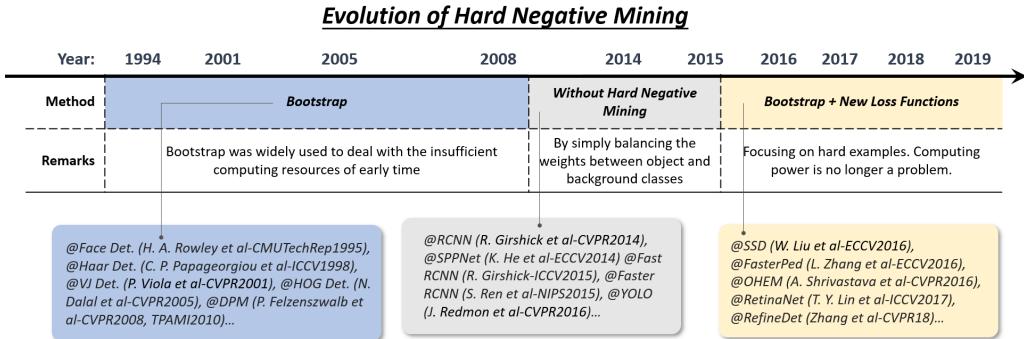


Figura 6.: Evoluzione di Hard Negative Mining [1]

**IMPARARE AD APPLICARE NON-MAXIMUM SUPPRESSION** Approcci più recenti per migliorare le già sopraccitate tecniche di NMS riguardano l'apprendimento automatico [47, 37, 48, 49]. L'idea dietro questi metodi è trattare la NMS alla stessa stregua di un filtro che assegna nuovi valori di confidenza a tutte le detection e quindi bisognoso di una fase di addestramento.

### *Hard Negative Mining*

Uno dei problemi a cui bisogna far fronte quando si tratta la *Object Detection* è lo sbilanciamento tra gli oggetti che vogliamo rilevare e conseguentemente classificare e tutto quello che non ci interessa. La prima soluzione che potrebbe venire in mente per risolvere questo problema è addestrare il modello a riconoscere lo sfondo, ma questo porta a risultati distruttivi durante l'addestramento in termini di efficienza. Tecniche di Hard Negative Mining (HNM) servono proprio a risolvere queste problematiche. Una breve storia è possibile vederla in Figura 6.

**BOOTSTRAP** Per Bootstrap si fa riferimento ad un gruppo di tecniche attraverso le quali si fa iniziare la fase di addestramento prendendo con dello sfondo. Successivamente quando si rilevano esempi di sfondo classificati erroneamente si aggiungono al processo di addestramento. Questo approccio risulta efficiente in quanto si evita di addestrare il modello su milioni di esempi di sfondi [43, 50, 51].

**HNM IN DETECTOR BASATI SU DEEP LEARNING** Recenemente modelli come Faster RCNN e You Only Look Once (YOLO) per risolvere questo problema bilanciano i pesi tra finestre con esempi negativi e positivi. Nonostante tutto però non viene risolto completamente il problema

di sbilanciamento, quindi c'è stato un ritorno al *bootstrap*. Un altro modo per risolvere lo sbilanciamento è l'introduzione di nuove funzioni di Loss, come ad esempio la Focal Loss in RetinaNet [52].

### 1.1.2 Dataset

L'insieme dei dati con cui addestrare e testare le performance dei modelli via via sviluppati nel corso del tempo ha subito un'evoluzione. Costruire dataset sempre più grandi e con meno bias è sempre stato un obiettivo principale che si ponevano i ricercatori, tutto ciò per realizzare dei benchmark che mettessero sempre di più a dura prova i nuovi modelli. Nel seguito di questa sezione analizzeremo in breve alcuni dei dataset più famosi nell'ambito della computer vision, con particolare attenzione per alcuni incentrati sulla rilevazione di pedoni.

**MIT PED.** Risale all'inizio del nuovo millennio ed è uno dei primi dataset che ha come scopo il riconoscimento di pedoni. Rispetto agli standard odierni risulta molto piccolo in quanto contiene circa 509 immagini usabili per l'addestramento e 200 usabili per la fase di test. [53]

**INRIA** Risalente al 2005, nasce dall'esigenza di creare un dataset dove la detection diventasse più complicata rispetto a quello fatto dal MIT [32]. Contiene 1805 immagini ad una risoluzione di  $64 \times 128$  pixel di esseri umani. Le immagini usate per l'insieme di training sono 2478, ovvero 1239 immagini, contenenti esempi positivi, prese dal totale più le stesse immagini ma riflesse secondo l'asse delle y.

**PASCAL VOC** Pascal Visual Object Classes (VOC) è collocabile in un periodo che va dal 2005 al 2012 [54, 55]. VOC consiste di due parti complementari, la prima è un dataset pubblico e disponibile per esperimenti e benchmark, la seconda è una sfida annuale. Nel corso degli anni ne sono state sviluppate diverse versioni, individuabili dal pattern VOCANNO. I task effettuabili su questo dataset spaziano classificazione di immagini alla object detection passando anche per il rilevamento di azioni. Per la sfida nel 2007 sono state raccolte solo immagini dal social network Flickr. Le immagini sono molto variegate, ad esempio come scritto nell'articolo di presentazione del dataset [54] ci possono essere motociclette in diverse pose e forme, come può essere il veicolo per strada o come soggetto principale del fotogramma. Un altro esempio è la classe "person", mentre in molti dataset con questa classe ci si riferisce ad un pedone

in VOC2007 con questa classe si fa riferimento ad un essere umano in diversi contesti. Per realizzare il dataset sono state definite delle keyword con cui effettuare query su Flickr. Queste suddette keyword sono state definite sulla base delle classi degli oggetti che si desiderava annotare. Tramite queste query sono state recuperate 500.000 immagini, non prendendo in considerazione la data di acquisizione, il nome del fotografo, la location e via discorrendo. Le query venivano effettuate a gruppi di 100.000 immagini alla volta, di cui venivano selezionate casualmente solamente alcuni fotogrammi ed inseriti nel dataset. Questa operazione è stata ripetuta fino ad ottenere la quantità desiderata di file. L'operazione successiva è stata quella di eliminare i duplicati o comunque immagini molto somiglianti tra di loro, una volta fatto questo sono passati ad annotarle. Di queste 500.000 immagini agli annotatori ne sono state presentate 44.269. Gli annotatori avevano la facoltà di scartare alcune immagini se le ritenevano non adatte ad essere annotate o avevano una confidenza bassa sull'eventuale annotazione da effettuare. Nonostante ciò è stato scoperto un'elemento che portava a del bias all'interno del dataset. L'elemento in questione riguardava il modo in cui le immagini sono state recuperate. Quando si effettua una query su Flickr il server restituisce le immagini in ordine cronologico di upload sulla piattaforma. Il dataset è stato realizzato nel Gennaio 2007, quindi buona parte delle immagini erano ambientate in un contesto natalizio o perlopiù invernale. Con VOC2008 il problema è stato risolto aggiungendo una data casuale all'interno della query per recuperare le immagini.

**CALTECH PEDESTRIAN DATASET** Il *Caltech Pedestrian Dataset* [56] nasce nel 2009 ed è molto più ampio di tutti gli altri dataset visti in precedenza. Le immagini sono state ricavate da circa 10 ore di video girato a 30 frame al secondo in un ambiente urbano con traffico regolare. È quindi presente un numero di frame che è nell'ordine di grandezza di  $10^6$ . La telecamera con cui sono state acquisite le registrazioni è stata piazzata su un'autovettura con un guidatore che attraversava le strade di Los Angeles guidando in maniera normale. La risoluzione delle immagini è di  $640 \times 480$  e come conseguenza del fatto che il sistema è stato più volte smontato e rimontato ci sono piccole variazioni nella posizione della telecamera.

Sono stati annotati 250.000 fotogrammi, per un totale di 350.000 BB e 2300 pedoni univoci. Di tutti i fotogrammi, circa il 50% non hanno pedoni, mentre il 30% del totale ne ha almeno due. In media un pedone è visibile per un tempo di 5 secondi. Come su dataset più recenti i

pedoni sono raggruppati secondo la loro distanza dal guidatore. In particolare abbiamo che i pedoni sono vicini se la loro altezza è maggiore di 80 pixels, sono ad una distanza media se la loro altezza è compresa tra 30 ed 80 pixels, mentre sono considerati lontani se la loro altezza è al più 30 pixels. Questa discretizzazione riguardante la distanza dei pedoni è stata realizzata seguendo la distribuzione delle altezze degli stessi all'interno del dataset. Sono state inoltre prese in considerazione statistiche riguardanti l'occlusione dei pedoni e la loro posizione rispetto alla telecamera.

Il dataset è stato creato da 11 sessioni dove in ognuna di queste venivano esplorati 5 quartieri. Dopodiché le prime sei sessioni sono state destinate all'addestramento, mentre le rimanenti cinque sono state adibite a test set.

**KITTI** [57] Questo dataset è realizzato a partire da hardware usato per la guida autonoma. La particolarità è che offre informazioni tridimensionali dell'ambiente grazie a dei sensori laser dedicati che mappano il territorio circostante. Per la realizzazione sono state quindi utilizzate due telecamere a colori con una risoluzione di  $1392 \times 512$  pixels, uno scanner laser ed un localizzatore GPS con un'unità di correzione RTK, il tutto orchestrato da un calcolatore su cui girava un database in real time. Tutto questo hardware è stato montato su una station wagon che è stata guidata per dei normali scenari cittadini. Inoltre le annotazioni non sono BB in due dimensioni, ma bensì dei parallelepipedi che avvolgono gli oggetti. Per la loro realizzazione sono stati presi annotatori umani che hanno posizionato BB tridimensionali su oggetti come auto, furgoni, camion, tram, pedoni e ciclisti. Inoltre gli annotatori sono stati istruiti per marcare ogni BB come visibile, semi occlusa, occlusa o troncata.

**ILSVRC** Come per VOC anche ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [58] è una challenge organizzata annualmente dal 2010 al 2017. Inoltre, sempre come per VOC è presente un dataset pubblico disponibile per addestramenti e test. Come per alcuni dei dataset precedentemente introdotti le immagini state prese in parte da Flickr, con l'aggiunta però di altri motori di ricerca. In particolare andremo ad analizzare la costruzione della parte di dataset riguardante la *object detection*, però il dataset contiene anche una parte dedicata alla classificazione di immagini ed un'altra parte dedicata alla rilevazione singola di oggetti all'interno dell'immagine.

Le classi di oggetti presenti sono 200 e sono stati scelte partendo

dalle 1000 classi usate per la classificazione di immagini. Una prima riduzione per arrivare a 494 classi è stata inizialmente effettuata tramite l'eliminazione di label corrispondenti ad oggetti che occupavano gran parte del fotogramma o semplicemente non adatti alla *object detection*. Un ulteriore operazione di unificazione tra classi simili ha portato il numero a 200.

Per il training set le immagini hanno tre provenienze differenti. Le prima sorgente di immagini è la parte di dataset dedicata alla rilevazione singola di oggetti. Per aggiungere ulteriori esempi negativi sono state effettuate prese immagini da motori di ricerca. Le immagini provenienti da queste due prime fonti sono state annotate solamente con un sottonieme delle classi totali. L'ultima sorgente di immagini è la piattaforma Flickr su cui sono state fatte query generiche. Riguardo invece la porzione di dataset dedicata alla validazione e al test la situazione è molto simile in quanto la fonte primaria (77%) è la porzione di ILSVRC2012 adibita alla rilevazione di oggetti singoli, a cui però sono state sottratte le immagini nel quale gli oggetti occupavano più del 50% dell'area totale del fotogramma. Per aggiungere ulteriori esempi, come per il training set, sono state effettuate interrogazioni generiche su Flickr. A differenza dell'insieme di immagini di addestramento, per la validazione e il test, sono state usate tutte e 200 le classi disponibili. In totale per il training set sono presenti circa 458.000 immagini, mentre per il validation e test set sono presenti 46.000 fotogrammi.

**CITYPERSONS** Zhang *et al.* [59] propongono un nuovo dataset derivante da Cityscapes [60], ma invece che focalizzarsi sulla segmentazione delle scene in contesti urbani CityPersons è incentrato sulla rilevazione di pedoni. Infatti per ogni frame di Cityscapes sono stati annotati esseri umani tramite BB.

Le classi usate in questo dataset sono quattro, e variano a seconda della postura del pedone. Con **pedestrian** vengono indicate le persone in piedi, che corrono o camminano, mentre con **rider** si indicano quelle persone che sono alla guida di un mezzo a due ruote. Sono presenti altre due classi per indicare le persone sedute (**sitting person**) e persone con pose inusuali (**other person**).

Viene standardizzato il processo di annotazione dei **pedestrian** e **riders** in quanto le BB hanno tutte la stessa proporzione (0.41), è quindi sufficiente tracciare una linea che va dalla testa ai piedi del pedone per generare una BB delle giuste dimensioni. In questo modo si perfeziona l'allineamento della regione con l'oggetto da rilevare, e quindi si miglio-

rano le prestazioni dell'eventuale modello. Per le rimanenti due classi invece non è stato standardizzato il processo, quindi vengono semplicemente disegnate BB che contengono la persona. Un'altra operazione che è stata effettuata è stata ricercare tra le immagini tutte quelle persone false (manichini, statue, riflessi) per marcarli come regioni da ignorare.

Il dataset è composto da 5000 immagini, con un totale di circa 35000 annotazioni e 13000 regioni da ignorare. A differenza di KITTI e Caltech c'è una densità di persone per frame sette volte superiore ed il numero di individui distinti è prossimo a 20.000, quindi rispettivamente 15 e 3 volte superiore. La suddivisione tra test e train set è la stessa di Cityscapes.

**MS-COCO** [2] Attualmente MS-COCO grazie alla difficoltà di ottenere buone prestazioni è considerato uno dei più interessanti dataset per la *object detection*. Rispetto a ILSVRC ha meno classi, ma contiene molte più immagini ed annotazioni. Il punto di forza di questo dataset è la grande molteplicità di contesti in cui sono stati catturate le immagini e soprattutto la densità di oggetti che rispecchia in maniera del tutto simile quella del mondo reale. Inoltre una proprietà importante è che gli oggetti si trovano quasi sempre in contesti appropriati.

Andando più nello specifico ci si trova davanti ad un dataset che nella sua ultima versione ha, per la *object detection*, circa 165.000 immagini dedicate all'addestramento, e circa 160.000 dedicate alla validazione ed al test. Le classi in totale sono 91. In media ci sono 3.5 categorie diverse e 7.7 oggetti per immagine. In Figura 7 è presente uno schema riassuntivo delle statistiche di questo dataset, ed un'altra cosa che si può notare è che, rispetto ad esempio a ILSVRC e VOC, ci sono molte meno immagini con una sola BB, infatti in MS-COCO solamente il 10% rientra in questa categoria.

**OPEN IMAGES** Open Images Detection (OID) [61], nella versione 5, è un dataset composto da 9.2 milioni di immagini. Ai giorni d'oggi rappresenta una delle sfide più interessanti insieme a ILSVRC e MS-COCO. Come per gli altri dataset anche questo si adatta a più task poiché le annotazioni sono state effettuate a livello di singola immagine, di *object detection*, segmentazione e interazioni tra oggetti. Analizzando l'aspetto riguardante la rilevazione di oggetti ha ancora più classi rispetto a ILSVRC in quanto per la si arriva a 600 differenti label su 1.9 milioni di immagini con BB realizzate per la maggior parte da annotatori professionali interni a Google. In aggiunta i contesti e le ambientazioni in cui sono state catturate le immagini sono molto variegati.

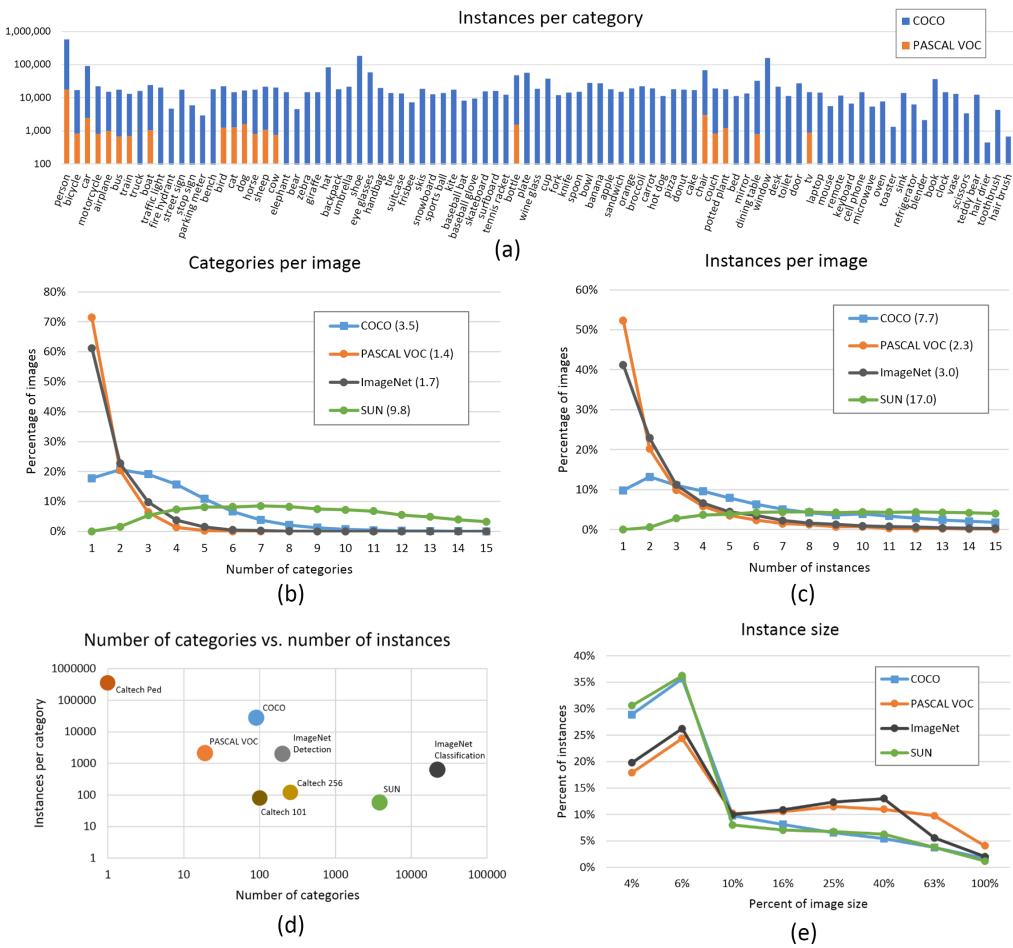


Figura 7.: Statistiche riassuntive di MS-COCO [2]

**EUROCITY** EuroCity è un dataset proposto da Braun *et al.* nel 2019 [62]. Le immagini sono state acquisite da un veicolo in movimento in 31 diverse città europee appartenenti a 12 differenti stati. La risoluzione è di  $1920 \times 1024$  con un framerate di 20 immagini al secondo. Una particolarità delle immagini acquisite da queste telecamere è lo spazio colore che arriva a 16 bit. Quest'alta gamma dinamica si traduce in una più alta qualità dei fotogrammi in condizioni di luce non ottimale, come può essere un controsole o una notturna. Per ogni città in media sono state catturate 1.7 ore di video e le immagini sono state campionate ogni 4 secondi (80 frame). Questo porta ad avere una ripetizione minore di pedoni o oggetti uguali, soprattutto in condizioni trafficate.

### 1.1.3 *Metriche*

## 1.2 DETECTOR BASATI SU METODI TRADIZIONALI

VJ

HOG

DPM

## 1.3 DETECTOR BASATI SU DEEP LEARNING

### 1.3.1 *Estrattori di feature*

### 1.3.2 *One Stage Detector*

YOLO

SSD

DSSD

M2DET

REFINENET

### 1.3.3 Two Stage Detector

RCNN

SPNNET

FASTRCNN

FASTERRCNN

FPN

MASKRCNN

## 1.4 RETINANET

### 1.4.1 Focal Loss



# 2

---

## SISTEMA

---

### 2.1 DATASET

Inizialmente i dataset usati per gli esperimenti sono due. Il primo è KAIST Multispectral Pedestrian Dataset (KAIST MPD) [63], per cui è disponibile un'ampia documentazione, il secondo è un dataset gratuito realizzato dalla FLIR [64] per cui è disponibile una documentazione molto stringata.

#### 2.1.1 KAIST Multispectral Pedestrian Dataset

Il Korea Advanced Institute of Science and Technology (KAIST) propone in [63] un dataset che fornisce coppie di immagini termiche e a colori. La particolarità che offre questo dataset è che le due immagini sono allineate. Inoltre sono state raccolte sufficienti immagini sia diurne che notturne.

**SPECIFICHE HARDWARE** KAIST ha sviluppato una piattaforma basata su una camera a colori, una termica ed un *Beam Splitter*, oltre che ad un supporto a tre assi chiamato *camera jig*. Un *Beam Splitter* è un dispositivo ottico di forma cubica formato in molti casi da due prismi che divide la luce in due parti. In questo caso viene utilizzato per l'allineamento delle due immagini in quanto permette il passaggio dello spettro termico mentre quello visibile viene riflesso. Il dispositivo usato per la realizzazione del dataset è stato costruito a partire da un wafer di silicio zincato.

Le telecamere utilizzate sono una *PointGrey Flea3* per la parte a colori ed una *FLIR-A35* per la parte termica. La prima acquisisce immagini ad una risoluzione di 640x480 pixels con un Field of View (FOV) di 103.6°, mentre la seconda ha una risoluzione di 320x256 con un FOV di 39°. Come si può notare il campo visivo della telecamera visibile è più ampio di quello della telecamera termica, motivo per cui viene sacrificata parte dell'immagine visibile al fine di allineare i due fotogrammi. Il *framerate* è

di 20 FPS.

**CALIBRAZIONE** L’idea per la realizzazione di questa architettura hardware è stata ripresa dal lavoro di Bienkowski *et al.* [65]. Sempre in questo lavoro però non si fa riferimento alla metodologia usata per la calibrazione. Parleremo in questo paragrafo dell’approccio utilizzato per la realizzazione di questo dataset. Innanzitutto è stata calcolata la traslazione fra le due telecamere, applicando la calibrazione stereo. Si può osservare che gli assi ottici delle telecamere al di là della divisione del fascio di luce sono paralleli a causa delle impostazioni tecniche. Di conseguenza, fra i due domini dell’immagine, è presente unicamente una traslazione ed è necessario solamente aggiustare la posizione tramite *camera jig* a tre assi finché la traslazione non diventa nulla. Dopo l’aggiustamento, i due domini sono rettificati fino ad avere la stessa distanza focale virtuale. Al termine di queste procedura, oltre alla focale, i domini condividono i punti principali e sono privi di baseline. Il dominio dell’immagine, virtualmente allineato, ha 640x512 pixel di risoluzione spaziale e un FOV di 39°, analogamente a quella umano. Visto che un pattern a scacchiera convenzionale non è osservabile con telecamera termica, viene invece utilizzata una tavola di calibrazione speciale, con un certo numero di buchi. Quando viene scaldata, si ottiene una differenza di temperatura fra la tavola e i buchi, che possono essere osservati nel termico.

**CORREZIONE DEI COLORI** Per via del passaggio all’interno dei prismi del *Beam Splitter* le immagini catturate, soprattutto nello spettro del visibile, mostrano distorsioni piuttosto evidenti dei colori. Per gestire questo problema è stato deciso di acquisire un fotogramma di riferimento completamente bianco che mostrava distorsioni di colore. Per motivi legati al sensore utilizzato all’interno della telecamera visibile la distorsione del colore può essere considerata come una funzione lineare. Quindi ogni pixel dell’immagine di riferimento può essere usato come coefficiente di correzione per le altre immagini, dividendo il livello di intensità di queste immagini per questi coefficienti.

**ACQUISIZIONE DEI DATI E ANNOTAZIONI** Tutto il marchingegno composto dalle due telecamere, il *Beam Splitter* ed il *Camera jig* è stato montato sul tetto di un’automobile al fine di realizzare immagini egocentriche del traffico. In particolare, come già accennato in precedenza, sono state realizzate raccolte di dati sia di giorno che di notte.

Il numero totale delle coppie di immagini catturate sono 95328 le

quali sono state annotate manualmente con un totale di 103128 BB. Per realizzare le annotazioni è stata usata una versione modificata del Piotr's Computer Vision Toolbox [66]. Le BB sono state annotate con quattro differenti label:

- **person**: individuo singolo ben individuabile
- **people**: individui non distinguibili
- **cyclist**: persone che stanno utilizzando una bicicletta
- **person?**: individuo non ben identificabile per via di fotogrammi molto densi

Inoltre, anche se per lo scopo della tesi non sono stati presi in considerazione, ogni BB ha una corrispondenza temporale che identifica il singolo individuo attraverso i vari frame.

**TRAIN E TEST SET** Per dividere tra *train set* e *test set* è stato usato un criterio ben definito:

- Il numero di pedoni nei due set è simile
- Il numero di frame notturni e diurni nei due set è simile
- I due set non si sovrappongono

#### PROPRIETA

- Attributo di scala: per ogni BB è associato un valore di scala. Questo valore dipende dalla distanza che ha il pedone dall'automobile, ed è giustificato dalla seguente affermazione. Supponendo che una vettura in area urbana viaggia ad una velocità compresa tra 30 e 50 km/h lo spazio di arresto varia tra gli 11 ed i 28 metri. Questo intervallo, scalato opportunamente rispetto alla risoluzione dell'immagine, e considerando anche che l'altezza media di un pedone è di 1.7 metri corrisponde ad un range che va da 45 a 115 pixel. All'interno di questo range vengono definite *medium*, al di sopra *far* ed al di sotto *near*.
- Occlusione: questo attributo associa ad ogni BB un valore che rappresenta l'occlusione del pedone. I valori possibili sono *no occlusion*, *partial occlusion*, *heavy occlusion*. I primi sono circa il 78.6%, i secondi circa il 12.6% e gli ultimi 8.8%.

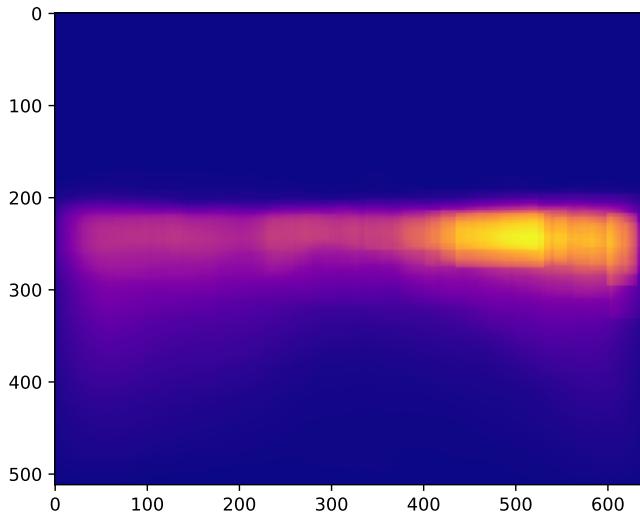


Figura 8.: Heatmap riguardante la posizione dei pedoni

- Posizione: l'impostazione dell'hardware rispecchia il più possibile quello di un essere umano, motivo per cui questo particolare setup concentra il rilevamento di pedoni nell'area centrale dell'immagine, in particolare nel lato destro. Questo è motivato dal fatto che nel paese dove sono stati acquisiti questi dati la guida è sulla destra. In Figura 8 è possibile vedere questo fenomeno.
- Cambio d'aspetto: l'aspetto dei pedoni all'interno del dataset è molto variabile. In condizioni di pieno sole i pedoni sono ben visibili e con dei contorni ben definiti, mentre la differenza di temperatura tra l'ambiente circostante ed il pedone è meno marcata. Quindi nello spettro a colori sono presenti pedoni ben definiti, mentre nel termico no. Di notte invece, per via delle temperature ambientali più basse e per l'assenza di luce si verifica il contrario. In figura **INSERIRE FIGURA** è presente un esempio.

### 2.1.2 FLIR Thermal Starter Dataset

Come già accennato in precedenza la documentazione riguardante questo dataset è molto stringata, limitandosi dunque ad una sola pagina web molto riassuntiva. Cercheremo in questa sezione di parlare degli aspetti che caratterizzano questo dataset.

Il dataset in questione offre immagini termiche con annotazioni e l'equivalente a colori non annotato. A differenza di 2.1.1 le due immagini non sono allineate, quindi non è possibile portare le annotazioni delle immagini termiche sulle immagini a colori. Le immagini sono state acquisite tramite telecamere montate su una vettura e contiene un totale di 14453 immagini, di cui 10228 campionate da video di breve durata e 4224 provenienti da video di 144 secondi. Tutte le immagini sono state acquisite su strade ed autostrade a Santa Barbara, in California. L'arco temporale varia da Novembre a Maggio, nello stessa quantità di giorno e notte. Il meteo è generalmente buono.

Le immagini termiche sono state scattate con una *FLIR Tau2*, mentre quelle RGB con una *FLIR BlackFly*. Entrambi i device sono stati impostati in maniera tale da avere lo stesso FOV e per quanto riguarda il resto sono state lasciate entrambe alle impostazioni di default. Le videocamere sono state posizionate sullo stesso supporto a distanza di circa 1.9 pollici (circa 4.8 centimetri) l'una dall'altra. Il *framerate* è di 2 frame al secondo in scenari densi di annotazioni, mentre in scenari più tranquilli è stato deciso di scendere ad un frame al secondo.

Le annotazioni dove possibile ricalcano i codici adottati dal dataset COCO, ed hanno i seguenti codici:

- 1 People: esseri umani.
- 2 Bicycles: biciclette e motocicli. Questa è l'unica categoria non consistente con il formato adottato da COCO.
- 3 Cars: automobili e veicoli piccoli.
- 18 Dogs: cani
- 91 Other Vehicle: camion, rimorchi e imbarcazioni.

Le annotazioni sono state fatte manualmente da esseri umani ai quali è stato comunicato di fare BB il più piccole possibili e che omettessero piccole parti di oggetti, accessori personali e parti occluse. Inoltre è stato comunicato di non annotare oggetti di piccole dimensioni o molto occlusi e persone delle quali si vede solo braccia o gambe.

### 2.1.3 Video di RFI

## 2.2 ADDESTRAMENTO INIZIALE DI RETINANET

In questa sezione presenteremo i risultati iniziali dell’addestramento di RetinaNet sui due dataset descritti in sezione 2.1. Per lo scopo è stata usata una versione di *RetinaNet* implementata tramite *Keras* reperibile in forma originale al seguente link. Durante lo sviluppo del lavoro di tesi le modifiche al codice originale sono state molteplici, tanto da aver richiesto un *fork* della *repository* originale reperibile al seguente link. Per tenere traccia dell’addestramento è stato usato *Weight & Biases*.

### 2.2.1 Transfer Learning

Inizialmente è stata usata la tecnica del *Transfer Learning*. Una rapida spiegazione del significato di questa espressione ce la fornisce il libro *Deep Learning* di *Goodfellow et al* [67]

Transfer learning and domain adaptation refer to the situation where what has been learned in one setting is exploited to improve generalization in another setting.

Per un’introduzione più dettagliata su cos’è il *transfer learning* e sui vari tipi è stato preso spunto da *A survey on Transfer Learning* di *S. J. Pan e Q. Yang* [68].

La necessità di attuare tecniche di *transfer learning* deriva dal fatto che molti modelli di Machine Learning lavorano bene solo sotto determinate assunzioni, soprattutto quella che i dati di addestramento e di test derivino dallo stesso spazio delle *feature* e dalla stessa distribuzione. I problemi sorgono quando cambia la distribuzione, in quanto è necessario procedere ad una nuova fase di training.

Le casistiche in cui il *transfer learning* è applicabile sono molteplici, ad esempio l’analisi dei sentimenti, dove il compito è classificare le recensioni di un determinato prodotto in positive o negative. Per un compito del genere il primo passo da effettuare è la raccolta e l’annotazione di recensioni. Successivamente è necessaria una fase di addestramento di un modello usando come dati le recensioni precedentemente raccolte ed annotate. Lo scopo è poter usare lo stesso modello per vari prodotti, in questo caso però si va incontro al problema che le distribuzioni dei dati su diversi prodotti possono differire anche di molto. La soluzione sarebbe quindi di annotare altre recensioni, ma richiederebbe uno sforzo notevole.

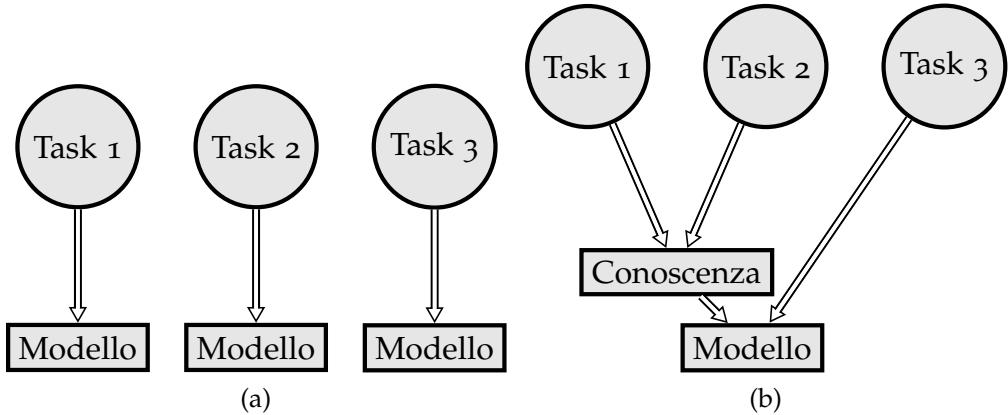


Figura 9.: Differenze tra apprendimento tradizionale e transfer learning

L’idea è quindi di adattare un modello di classificazione, addestrato su alcuni prodotti, per aiutare la fase di addestramento su articoli differenti. Le differenze tra processi di apprendimento tradizionali e *transfer learning* sono mostrate in Figura 9.

**NOTAZIONE PRELIMINARE** Per introdurre un po’ più nello specifico le varie tipologie di *transfer learning* è necessario definire alcuni concetti. Il primo di questi è il *Dominio*  $\mathcal{D}$ , definito come una tupla  $\mathcal{D} = \{\mathcal{X}, P(X)\}$ , dove  $\mathcal{X}$  è lo spazio delle *feature* e  $P(X)$  con  $X = \{x_1, x_2, \dots, x_n\} \in \mathcal{X}$  è una distribuzione di probabilità marginale. In generale due domini possono essere considerati differenti se hanno differenti spazi delle *feature* o distribuzioni differenti.

Dato uno specifico dominio  $\mathcal{D} = \{\mathcal{X}, P(X)\}$  è possibile definire il *task*. Un *task*  $\mathcal{T}$  è una tupla  $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$  con  $\mathcal{Y}$  spazio dei *label* e  $f(\cdot)$  funzione di predizione. La particolarità del *task* è il fatto che non è osservabile ma può essere appreso dai dati di addestramento, che consistono di una coppia  $\{x_i, y_i\}$  con  $x_i \in \mathcal{X}$  e  $y_i \in \mathcal{Y}$ . Una volta completata la fase di addestramento dovrebbe essere possibile usare la funzione  $f(\cdot)$  per predirre il *label*  $f(x)$  corrispondente ad una nuova istanza  $x$ .

Chiameremo il dominio sorgente  $\mathcal{D}_S$  ed il dominio target  $\mathcal{D}_T$ . In particolare avremo a disposizione anche i dati  $D_S$  del dominio sorgente, definiti come  $D_S = \{(x_{S_1}, y_{S_1}), \dots, (x_{S_n}, y_{S_n})\}$  tali che  $x_{S_i} \in \mathcal{X}_S$  è l’istanza del dato e  $y_{S_i} \in \mathcal{Y}_S$  è il corrispondente *label*. In maniera similare definiamo anche i dati del dominio target  $D_T = \{(x_{T_1}, y_{T_1}), \dots, (x_{T_n}, y_{T_n})\}$  tali che  $x_{T_i} \in \mathcal{X}_T$  è l’istanza del dato e  $y_{T_i} \in \mathcal{Y}_T$  è il corrispondente *label*.

Dire che due domini  $\mathcal{D}_S$  e  $\mathcal{D}_T$  implica che  $\mathcal{X}_S \neq \mathcal{X}_T$  oppure  $P_X(X) \neq$

$P_T(X)$ . In maniera del tutto analoga è possibile definire la differenza tra due *task*  $T_S$  e  $T_T$ . Nel caso in cui i due domini ed i due task sono uguali ci si riconduce ad un tradizionale problema di apprendimento. Quando invece c'è una relazione, implicita o esplicita, tra gli spazi delle *feature* dei due domini si dice che il dominio sorgente e target sono in relazione tra di loro.

**TIPOLOGIE DI TRANSFER LEARNING** Prima di introdurre le varie tipologie di *transfer learning* è necessario definire formalmente questo concetto e fare alcune premesse.

**Definizione 2.2.1.** (Transfer Learning) Dato un dominio sorgente  $\mathcal{D}_S$ , un task di apprendimento  $T_S$ , un dominio target  $\mathcal{D}_T$  ed un task di apprendimento  $T_T$ , il transfer learning tenta di migliorare l'apprendimento di  $f(\cdot)_T \in \mathcal{D}_T$  usando la conoscenza in  $\mathcal{D}_S$  e  $T_S$ , con  $\mathcal{D}_S \neq \mathcal{D}_T$  o  $T_S \neq T_T$ .

Quando si parla di *transfer learning* bisogna mettere in conto tre problemi: *cosa*, *come* e *quando* trasferire.

- Cosa trasferire: quale parte della conoscenza bisogna trasferire tra domini o task sorgenti e target. In particolare possiamo dire che alcune conoscenze sono comuni tra i diversi domini, mentre altre sono specifiche.
- Come trasferire: a questo problema pone una soluzione l'algoritmo di trasferimento della conoscenza.
- Quando trasferire: ci chiediamo in quali situazioni è realmente necessario applicare tecniche di *transfer learning* ed in quali non è assolutamente necessario. Ad esempio in casi in cui i due domini sorgente e target non sono in relazione tra di loro il *transfer learning* potrebbe non portare ad alcun risultato positivo. Si tende quindi sempre a parlare di *transfer learning* dando per scontato che i due domini siano in qualche modo relazionati tra di loro, in quanto altrimenti non avrebbe senso andare oltre l'apprendimento tradizionale.

Possiamo ora descrivere le tre categorie di *transfer learning*:

- *Transfer Learning Induttivo*: il task target è differente dal task sorgente e non importa se il dominio sorgente ed il dominio target sono uguali o meno. Per indurre un modello predittivo oggettivo, da usare nel dominio target, sono necessari alcuni dati etichettati nel

dominio sorgente. A seconda della tipologia ed alla quantità di annotazioni possiamo dividere questo tipo di *transfer learning* in ulteriori due sottocategorie.

- Molti dati annotati nel dominio sorgente: ci si riconduce al caso del *multitask learning*, tuttavia mentre lo scopo di quest'ultimo è operare bene in entrambi i domini, lo scopo del *transfer learning* induttivo è operare bene solamente sul dominio obiettivo.
- Nessun dato annotato nel dominio sorgente: le similarità in questo caso ci portano a pensare al *self-taught learning*, dove le annotazioni tra il dominio sorgente ed obiettivo sono totalmente differenti, e quindi non direttamente utilizzabili.
- *Transfer Learning Trasduttivo* (*si traduce così "transductive"*?): in questo caso il task obiettivo ed il task sorgente sono i medesimi, mentre i domini sono differenti. Abbiamo quindi molti dati annotati nel dominio sorgente e nessuna annotazione nel dominio obiettivo. Possiamo a sua volta dividere questa categoria in ulteriori due sottocategorie.
  - Gli spazi delle feature tra sorgente e obiettivo sono differenti, più formalmente abbiamo  $\mathcal{X}_S \neq \mathcal{X}_T$
  - Gli spazi delle feature tra sorgente e obiettivo sono gli stessi, ma cambia la distribuzione di probabilità marginale, quindi  $P(X_S) \neq P(X_T)$ . Quest'ultimo caso è chiamato anche *Domain Adaptation*.
- *Transfer Learning non supervisionato*: il task obiettivo è differente dal task sorgente, ma hanno una qualche tipo di relazione tra di loro. Tuttavia l'attenzione si focalizza sul risolvere compiti di apprendimento non supervisionati nel dominio obiettivo, come possono essere il *clustering*, *dimensionality reduction* o *density estimation*. Non abbiamo quindi annotazioni né nel dominio sorgente, né nel dominio obiettivo.

In Tabella 1 sono riassunte tutte le caratteristiche principali delle varie categorie di *transfer learning*. **SISTEMARE TABELLA 1**

### 2.2.2 Addestramento sulle immagini RGB

Per il primo esperimento è stato deciso di effettuare un training di *Retina-Net* partendo dai pesi della rete precedentemente addestrata sul dataset

gi a ivo tivo visionato	Similarità	Annotazioni Sorgente	Annotazioni Target	Campo di applicabilità
ivo	Multitask Learning	SI	SI	Regressione e Classificazione
	Self-taught Learning	NO	SI	Regressione e Classificazione
tivo	Domain adaptation	SI	NO	Regressione e Classificazione
visionato	46501	NO	NO	Clustering, dimensionality reduction

Tabella 1.: Schema riassuntivo delle categorie di Transfer Learning

di COCO. Il dataset utilizzato è KAIST MPD, descritto precedentemente in 2.1.1. Il motivo è legato al fatto che è l'unico dataset a nostra disposizione a disporre di annotazioni sulle immagini RGB.

Questa prima fase di addestramento è durata circa 40 ore, e come si può vedere dal grafico in Figura 10 è proceduta senza particolari problemi fino ad arrivare a convergenza intorno all'epoca 45. Le classi usate per l'addestramento sono solamente *person* e *cyclist*. È stato deciso di non prendere in considerazioni le rimanenti classi in quanto sono persone non ben distinguibili. Tutti gli esperimenti sono stati eseguiti su una macchina remota dotata di una GPU Nvidia Titan X.

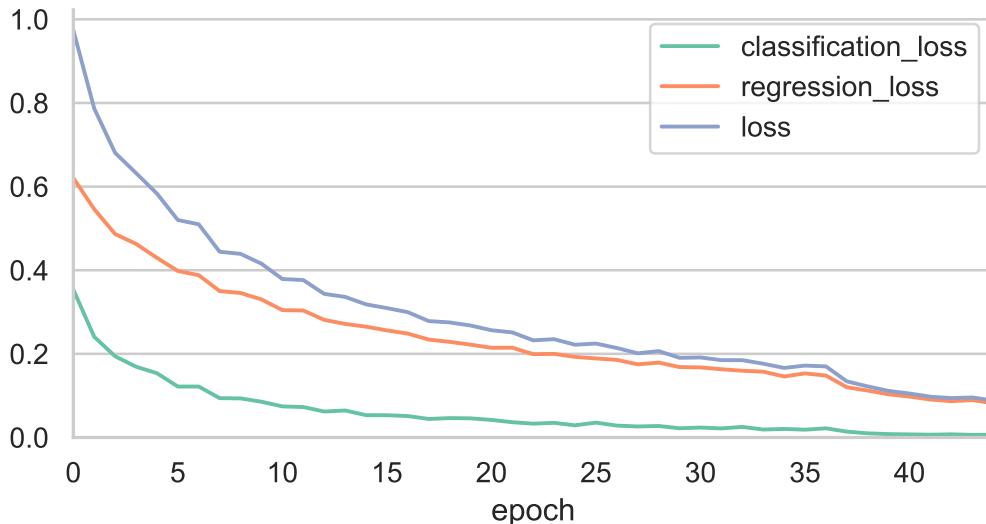


Figura 10.: Transfer learning da COCO a KAIST

Dopo la fase di addestramento sono stati eseguite le valutazioni sulla parte di dataset adibita ai test. Inizialmente le classi utilizzate per i test sono le stesse usate per l'addestramento. I risultati complessivi vengono riassunti in Tabella 2. La Mean Average Precision (mAP) mostrata

	# Istanze	mAP
Person	45195	0.4184
Cyclist	1396	0.1154
Complessivo	46501	0.4093

Tabella 2.: Test complessivo delle performance dopo l'addestramento

	# Istanze	mAP		# Istanze	mAP
Person	33688	0.4493	Person	11507	0.3281
Cyclist	818	0.2015	Cyclist	578	0.0029
Complessivo	34506	0.4434	Complessivo	12085	0.3125

(a) Giorno

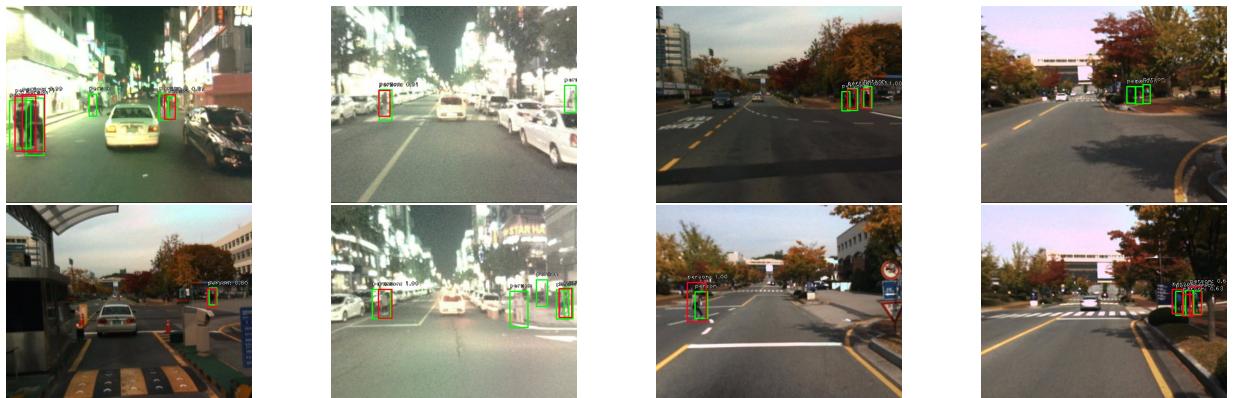
(b) Notte

Tabella 3.: Risultati della valutazione separata

nell'ultima riga della tabella è stata calcolata come media pesata secondo il numero di esempi all'interno del *test set*.

La valutazione è stata anche effettuata in maniera separata sia sulla parte di *test set* diurna che notturna. I risultati sono mostrati in Tabella 3a e Tabella 3b. Come lecito aspettarsi, avendo visibilità limitata in notturna si ottengono risultati mediamente peggiori. In Figura 11 è possibile vedere un esempio di predizioni fatte sul *test set*.

Il successivo step della valutazione è stato effettuato su più classi, per questa comparazione delle performance è stata presa in considerazio-

Figura 11.: Esempio di predizioni, in verde la *ground truth* in rosso le predizioni.

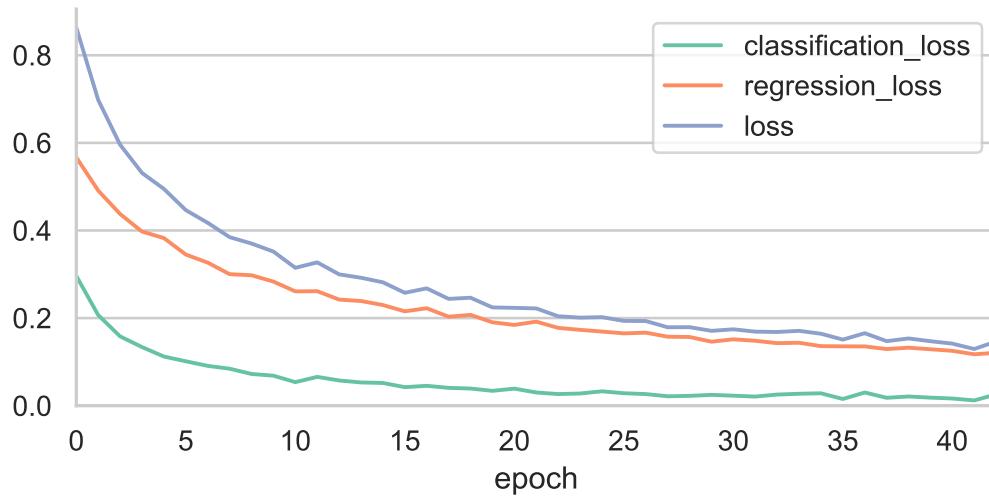


Figura 12.: Transfer learning da KAIST Visibile a KAIST Termico

ne anche la classe *people*, ma rinominandola in *person* in maniera tale da farla digerire **rivedere questo periodo** alla rete già addestrata. **INSERIRE VALUTAZIONE EFFETTUATA CON TUTTE LE CLASSI, APPENA LE GPU SI LIBERANO. I PESI SONO QUELLI DELLA RUN RUBY-YOGURT**

### 2.2.3 Passaggio alle immagini termiche su KAIST

Dopo la fase di addestramento descritta in Sezione 2.2.2 è stato effettuato il passaggio all’addestramento sulle immagini termiche del dataset di KAIST MPD. La base di partenza per questa fase di *training* sono i pesi derivanti dall’addestramento effettuato in Sezione 2.2.2, è quindi stata attuata una tecnica di *Transfer Learning* (2.2.1) in maniera da ridurre i tempi di addestramento. In questo caso, operando su due tipologie di immagini differenti cambia il dominio, più in particolare lo spazio delle feature. La distribuzione di probabilità sulle varie immagini resta la medesima in quanto il training è svolto sulle stesse immagini, come rimane invariato anche il task, ovvero il riconoscimento di pedoni e ciclisti. La fase di apprendimento è proceduta senza particolari intoppi, come si può vedere in Figura 12 le varie loss scendono in maniera stabile e controllata. In maniera del tutto similare a quanto si può vedere in Figura 10.

I risultati della valutazione sul test set sono mostrati in maniera complessiva in Tabella 4. Il test è stato effettuato usando i label *person*, *cyclist*

	# Istanze	mAP
Person	53443	0.3663
Cyclist	1396	0.0392
Complessivo	54839	0.3580

Tabella 4.: Test complessivo delle performance dopo l'addestramento

	# Istanze	mAP		# Istanze	mAP
Person	38802	0.3415	Person	14641	0.4440
Cyclist	818	0.2015	Cyclist	578	0.0029
Complessivo	39620	0.3353	Complessivo	15219	0.4288

(a) Giorno

(b) Notte

Tabella 5.: Risultati della valutazione separata

e *people*, quest'ultimi opportunamente rinominati in *person* per farli riconoscere correttamente a RetinaNet. La prima cosa che si nota è un calo delle performance rispetto al test sulle immagini visibili (Tabella 2). La spiegazione di questo calo delle performance si ottiene analizzando i risultati della valutazione separata effettuata sulla parte di dataset diurna e notturna, presente in Tabella 5. La prima cosa che si nota guardando la Tabella 5a è un drastico calo della mAP nella parte diurna, mentre in Tabella 5b abbiamo un leggero incremento. Il motivo è dovuto alla differenza di temperatura tra il pedone e quello che lo circonda. Di giorno, essendo la temperatura atmosferica più alta rispetto alla notte questa differenza di temperatura è conseguentemente più bassa, rendendo così il pedone, o chi per lui, più difficilmente riconoscibile. Di notte invece accade l'esatto contrario, la temperatura atmosferica è generalmente più bassa, quindi aumenta il delta di temperatura tra il pedone e l'ambiente circostante, rendendolo più facilmente riconoscibile. Considerazioni simili possono essere fatte sull'analisi svolta sulla parte di dataset RGB in quanto di notte essendoci poca luce si fatica di più a riconoscere un oggetto che non è illuminato di luce propria, di giorno invece accade l'esatto contrario. **aggiungere tabella con una colonna in più che rappresenta la differenza di performance tra le valutazioni**

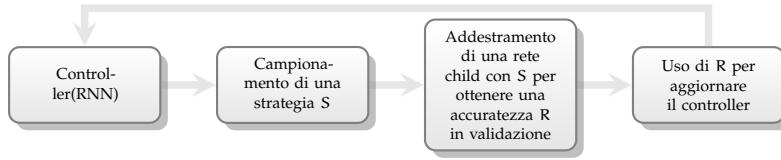


Figura 13.: Schema di funzionamento di AutoAugment

#### 2.2.4 ALTRA ROBA DA SCRIVERE

### 2.3 DATA AUGMENTATION

Nel Deep Learning avere un dataset ampio è un prerequisito fondamentale per far apprendere al proprio modello le caratteristiche che portano ad ottenere buoni risultati ed evitare il fenomeno dell’overfitting. Nel caso di **SEZIONE DA SCRIVERE SU ANNOTAZIONI MANUALI** per ovvie questioni di tempo e praticità le annotazioni effettuate sul dataset sono molte poche, rendendolo così appena sufficiente ad i nostri scopi. Per ovviare a questo problema esiste la *Data Augmentation*. Con questo termine si fa riferimento a tutte quelle tecniche che permettono di creare nuovi dati tramite manipolazioni dei dati originali. Ad esempio, restando nel dominio delle immagini, la più semplice tecnica di *Data Augmentation* è applicare casualmente alle immagini in input trasformazioni come possono essere la rotazione o l’inversione secondo un asse. Recentemente l’interesse è virato più verso una *Data Augmentation* mirata al dataset su cui verrà applicata, ovvero con una serie di regole che comprendono il tipo di trasformazione da applicare, l’intensità e la probabilità con cui verrà applicata. Sono necessarie quindi nuove fasi di addestramento per apprendere queste regole che a tutti gli effetti possono essere considerate alla stessa stregua di iperparametri del nostro modello.

#### 2.3.1 Auto Augment

Lo scopo di *AutoAugment* [3] è automatizzare il processo che porta ad ottenere delle regole di *Data Augmentation* per un determinato dataset obiettivo. La ricerca delle migliori policy per il dataset viene trattato alla pari di un problema di ricerca discreta, in Figura 13 è possibile vedere uno schema riassuntivo della struttura di *AutoAugment*. I componenti principali di *AutoAugment* sono due: un algoritmo di ricerca ed uno spazio di ricerca. L’algoritmo di ricerca, implementato tramite una RNN campiona una policy S. Questa policy al suo interno contiene le informa-

	Original	Sub-policy 1	Sub-policy 2	Sub-policy 3	Sub-policy 4	Sub-policy 5
Batch 1						
Batch 2						
Batch 3						
	ShearX, 0.9, 7 Invert, 0.2, 3	ShearY, 0.7, 6 Solarize, 0.4, 8	ShearX, 0.9, 4 AutoContrast, 0.8, 3	Invert, 0.9, 3 Equalize, 0.6, 3	ShearY, 0.8, 5 AutoContrast, 0.7, 3	

Figura 14.: Esempio di policy con 5 sub-policy, immagine tratta da [3]

zioni riguardanti quale trasformazione applicare, con quale probabilità applicarla, e con quale potenza. Un esempio di policy contenuta all'interno dello spazio di ricerca è possibile vederla in Figura 14. La prima subpolicy definisce una specifica sequenza di operazioni, la prima è *Shear X* con probabilità di essere applicata pari a 0.9 e potenza di 7/10. La seconda operazione che verrà effettuata è *Invert* con probabilità 0.8. In totale le operazioni nello spazio di ricerca sono 16, ed ognuna di esse ha un intervallo di potenza discretizzato con cui può essere applicata che varia da 1 a 10. In maniera del tutto similare vengono discretizzati anche i valori di probabilità in un intervallo di 11. Trovare quindi una subpolicy è un problema di ricerca in uno spazio di dimensione  $(16 \times 10 \times 11)^2$ . Considerando che una policy contiene 5 subpolicy lo spazio di ricerca arriva a  $(16 \times 10 \times 11)^{10}$ , che è nell'ordine di grandezza di  $10^{32}$ .

L'algoritmo di ricerca prende spunto dalle tecniche di apprendimento per rinforzo. Le sue componenti sono due, il controller RNN e l'algoritmo di addestramento, che in questo caso è il *Proximal Policy Optimization* [69]. L'addestramento del controller RNN è realizzato con un segnale di ricompensa che rappresenta il miglioramento in termini di generalizzazione della rete child addestrata con la policy S. Nel caso di Figura 13 la ricompensa è R, ovvero il risultato della valutazione sul validation set. Alla fine di questa fase di ricerca vengono trovate le migliori 5 policy, che vengono concatenate in un'unica grande policy contentente 25 subpolicy.

**AUTOAUGMENT PER OBJECT DETECTION** Fino ad ora abbiamo parlato di *AutoAugment* in termini di classificazione di una immagine, lo scopo della tesi è però realizzare *Object Detection*. Gli stessi autori di [3] in [70]

hanno evoluto questa tecnica per far sì che operi anche su questo tipo di compiti.

Come in precedenza il problema viene trattato come una ricerca in uno spazio discreto. Sempre come prima vengono definite policy come insiemi di  $K$  subpolicy, con queste ultime composte da  $N$  operazioni da applicare all'immagine in maniera sequenziale. In maniera del tutto analoga ogni operazione ha una probabilità con cui sarà applicata, a differenza della classificazione è stata resa più granulare la scelta dei possibili valori discretizzati, chiamandoli  $M$  per la potenza e  $L$  per la probabilità. Una importante differenza invece riguarda le operazioni che è possibile applicare all'immagine, in quanto in aggiunta alle classiche operazioni di trasformazione dell'intero frame si aggiungono ulteriori 6 operazioni applicabili solamente a livello di BB, raggiungendo così un totale di 22 operazioni. La dimensione dello spazio di ricerca di una subpolicy raggiunge quindi le dimensioni di  $(22 \times L \times M)^2$ , e più in particolare lo spazio di ricerca di una policy arriva a  $(22 \times L \times M)^{10}$ .

Gli autori in esperimenti preliminari hanno trovato che porre  $L = M = 6$  è un buon *trade-off* tra performance e costi computazionali, in quanto si arriva ad uno spazio di ricerca nell'ordine di  $10^{28}$ . Purtroppo però rimane comunque computazionalmente molto oneroso in quanto con 400 Tensor Processing Unit (TPU) sono state impiegate circa 48 ore.

Tuttavia sono state rilasciate delle policy apprese da dataset differenti con cui sono stati eseguiti esperimenti che verranno successivamente descritti.

### 2.3.2 Rand Augment

Il più grande svantaggio di *AutoAugment* è l'elevato costo computazionale per una fase di ricerca separata su un task proxy. *RandAugment* [71] è un nuovo modo di realizzare *Data Augmentation* proposto dagli stessi autori di [3] e [70] che risolve questa problematica. La fase di ricerca di *AutoAugment* porta ad avere, nella sua versione orientata alla classificazione di immagini, più di 30 parametri, motivo per cui il lavoro qui descritto mira a ridurne drasticamente il numero fino addirittura a portarli a due soli. La selezione delle policy di augmentation viene infatti realizzata in maniera totalmente casuale, ciò vuol dire che se abbiamo  $K$  operazioni possibili la probabilità di selezionarne una è di  $\frac{1}{K}$ . Inoltre applicando  $N$  operazioni ad un'immagine si possono esprimere tutte le possibili policy come le  $N$  disposizioni con ripetizione di  $K$  operazioni, quindi  $K^N$ .

L'ultima variabile da prendere in considerazione è la forza con cui

vengono applicate queste operazioni. Viene ripresa la stessa scala usata in [3], ovvero un intero che varia da 0 a 10. Al fine di ridurre il numero dei parametri gli autori sono intervenuti anche in questo definendo un singolo valore  $M$  per parametrizzare la potenza con cui vengono applicate le operazioni.

Riassumendo quindi gli iperparametri sono due: il numero  $N$  di operazioni da applicare e la forza  $M$  con cui verranno applicate. Essendo così pochi gli iperparametri può essere usato un classico algoritmo di *GridSearch* per ottimizzarli. Nel caso di questa tesi è stato usato un ottimizzatore bayesiano presente sulla piattaforma Comet.ml di cui non vengono però rilasciati dettagli tecnici o implementativi.

Risulta quindi un algoritmo molto semplificato rispetto a quello descritto in Sezione 2.3.1. Come è possibile vedere in Codice 2.1 sono solo poche righe di codice in cui vengono selezionate casualmente  $N$  operazioni tra quelle disponibili e vengono applicate successivamente con una forza pari a  $M$ .

Listing 2.1: Algoritmo di RandAugment in Python [71]

```
transforms = [
    'Identity', 'AutoContrast', 'Equalize',
    'Rotate', 'Solarize', 'Color', 'Posterize',
    'Contrast', 'Brightness', 'Sharpness',
    'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugment(N, M):
    """Generate a set of distortions.

    Args:
        N: Number of augmentation transformations to apply
            sequentially.
        M: Magnitude for all the transformations.
    """

    sampled_ops = np.random.choice(transforms, N)
    return [(op, M) for op in sampled_ops]
```

Un altro svantaggio di *AutoAugment* è che le policy vengono apprese su un sottoinsieme che può non rispecchiare le caratteristiche dell'intero dataset. In *RandAugment* essendo la fase di ottimizzazione molto più semplificata è possibile operare direttamente sull'intero dataset, portando così ad ottenere risultati potenzialmente migliori.



# 3

---

## ESPERIMENTI

---

3.1 ORGANIZZAZIONE DEI DATASET

3.2 ESPERIMENTI INIZIALI

3.3 DATA AUGMENTATION



# 4

---

## CONCLUSIONI

---



# A

---

## CODICE DI IOU OVER TIME

---

Listing A.1: Algoritmo di IoU over time in Python

```
def compute_intersection_over_union(box1, box2):
    x1 = max(box1[0], box2[0])
    y1 = max(box1[1], box2[1])
    x2 = min(box1[2], box2[2])
    y2 = min(box1[3], box2[3])
    intersection_area = max(0, x2 - x1 + 1) * max(0, y2 - y1 + 1)
    box_1_area = (box1[2] - box1[0] + 1) * (box1[3] - box1[1] + 1)
    box_2_area = (box2[2] - box2[0] + 1) * (box2[3] - box2[1] + 1)
    intersection_over_union = intersection_area / float(box_1_area + box_2_area - intersection_area)
    return intersection_over_union

def calc_bbox_size(bbox):
    """
    Get the size of bbox in input
    # Arguments
        bbox      : coordinates of bbox (x_min, y_min, x_max, y_max).
    # Returns
        size of bbox
    """
    return (bbox[2]-bbox[0])*(bbox[3]-bbox[1])

def iou_evaluation(detections, threshold):
    index_max_detections = np.argmax(list(map(lambda x: len(x), detections)))
    id_tree = 0
    idx = index.Index(interleaved=True)
    iou_values = [0]*len(detections[index_max_detections])
    for_debug = np.concatenate(np.delete(detections, index_max_detections, axis = 0))
    for detection in for_debug:
        idx.insert(id_tree, detection[0:4], obj = detection[4:6])
        id_tree = id_tree + 1
    for i in range(len(detections[index_max_detections])):
        intersection_bbox = idx.intersection(detections[index_max_detections][i][0:4], objects=True)
        num_elem = 0
        for item in intersection_bbox:
            num_elem = i+1
            iou_values[i] += compute_intersection_over_union(item.bbox, detections[index_max_detections][i][0:4])
        if num_elem is not 0:
            iou_values[i] /= num_elem
    iou_values = np.divide(iou_values, len(detections)-1)
    bbox = []
    labels = []
    scores = []
    for det, iou in zip(detections[index_max_detections], iou_values):
        if iou > threshold:
            intersection_bbox = idx.intersection(det[0:4], objects=True)
            gen = list(intersection_bbox)
            max_bbox_index = np.argmax(list(map(lambda x: calc_bbox_size(x), list(map(lambda x: x.bbox, gen)))))
            bbox.append(gen[max_bbox_index].bbox)
            scores.append(max(list(map(lambda x: x.object[0], gen))))
            labels.append(gen[max_bbox_index].object[1])
    return np.array(bbox), np.array(scores), np.array(labels, dtype=int)
```



# B

---

IMPLEMENTAZIONE DI RANDAUGMENT SU  
RETINANET

---



---

## ACRONIMI

---

**mAP** Mean Average Precision

**KAIST MPD** KAIST Multispectral Pedestrian Dataset

**GPU** Graphics Processing Unit

**KAIST** Korea Advanced Institute of Science and Technology

**FOV** Field of View

**BB** Bounding Box

**RNN** Recurrent Neural Network

**TPU** Tensor Processing Unit

**CNN** Convolutional Neural Network

**FCN** Fully Convolutional Network

**RNN** Recurrent Neural Network

**NMS** Non-Maximum Suppression

**HNM** Hard Negative Mining

**YOLO** You Only Look Once

**VOC** Pascal Visual Object Classes

**ILSVRC** ImageNet Large Scale Visual Recognition Challenge

**MS-COCO** Microsoft - Common Object in COntext

**OID** Open Images Detection



---

## BIBLIOGRAFIA

---

- [1] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *CoRR*, abs/1905.05055, 2019. (Cited on pages 5, 9, 10, 12, 13, 15, 16, and 17.)
- [2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. (Cited on pages 5, 22, and 23.)
- [3] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. (Cited on pages 5, 40, 41, 42, and 43.)
- [4] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *CoRR*, abs/1907.09408, 2019. (Cited on page 9.)
- [5] Irving Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987. (Cited on page 10.)
- [6] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008. (Cited on page 10.)
- [7] Martin A Fischler and Robert A Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on computers*, (1):67–92, 1973. (Cited on page 10.)
- [8] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International journal of computer vision*, 77(1-3):259–289, 2008. (Cited on page 10.)

- [9] Dariu M Gavrila and Vasanth Philomin. Real-time object detection for "smart" vehicles. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 87–93. IEEE, 1999. (Cited on page 10.)
- [10] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):509–522, 2002. (Cited on page 10.)
- [11] Bo Wu and Ramakant Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 90–97. IEEE, 2005. (Cited on page 10.)
- [12] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991. (Cited on page 10.)
- [13] Alex Pentland, Baback Moghaddam, Thad Starner, et al. View-based and modular eigenspaces for face recognition. 1994. (Cited on page 10.)
- [14] Régis Vaillant, Christophe Monrocq, and Yann Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250, 1994. (Cited on page 11.)
- [15] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (Cited on page 11.)
- [16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. (Cited on page 11.)
- [17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. (Cited on page 11.)

- [18] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009. (Cited on pages 11 and 15.)
- [19] Tomasz Malisiewicz, Abhinav Gupta, and Alexei Efros. Ensemble of exemplar-svms for object detection and beyond. 2011. (Cited on page 11.)
- [20] Tomasz Malisiewicz. *Exemplar-based representations for object detection, association and beyond*. Carnegie Mellon University, 2011. (Cited on page 11.)
- [21] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 73–80. IEEE, 2010. (Cited on page 12.)
- [22] Jan Hosang, Rodrigo Benenson, Piotr Dollár, and Bernt Schiele. What makes for effective detection proposals? *IEEE transactions on pattern analysis and machine intelligence*, 38(4):814–830, 2015. (Cited on page 12.)
- [23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. (Cited on pages 12 and 15.)
- [24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. (Cited on pages 13 and 14.)
- [25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. (Cited on page 13.)
- [26] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. (Cited on page 13.)
- [27] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object

- detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. (Cited on page 13.)
- [28] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4203–4212, 2018. (Cited on page 13.)
- [29] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *european conference on computer vision*, pages 354–370. Springer, 2016. (Cited on page 13.)
- [30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sep. 2010. (Cited on page 14.)
- [31] Antonio Torralba and Pawan Sinha. Detecting faces in impoverished images. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB, 2001. (Cited on page 14.)
- [32] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. 2005. (Cited on pages 14 and 18.)
- [33] Santosh K Divvala, Derek Hoiem, James H Hays, Alexei A Efros, and Martial Hebert. An empirical study of context in object detection. In *2009 IEEE Conference on computer vision and Pattern Recognition*, pages 1271–1278. IEEE, 2009. (Cited on page 15.)
- [34] Zeming Li, Yilun Chen, Gang Yu, and Yangdong Deng. R-fcn++: Towards accurate region-based fully convolutional networks for object detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. (Cited on page 15.)
- [35] Sean Bell, C Lawrence Zitnick, Kavita Bala, and Ross Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2874–2883, 2016. (Cited on page 15.)

- [36] Jianan Li, Yunchao Wei, Xiaodan Liang, Jian Dong, Tingfa Xu, Jia-shi Feng, and Shuicheng Yan. Attentive contexts for object detection. *IEEE Transactions on Multimedia*, 19(5):944–954, 2016. (Cited on page 15.)
- [37] Chaitanya Desai, Deva Ramanan, and Charless C Fowlkes. Discriminative models for multi-class object layout. *International journal of computer vision*, 95(1):1–12, 2011. (Cited on pages 15 and 17.)
- [38] Zheng Song, Qiang Chen, Zhongyang Huang, Yang Hua, and Shui-cheng Yan. Contextualizing object detection and classification. In *CVPR 2011*, pages 1585–1592. IEEE, 2011. (Cited on page 15.)
- [39] Xinlei Chen and Abhinav Gupta. Spatial memory for context reasoning in object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4086–4096, 2017. (Cited on page 15.)
- [40] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018. (Cited on page 15.)
- [41] Saurabh Gupta, Bharath Hariharan, and Jitendra Malik. Exploring person context and local scene context for object detection. *arXiv preprint arXiv:1511.08177*, 2015. (Cited on page 15.)
- [42] Yong Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Structure inference net: Object detection using scene-level context and instance-level relationships. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6985–6994, 2018. (Cited on page 15.)
- [43] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3, 2001. (Cited on pages 16 and 17.)
- [44] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. (Cited on page 16.)

- [45] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian conference on computer vision*, pages 290–306. Springer, 2014. (Cited on page 16.)
- [46] Damian Mrowca, Marcus Rohrbach, Judy Hoffman, Ronghang Hu, Kate Saenko, and Trevor Darrell. Spatial semantic regularisation for large scale object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2003–2011, 2015. (Cited on page 16.)
- [47] Li Wan, David Eigen, and Rob Fergus. End-to-end integration of a convolution network, deformable parts model and non-maximum suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 851–859, 2015. (Cited on page 17.)
- [48] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4507–4515, 2017. (Cited on page 17.)
- [49] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision*, pages 198–213. Springer, 2016. (Cited on page 17.)
- [50] Constantine P Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 555–562. IEEE, 1998. (Cited on page 17.)
- [51] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. In *Advances in Neural Information Processing Systems*, pages 875–881, 1996. (Cited on page 17.)
- [52] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. (Cited on page 18.)
- [53] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International journal of computer vision*, 38(1):15–33, 2000. (Cited on page 18.)

- [54] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. (Cited on page 18.)
- [55] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015. (Cited on page 18.)
- [56] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. 2009. (Cited on page 19.)
- [57] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. (Cited on page 20.)
- [58] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. (Cited on page 20.)
- [59] Shanshan Zhang, Rodrigo Benenson, and Bernt Schiele. Citypersons: A diverse dataset for pedestrian detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3221, 2017. (Cited on page 21.)
- [60] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. (Cited on page 21.)
- [61] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, et al. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github. com/openimages*, 2:3, 2017. (Cited on page 22.)
- [62] Markus Braun, Sebastian Krebs, Fabian Flohr, and Dariu M Gavrila. The eurocity persons dataset: A novel benchmark for object detection. *arXiv preprint arXiv:1805.07193*, 2018. (Cited on page 24.)

- [63] Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi, and In So Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1037–1045. IEEE Computer Society, 2015. (Cited on page 27.)
- [64] Free flir thermal dataset for algorithm training. <https://www.flir.com/oem/adas/adas-dataset-form/>. Accessed: 2019-11-20. (Cited on page 27.)
- [65] L. Bienkowski, C. Homma, K. Eisler, and Christian Boller. Hybrid camera and real-view thermography for nondestructive evaluation. 01 2012. (Cited on page 28.)
- [66] Piotr Dollár. Piotr’s Computer Vision Matlab Toolbox (PMT). <https://github.com/pdollar/toolbox>. (Cited on page 29.)
- [67] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>. (Cited on page 32.)
- [68] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010. (Cited on page 32.)
- [69] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. (Cited on page 41.)
- [70] Barret Zoph, Ekin D. Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V. Le. Learning data augmentation strategies for object detection. *CoRR*, abs/1906.11172, 2019. (Cited on pages 41 and 42.)
- [71] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical data augmentation with no separate search. *CoRR*, abs/1909.13719, 2019. (Cited on pages 42 and 43.)