



UNIVERSITÀ DEGLI STUDI DI FIRENZE
SCUOLA DI SCIENZE MATEMATICHE FISICHE E NATURALI
DIPARTIMENTO DI INFORMATICA

Tesi di laurea Magistrale in Informatica - Data Science

RICONOSCIMENTO DI AZIONI UMANE USANDO TECNICHE DI APPRENDIMENTO PROFONDO PER LA STIMA DELLA POSA

Candidato
Andrea Moscatelli

Relatore
Marco Bertini

Correlatore
Correlatore 1

ANNO ACCADEMICO 2018 - 2019

Indice

Prefazione	iii
Introduzione	iv
1 Come imparano le macchine?	1
1.1 Deep learning	3
1.2 Le reti neurali	3
1.3 Le reti neurali ricorrenti	3
1.4 Le LSTM	3
2 Stima della posa	4
3 PoseNet	7
3.1 Stima dei key-points	8
3.2 Raggruppamento dei key-points in istanze di persona	10
4 Detectron2	12
4.1 Mask R-CNN	15
4.1.1 Predizione della posa con Mask R-CNN	16
5 Il dataset NTU RGB+D	18
5.1 NTU RGB+D in dettaglio	20
5.2 Criteri di valutazione	23
5.2.1 Valutazione Cross-subject	23
5.2.2 Valutazione Cross-view	23

6 Classificazione	24
6.1 Struttura della rete	24
6.2 Tecniche	24
6.2.1 Semplice	24
6.2.2 Tecnica dei centri	24
6.2.3 Tecnica delle differenze	24
7 Risultati ottenuti	25
8 Conclusioni	26
9 Sviluppi futuri	27
Bibliografia	28

Prefazione

prefazione

Introduzione

Introduzione

Capitolo 1

Come imparano le macchine?

Quando parliamo di *machine learning*, ovvero *l'apprendimento delle macchine*, viene spontaneo domandarsi “Come impara una macchina?”.

L'apprendimento è di per sé un processo iterativo che permette di mutare le conoscenze a seconda delle informazioni che vengono raccolte. Nel machine learning un computer impara, ad esempio, un task di classificazione direttamente da immagini, testi e suoni grazie ad un *modello* e può raggiungere prestazioni allo stato dell'arte superando talvolta persino l'accuratezza umana. Il modello rappresenta cioè lo scopo dell'analisi, ossia *come* si vuole che il nostro computer impari l'algoritmo.

I modelli possono essere allenati seguendo 4 tipologie diverse di apprendimento:

- **Apprendimento supervisionato (Supervised Learning)** - In questo primo caso, il processo di apprendimento può essere pensato come un'insegnante che supervisiona il nostro algoritmo e che interromperà l'apprendimento solo quando l'algoritmo raggiungerà un livello accettabile del compito da imparare.

L'apprendimento supervisionato lo si ha quando nel nostro *dataset* di allenamento sono a disposizione sia i dati di input X che quelli di output Y e vogliamo insegnare al nostro algoritmo quella funzione f tale che

$$Y = f(x).$$

L'obiettivo è quello di approssimare sufficientemente bene la funzione f di modo da poter prevedere correttamente il valore Y_i per un futuro X_i non compreso nel nostro dataset di allenamento.

I problemi di apprendimento supervisionato si possono dividere in problemi di:

1. *Classificazione*, ovvero insegnare ad una macchina a categorizzare i dati in input. In questo scenario, nel nostro dataset di allenamento ad ogni dato di input sarà associata un'etichetta che ne indica la categoria appartenente. Più grande sarà il dataset e maggiore sarà l'informazione a disposizione dell'algoritmo per imparare.
2. *Regressione* - ovvero insegnare ad una macchina a predire il valore di ciò che sta analizzando partendo dai dati in input. A differenza della classificazione, in questo caso il risultato in output sarà un valore continuo e non un valore categorico. Ad esempio, dati in input le ore di studio di uno studente per la preparazione di un esame e le rispettive ore di sonno, prevedere la probabilità di superamento dell'esame in questione, oppure date in input la superficie e la posizione di un appartamento, prevederne il valore di mercato.

- **Apprendimento non supervisionato (Unsupervised Learning)** - in questo tipo di apprendimento, al contrario di quello supervisionato, non viene fornita nessuna etichetta di output Y per i nostri dati X e quello che ci si pone quindi come obiettivo è trovare delle relazioni tra i dati analizzati. In questo caso non c'è nessun "insegnante" a guidare l'apprendimento e non ci sono risposte corrette o sbagliate, l'algoritmo deve cercare di scoprire "da solo" se ci sono delle strutture decifrabili nei dati.

I problemi di apprendimento non supervisionato si possono dividere in:

1. *Raggruppamento* - detto anche *clustering*, si utilizzano quando è necessario raggruppare i dati che presentano caratteristiche simili. In questo caso l'algoritmo non fa uso di dati categorizzati, come visto in precedenza, ma estrae una regola di raggruppamento secondo caratteristiche che ricava dai dati stessi.

2. *Associazione* - strettamente legata al *Data mining*, questa classe di problemi è utile in tutti i casi per i quali siamo interessati a scoprire regole induttive nei dati analizzati, ad esempio la tendenza nei consumatori di un certo supermercato ad acquistare il prodotto *A* dopo aver acquistato il prodotto *B*. Ci si pone quindi l'obbiettivo di scovare schemi frequenti, associazioni, correlazioni o strutture casuali fra gli *item* di un database relazionale e si cerca quindi di scoprire le regole che predicono l'evento di un certo item in base agli eventi degli item ad esso legati.

- **Apprendimento parzialmente supervisionato (Semi-Supervised Learning)** - <https://lorenzogovoni.com/machine-learning-e-funzionamento/>
- **Apprendimento con rinforzo (Reinforcement Learning)** -

1.1 Deep learning

Il *deep learning* (o *apprendimento profondo* in italiano) è una tecnica di machine learning ed “insegna” ai computer una cosa che risulta estremamente naturale al cervello umano, ovvero *imparare per esempi*. Il deep learning è la tecnologia chiave grazie alla quale abbiamo automobili che si guidano da sole, riconoscimento e controllo vocale dei device, traduzioni automatiche, colorazione automatica di vecchi filmati in bianco e nero e molte altre cose ritenute impossibili solo fino a pochi anni fa.

<https://www.mathworks.com/discovery/deep-learning.html>

1.2 Le reti neurali

1.3 Le reti neurali ricorrenti

1.4 Le LSTM

Capitolo 2

Stima della posa

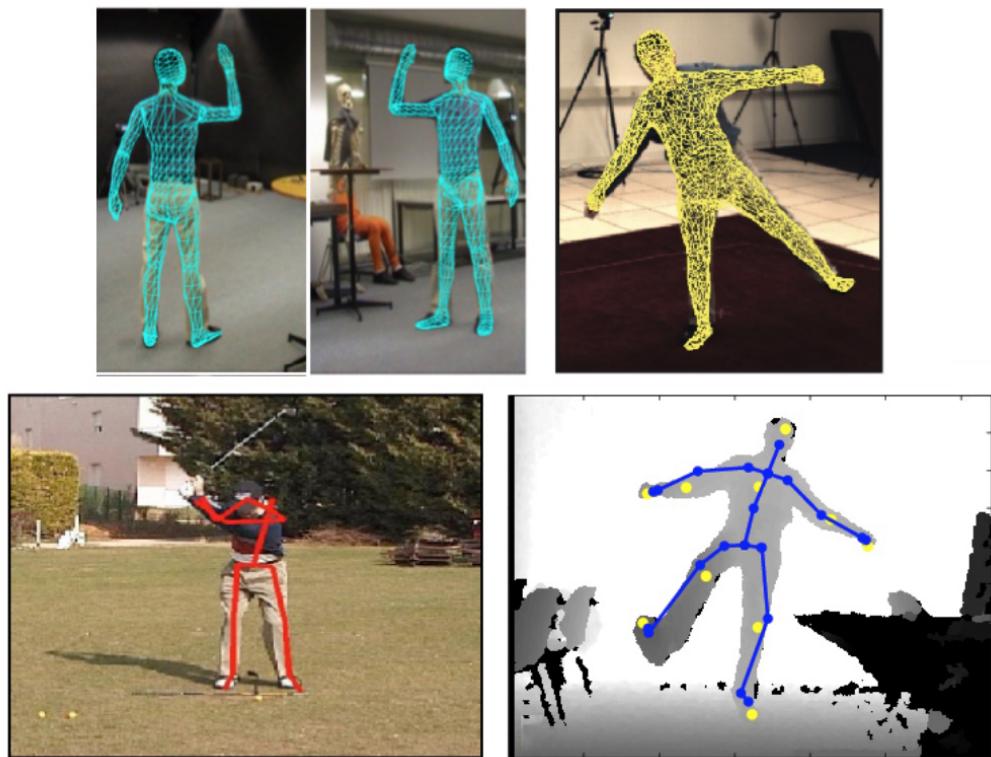


Figura 2.1. Esempi di stima della posa. In alto tre esempi di stima della posa utilizzando modelli di tipo volumetrico. In basso due esempi di stima della posa ottenuti utilizzando modelli di tipo scheletrici.

Cos è la stima della posa?

Quando parliamo di *stima della posa* ci riferiamo ad una tecnica di *computer vision* dedicata al riconoscimento di figure umane all'interno di video ed immagini, così da poter localizzare ad esempio dove, all'interno dell'immagine, si trova la testa, il braccio, la gamba destra, etc.. della persona inquadrata.

Questa tecnica non va assolutamente confusa con tecniche di riconoscimento di persone, infatti la stima della posa è in grado solo di riconoscere dove le parti del corpo di un individuo sono situate all'interno dell'immagine, non *chi* è inquadrato.

I campi di applicazione della stima della posa sono i più svariati: software interattivi che reagiscono al movimento della persona, robotica, realtà aumentata, animazione, fotoritocco intelligente, fitness, riabilitazione, etc. Stiamo parlando di un problema tutt'altro che semplice, infatti la condizione di luce dell'immagine, la variabilità dell'ambiente circostante, l'inclinazione del soggetto inquadrato, rendono il riconoscimento della posa un problema non affatto banale.

Spinti dal crescente interesse, negli ultimi anni sono stati sviluppati diversi algoritmi per la stima della posa, raggiungendo in molti casi risultati davvero sorprendenti con un'accuratezza prossima alla perfezione.



Figura 2.2. Un esempio di utilizzo in campo medico della stima della posa

La maggior parte dei software in circolazione in grado di stimare in maniera sufficientemente corretta la posa di un individuo non sono liberamente accessibili.

Due fra i migliori algoritmi (ad oggi) di *pose detection* sono sicuramente *Posenet* [1] e *Detectron 2* [4], dei quali ci occuperemo in maniera più approfondita nei capitoli seguenti.

Capitolo 3

PoseNet

I recenti progressi nel campo della visione artificiale hanno permesso alla comunità scientifica di spostarsi verso problemi ancora più articolati rispetto a quelli classici, con l'obiettivo di riconoscere figure umane in contesti non vincolati e molto variabili.

L'algoritmo *PoseNet* è stato ideato proprio con lo scopo di identificare una o più figure umane in qualsiasi contesto, anche in quelli “affollati”, ed essere in grado di identificare ogni persona stimandone i suoi *punti chiave* (o *key-points*).

Generalmente esistono due approcci principali per affrontare il problema del rilevamento di più persone in un'immagine, la stima della loro posa e la loro *segmentazione* (ovvero l'identificazione dei pixel che rappresentano ogni persona):

- l'approccio *top-down* inizia identificando e localizzando approssimativamente le singole istanze di persona delimitando il riquadro dell'immagine dentro il quale sono contenute, seguito da una fase di stima della posa o di separazione “primo piano-sfondo” nell'area identificata.
- Al contrario, l'approccio *bottom-up* inizia localizzando *entità semantiche individuali*, come ad esempio gambe, braccia, mani, etc, seguito dal loro raggruppamento in istanze di persone complete. PoseNet adotta questo secondo approccio.

Inoltre PoseNet utilizza una rete neurale convoluzionale per la quale il costo computazionale del riconoscimento delle pose è essenzialmente indipendente dal

numero di persone raffigurate nella scena ma dipende esclusivamente dalla scelta delle features della rete.

L'approccio adottato in PoseNet è quello di identificare dapprima tutti i punti chiave e successivamente raggrupparli in istanze di persona utilizzando un processo "greedy", ovvero partendo dal rilevamento "più sicuro" (e non come spesso accade, da un punto fisso di riferimento, ad esempio il naso). Anche se potrebbe sembrare una tecnica più "disordinata" i risultati empirici hanno dimostrato funzionare estremamente bene.

Oltre a stimare punti chiave sparsi, PoseNet stima anche delle maschere di segmentazione per ogni persona. Per fare ciò, viene allenata una seconda rete neurale con la quale viene associato ad ogni pixel x_i la probabilità di appartenenza di quel pixel ad ogni candidato j identificato. Se la probabilità è sufficientemente alta allora viene associato il pixel x_i al candidato j .

Questo algoritmo è stato allenato utilizzando il famoso dataset COCO [2] (versione 2016) che, fra molte altre cose, contiene anche l'annotazione dei 17 punti chiave (12 del corpo e 5 del volto) di migliaia di persone ed è riuscito a migliorarne l'*AP* (average-precision) da 0,655 a 0,687 diventandone il miglior risultato.

Essendo molto semplice questo metodo è anche quindi molto rapido, poiché non richiede alcuna fase supplementare di raffinamento dei risultati con tecniche di tipo *box-based* o *clustering*, rendendo di fatto PoseNet uno degli algoritmi più facilmente installabili su piccoli dispositivi, come ad esempio i cellulari.

Ma vediamo adesso più nel dettaglio come PoseNet stima e raggruppa i punti chiavi di una o più persone raffigurate in un'immagine.

3.1 Stima dei key-points

L'obiettivo di questa fase è quello di rilevare, in modo indipendente dall'istanza, tutti i key-points visibili appartenenti a qualsiasi persona dell'immagine. A tale scopo vengono prodotte delle *heat-maps*, ovvero dei canali della rete neurale dedicati al riconoscimento di particolari caratteristiche dell'immagine (una canale per ogni key-point) e degli *offset* (due canali per ogni key-point per gli spostamenti in orizzontale e verticale). Sia x_i la posizione 2-D nell'immagine, dove $i = 1, \dots, N$

e N è il numero di pixels; $D_R(y) = \{x : \|x - y\| \leq R\}$ un disco di raggio R centrato in y e $y_{j,k}$ la posizione 2-D del k -esimo key-point della j -esima istanza di persona, con $j = 1, \dots, M$, dove M è il numero di istanze nell'immagine.

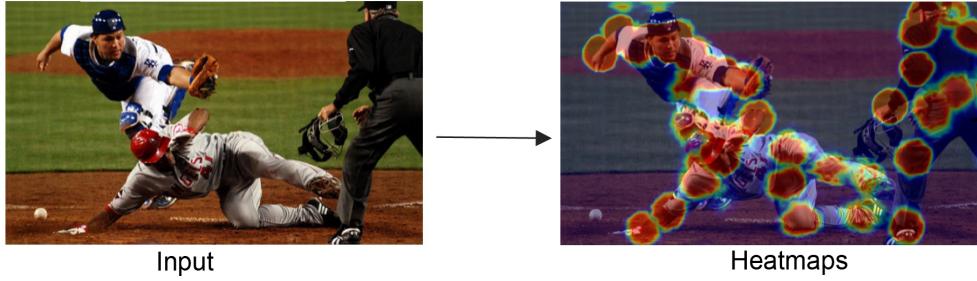


Figura 3.1. Generazione con PoseNet delle heat-maps per ogni tipologia di key-point

Per ogni tipo di key-point $k = 1, \dots, K$, viene impostato un task di classificazione binaria come segue. Viene generata una heat-map $p_k(x)$ tale che $p_k(x) = 1$ se $x \in D_R(y_{j,k})$ per qualsiasi istanza j , altrimenti $p_k(x) = 0$. Abbiamo quindi K tasks di classificazione binaria indipendenti (uno per ogni tipo di key-point) e ciascuno equivale a prevedere un disco di raggio R attorno a un tipo di key-point specifico appartenente a qualsiasi persona nell'immagine.

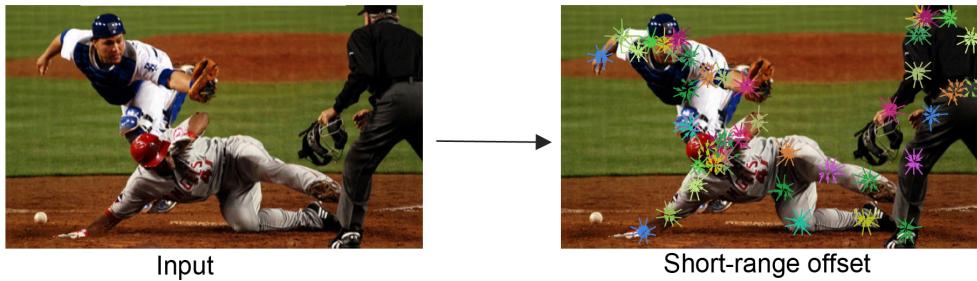


Figura 3.2. Esempio di stima degli offset a corto raggio con PoseNet

Oltre alle heat-maps, vengono anche usati vettori di *offset a corto raggio* $S_k(x)$ il cui scopo è quello di migliorare l'accuratezza della localizzazione dei key-points. Per ogni punto x all'interno dei dischi ricavati al passo precedente, il vettore di offset 2-D a corto raggio $S_k(x) = y_{j,k} - x$ rappresenta la distanza fra il punto x e il k -esimo key-point della j -esima persona. Vengono così generati K vettori per ogni punto x all'interno del disco definito che, combinati insieme

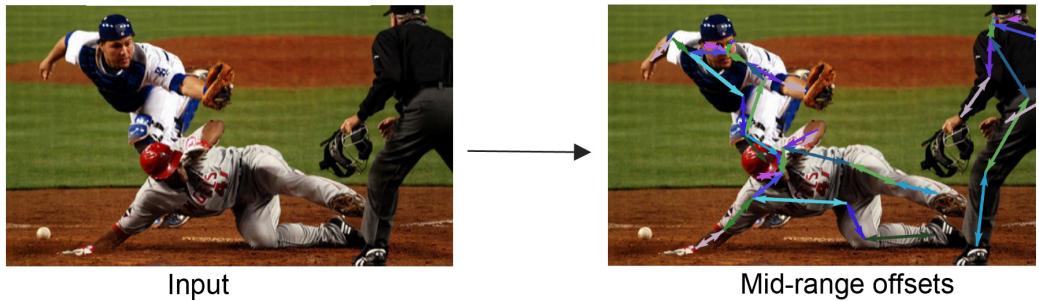


Figura 3.3. Esempio di stima degli offset a medio raggio con PoseNet. L'intento è quello di raggruppare i keypoints appartenenti alla stessa persona.

in una *trasformata di Hough* $h_k(x)$, miglioreranno l'accuratezza della posizione predetta di ogni key-point. Solo i punti che superano una certa soglia di Hough (0.01, come indicato da [1]) vengono considerati dei key-point, gli altri invece vengono scartati.

3.2 Raggruppamento dei key-points in istanze di persona

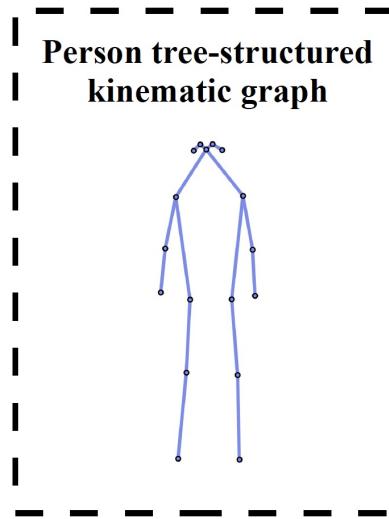


Figura 3.4. Struttura ad albero utilizzata da PoseNet per raggruppare i keypoints appartenenti alla stessa persona

A questo punto è però necessario capire come associare ogni key-point alle persone raffigurate nell'immagine (nel caso ce ne sia più di una). Seguendo lo schema delle connessioni fra tipi di key-points (rappresentati in figura 3.4) la rete viene allenata per restituire in output anche i cosiddetti *offset a medio raggio*, ovvero probabilità di connessioni fra key-points, col lo scopo di raggruppare quelli appartenenti alla stessa persona.

Un esempio di questa stima è raffigurato in figura 3.3 mentre una raffigurazione completa del sistema adottato da PoseNet per il riconoscimento delle pose di persone raffigurate in un'immagine è rappresentato in figura 3.5.

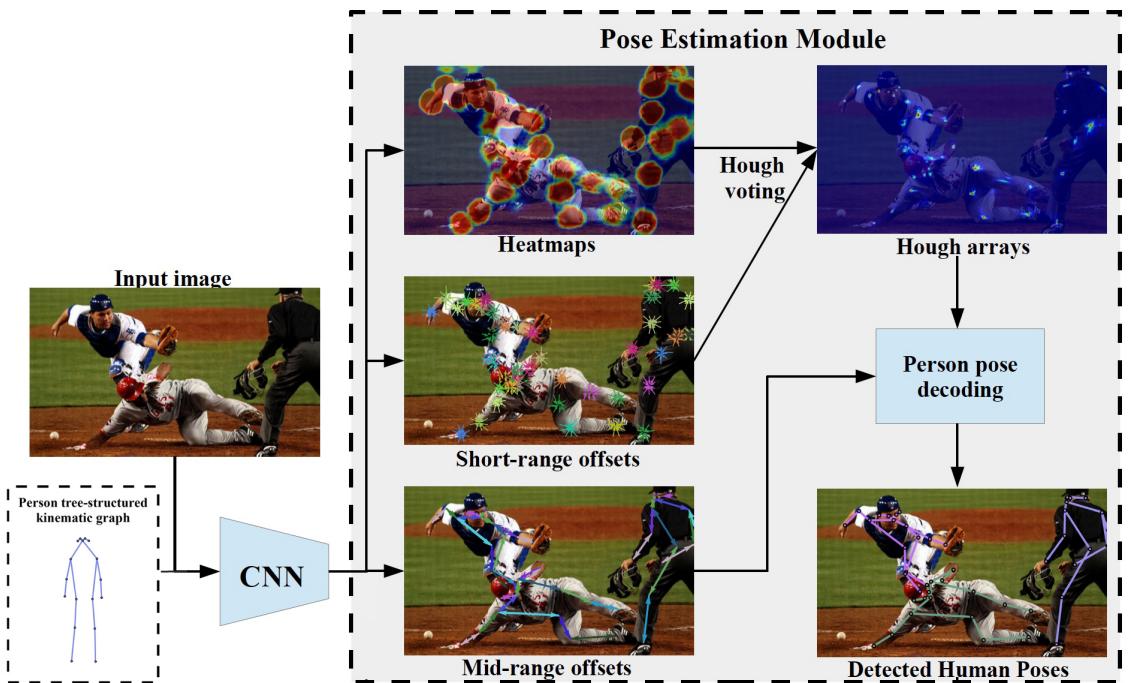


Figura 3.5. Combinazione delle fasi adottate da PoseNet per il riconoscimento della posa in un'immagine.

Capitolo 4

Detectron2

Detectron2 è un progetto open-source lanciato dalla *Facebook AI Research (FAIR)* ampiamente usato dalla comunità di ricerca in ambito *computer vision* e rappresenta, ad oggi, una piattaforma per il riconoscimento di oggetti allo stato dell'arte.

Il suo predecessore *Detectron* [5] fu un progetto iniziato nel 2016 con l'obiettivo di creare un sistema rapido e flessibile per il riconoscimento di oggetti in immagini ed originariamente basato su *Caffe2* [6] (un framework ideato per facilitare la sperimentazione e la divulgazione di nuovi modelli e algoritmi in ambito *deep learning*) e scritto in *Python*.

Negli anni Detectron è stato perfezionato e supportato da una grande quantità di progetti, compreso “*Mask-R-CNN*” [7] e “*Focal Loss for Dense Object Detection*” [8], vincitori rispettivamente del *Premio Marr* e di *Miglior articolo scientifico studentesco* all’*Internation Conference on Compuer Vision (ICCV)* del 2017. L'intuitività e l'efficacia di questi algoritmi hanno permesso un notevole sviluppo nella risoluzione di problemi complessi nell'ambito della computer vision, come ad esempio l'*instance segmentation*, e hanno sicuramente giocato un ruolo rilevante nell'avanzamento tecnologico dei sistemi di riconoscimento visivo.

Detectron2 è adesso basato su *Pytorch*, una libreria open-source dedita al machine learning ed ampiamente usata nel campo della computer-vision che

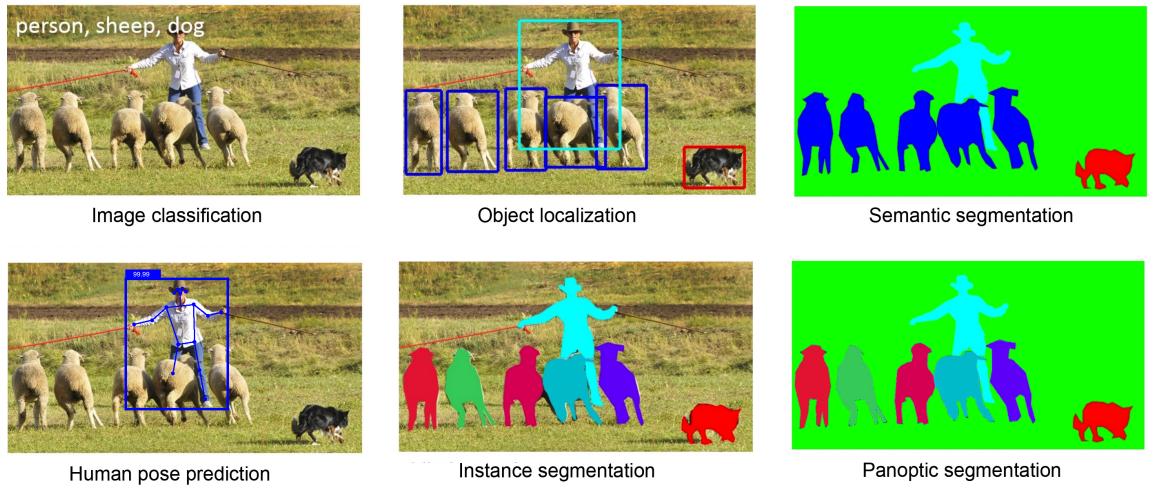


Figura 4.1. Tipologie di analisi visive.

ha inglobato in se anche il precedente framework Caffe2. Più nello specifico Detectron2 include ad oggi le implementazioni dei seguenti algoritmi di object-detection:

- Cascade R-CNN [16]
- Panoptic FPN [17]
- TensorMask [18]
- Mask R-CNN [7]
- RetinaNet [8]
- Faster R-CNN [9]
- RPN [9]
- Fast R-CNN [10]
- R-FCN [11]

utilizzando le seguenti reti *backbone* (ovvero reti precedentemente allenate con lo scopo di estrarre in maniera efficiente le *features* di un'immagine):

- ResNeXt{50,101,152} [12]

- ResNet{50,101,152} [13]
- Feature Pyramid Networks (con ResNet/ResNeXt) [14]
- VGG16 [15]

Inoltre, nel caso fosse necessario implementare nuove reti backbone, è possibile farlo facilmente grazie alla struttura modulare di Pytorch, che permette di separare il nuovo modello dai precedenti algoritmi di Detectron2.

Essendo stato interamente riscritto in Pytorch, Detectron2 è più rapido del suo

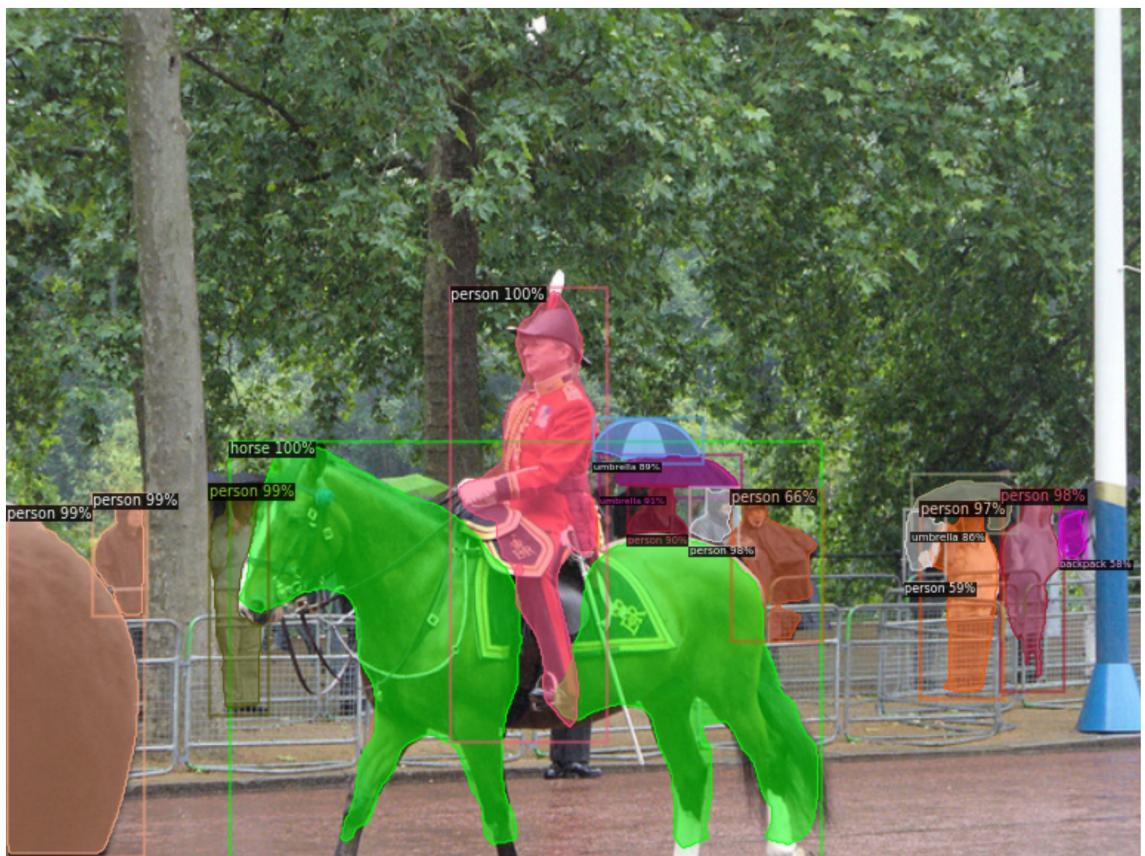


Figura 4.2. Instance segmentation con Detectron2

predecessore nei compiti di *object-detection*, *instance segmentation* e *human-pose prediction*, ed in più è in grado di fornire supporto per i nuovi task di *semantic segmentation* e *panoptic segmentation*, ovvero la combinazione fra instance-segmentation e semantic-segmentation (figura 4.1).

Oltre che nella ricerca, questa piattaforma viene usata anche da numerosi team di Facebook (e non solo) per l'addestramento di nuovi modelli in svariati campi della *computer vision*, come ad esempio la *realtà aumentata*, e in materia di sicurezza informatica, come ad esempio la *community integrity*, ovvero la difesa e la protezione di account su piattaforme social da contenuti maligni.

Per quanto riguarda la stima della posa umana in un'immagine, Detectron2 utilizza una Mask R-CNN [7] riadattata all'estrazione dei keypoints.

4.1 Mask R-CNN

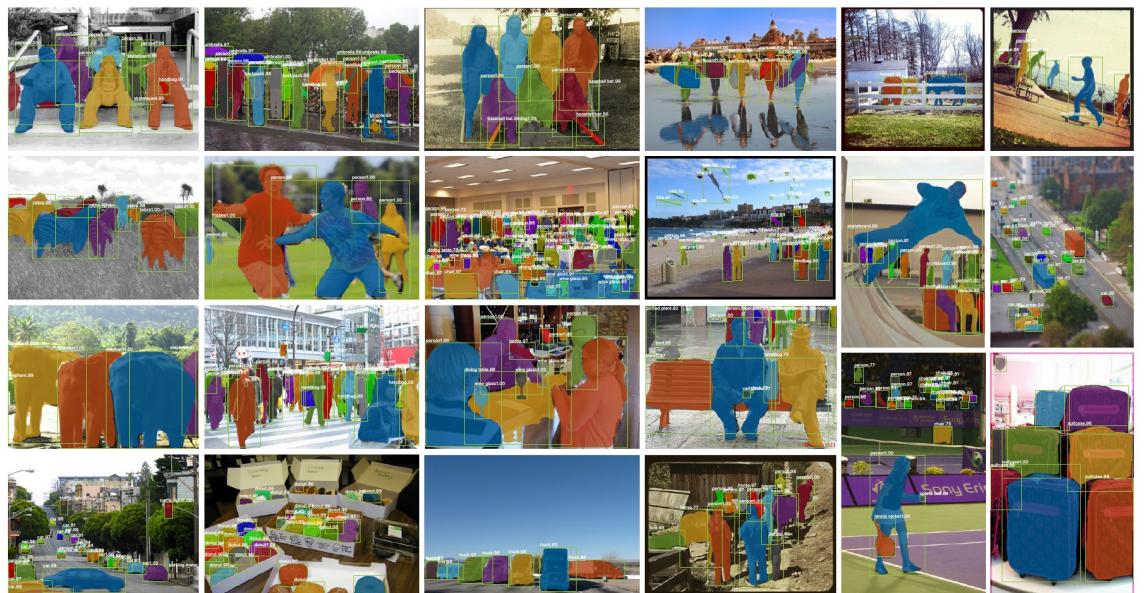


Figura 4.3. Alcuni esempi di instance segmentation con Mask R-CNN

Mask R-CNN nasce con l'intento di creare un framework semplice e flessibile per affrontare il problema dell'*instance segmentation*. Questo metodo è in grado di identificare gli oggetti in un'immagine e simultaneamente generare una maschera di segmentazione ben definita per ogni istanza. Mask R-CNN è sostanzialmente un'estensione di *Faster R-CNN* [9] in quanto aggiunge parallelamente ai due output già presenti per il *bounding box* e la classificazione anche quello per la predizione della maschera d'istanza. Nonostante sia un metodo nato per

l'instance segmentation è però facilmente adattabile ad altri tipi di predizioni, come ad esempio quella della posa umana.

Mask R-CNN è suddiviso in due fasi di procedura. La prima, identica a quella adottata per Faster R-CNN, chiamata “*Region Proposal Network*” (RPN) ha il compito di proporre porzioni di immagine nelle quali potrebbero essere raffigurate delle istanze. Nella seconda, viene predetto *in parallelo* alla classe e al delineamento preciso dell'istanza (*bounding box*) anche una maschera binaria per ogni *regione d'interesse* (RoI) dell'immagine.

Più precisamente, durante l'allenamento della rete, la *loss* definita per ogni RoI è la seguente: $L = L_{cls} + L_{box} + L_{mask}$. La loss di classificazione e quella del boundig box sono identiche a quelle definite per Fast R-CNN [10].

Per quanto riguarda invece L_{mask} , dato che decodifica K maschere binarie di dimensione $m \times m$ (dove K è il numero di classi), la sua dimensione è Km^2 per ogni RoI.

Questa definizione di L_{mask} permette alla rete, non solo di generare maschere completamente indipendenti l'una dall'altra, ma anche di disaccoppiare la predizione delle maschere dalla classificazione delle istanze. Questa caratteristica, innovativa rispetto alle pratiche comuni in materia di semantic segmentation, si è rivelata essere basilare per il raggiungimento di una buona segmentazione.

4.1.1 Predizione della posa con Mask R-CNN

Grazie alla sua grande flessibilità, questo framework può essere facilmente esteso alla stima della posa umana in un'immagine.



Figura 4.4. Alcuni esempi di pose prediction con Mask R-CNN

Le coordinate dei keypoints vengono trasformate in maschere di tipo *one-hot* e Mask R-CNN viene utilizzato per predire K maschere, una per ogni tipo di keypoint (gomito sinistro, spalla destra, etc..).

Più nello specifico, per ognuno dei K keypoint di un'istanza, il training target è una maschera binaria $m \times m$ di tipo *one-hot* dove cioè solo un pixel viene etichettato come positivo. Durante l'allenamento della rete, per ogni keypoint visibile nell'immagine, viene minimizzata una *cross-entropy* loss con regolarizzatore L2 (per incoraggiare l'identificazione di un singolo punto). Anche per questa procedura, le K maschere dei K tipi di keypoint sono completamente indipendenti l'una dall'altra.

Capitolo 5

Il dataset NTU RGB+D

Il recente sviluppo dei sensori di profondità ha permesso un notevole miglioramento degli studi in ambito 3D, come ad esempio il riconoscimento di oggetti o azioni in scenari tridimensionali.

Essendo però un campo scientifico piuttosto “giovane”, la ricerca di un adeguato dataset per il *riconoscimento di azioni umane* che sia solido, voluminoso e vario può non essere un compito facile.

Spesso i dataset pubblicamente disponibili sono composti da uno stretto gruppo di soggetti, il che riduce notevolmente la variabilità intra-classi e un’attività *umana* dipende fortemente dall’età, il sesso, la cultura e le condizioni fisiche di chi la svolge quindi avere una sufficiente variabilità di soggetti nel dataset che stiamo analizzando è di vitale importanza.

Un’altra cosa estremamente importante è il numero di azioni che stiamo analizzando. Se il nostro dataset contiene poche tipologie di attività umane, sarà facile distinguerle e classificarle fra loro, magari identificando un semplice pattern di movimento o addirittura la struttura di un oggetto coinvolto nell’azione da classificare. Se invece il numero di azioni trattate è sufficientemente alto, allora i pattern di movimento e gli oggetti con i quali si interagisce vengono condivisi fra più classi di azioni, rendendo la classificazione estremamente più difficile.

Il terzo punto da considerare nella scelta di un buon dataset è il punto di vista dal quale vengono riprese le azioni. La maggior parte dei dataset contengono video che riprendono l’azione da un unico punto di vista (spesso frontale), in

altri casi invece i punti di vista sono strettamente due (magari frontale e di lato), ottenuti solo perchè sono state utilizzate più telecamere contemporaneamente.

Come ultima cosa e forse la più importante, l'insufficienza di video nei dataset ci impedisce spesso di applicare tecniche di machine-learning al nostro problema, portandoci inevitabilmente ad una situazione di overfitting.

Consapevoli di quanto detto finora Amir Shahroudy, Jun Liu, Tian-Tsong Ng e Gang Wang hanno creato nel 2016 *NTU RGB+D* [19], un vasto e variegato dataset dedicato all'analisi di attività umane in ambito 3D. NTU RGB+D si compone



Figura 5.1. Alcuni esempi di frames del dataset NTU RGB+D. Nelle prime 4 righe è possibile notare la varietà degli attori partecipanti al progetto e dei diversi punti di ripresa utilizzati. La quinta riga mostra la varietà intra-classe per una stessa azione. L'ultima riga mostra per uno stesso frame i valori RGB, i valori RGB + giunti, mappa di profondità, mappa di profondità + giunti e i valori a infrarossi.

di 56880 RGB+D video, ottenuti riprendendo 40 differenti attori ed utilizzando Microsoft Kinect v2. Il dataset contiene i video RGB, le sequenze di profondità, lo skeleton dei soggetti (ovvero le posizioni 3D dei principali giunti corporei) e i frames a infrarossi. I soggetti sono stati ripresi da 80 punti di vista differenti e la loro età varia dai 10 ai 35 anni, il che rende più realistica la variazione di qualità delle azioni svolte. Anche se tutte le riprese sono di tipo *indoor*, gli scenari ricreati intorno agli attori sono estremamente variabili.

5.1 NTU RGB+D in dettaglio

Tipologie di dati - I dati raccolti sono stati ottenuti con sensori Microsoft Kinect v2 che consentono di estrapolare dalle riprese 4 tipologie di dato: mappe di profondità, informazioni 3D dei giunti corporei, frame RGB e sequenze a infrarossi.

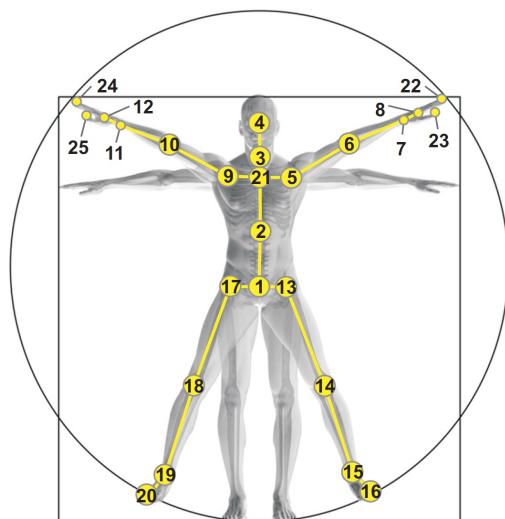


Figura 5.2. La configurazione dei 25 giunti utilizzati in NTU RGB+D. Le etichette utilizzate per i giunti sono: 1 - base della colonna vettrebrale, 2 - punto medio della colonna vertebrale, 3 - collo, 4 - testa, 5 - spalla sinistra, 6 - gomito sinistro, 7 - polso sinistro, 8 - mano sinistra, 9 - spalla destra, 10 - gomito destro, 11 - polso destro, 12 - mano destra, 13 - anca sinistra, 14 - ginocchio sinistro, 15 - caviglia sinistra, 16 - piede sinistro, 17 - anca destra, 18 - ginocchio destro, 19 - caviglia destra, 20 - piede destro, 21 - colonna vertebrale, 22 - punta della mano sinistra, 23 - pollice sinistro, 24 - punta della mano destra, 25 - pollice destro.

Le *mappe di profondità* sono sequenze di valori in millimetri che rappresentano la distanza di ogni punto dalla telecamera. Ogni mappa di profondità è stata poi ridotta ad una risoluzione di 512×424 senza perdita di informazione.

Le *informazioni 3D dei giunti corporei* sono coordinate 3D dei 25 giunti corporei più importanti. Per ogni giunto e per ogni frame viene inoltre saltato il corrispondente pixel nel video RGB e la relativa mappa di profondità. I 25 giunti corporei utilizzati sono raffigurati in figura 5.2.

I *video RGB* sono normali video registrati con una risoluzione 1920×1080 .

Le *sequenze a infrarossi* sono, come per le mappe di profondità, salvate frame per frame con una dimensione di 512×424 .

Le classi delle azioni - Il dataset contiene 60 tipi di azioni differenti divise in 3 gruppi principali: 40 azioni *quotidiane* (bere, mangiare, leggere, etc.), 9 azioni *legale alla salute* (starnutire, barcollare, svenire, etc...) e 11 azioni *di coppia* (dare un cazzotto a qualcuno, abbracciarsi, etc...)

I soggetti - Per la realizzazione di questo dataset sono stati utilizzati 40 attori di età compresa dai 10 ai 35 anni. In figura 5.1 è possibile vedere come l'insieme delle persone scelte sia estremamente variegato in età, altezza e sesso. Ogni attore è rappresentato con un ID unico in tutto il dataset.

I punti di ripresa - Ogni azione è stata registrata da 3 telecamere contemporaneamente di modo da generare 3 prospettive diverse della stessa azione. Le 3 telecamere sono state posizionate tutte alla stessa altezza e sempre con la stessa angolazione di ripresa, ovvero -45° , 0° e 45° .

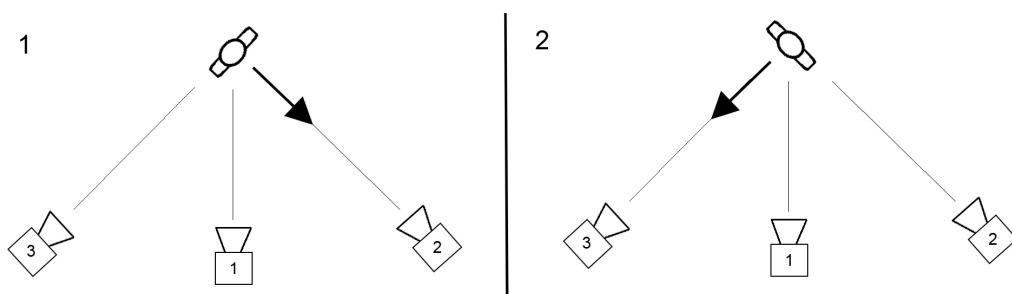


Figura 5.3. Disposizione delle telecamere per ogni setup. La telecamera 1 ha ripreso ogni azione da una posizione centrale mentre le telecamere 2 e 3 hanno ripreso le azioni con un'angolazione di 45° . Ogni attore ha ripetuto l'azione 2 volte: una rivolto verso la telecamera di destra e una verso quella di sinistra.

Ad ogni attore è stato inoltre richiesto di ripetere l'azione due volte: una rivolto verso la telecamere di sinistra ed una verso quella di destra. Così facendo le 6 riprese ottenute forniscono 5 prospettive diverse della stessa azione, ovvero quella frontale (ripresa 2 volte), quella dal profilo sinistro, quella dal profilo destro, quella ad una angolazione di 45° da sinistra e quella ad una angolazione di 45° da destra. Come per gli attori, l'ID assegnato ad ogni telecamera è fisso per tutto il dataset: la telecamera 1 è sempre quella che riprende da 0° , la telecamera 2 da 45° e la 3 da -45° . Uno schema della disposizione delle telecamere e della procedura di ripresa è raffigurato in figura 5.3

Per aumentare ulteriormente la variabilità delle riprese, ad ogni *setup* (ovvero ad ogni set di ripresa) le 3 telecamere sono state posizionate ad un'altezza e ad una distanza diversa dall'attore inquadrato, ovviamente mantenendo sempre la stessa altezza fra le 3 telecamere. In tabella 5.1 sono elencate le posizioni delle 3 telecamere per ogni setup.

Setup	Altezza (m)	Distanza (m)	Setup	Altezza (m)	Distanza (m)
1	1.7	3.5	2	1.7	2.5
3	1.4	2.5	4	1.2	3.0
5	1.2	3.0	6	0.8	3.5
7	0.5	4.5	8	1.4	3.5
9	0.8	2.0	10	1.8	3.0
11	1.9	3.0	12	2.0	3.0
13	2.1	3.0	14	2.2	3.0
15	2.3	3.5	16	2.7	3.5
17	2.5	3.0			

Tabella 5.1. Altezza e distanza delle 3 telecamere per ogni setup di ripresa.
Le altezze e le distanze sono espresse in metri.

5.2 Criteri di valutazione

Per avere una valutazione standard dei risultati ottenuti su questo dataset, gli autori definiscono nel loro lavoro [19] due particolari criteri di valutazione, di modo da poter allineare anche i risultati futuri e poterli comparare fra loro.

5.2.1 Valutazione Cross-subject

Per questa valutazione, i 40 attori sono stati divisi in due gruppi: *training* e *test* ed ogni gruppo è costituito da 20 attori. Così facendo è stato ottenuto un insieme di training di 40320 video e un insieme di test di 16560 video.

Gli ID degli attori dell'insieme di training sono: 1, 2, 4, 5, 8, 9, 13, 14, 15, 16, 17, 18, 19, 25, 27, 28, 31, 34, 35, 38. I rimanenti fanno parte dell'insieme di test.

5.2.2 Valutazione Cross-view

Per quanto riguarda invece la valutazione *cross-view*, tutti i video ripresi dalle telecamere 2 e 3 costituiscono l'insieme di training, mentre quelli ripresi dalla telecamera 1 l'insieme di test. In altre parole l'insieme di training include tutte le prospettive frontali del soggetto inquadrato e tutte quelle di profilo mentre l'insieme di test include tutte le prospettive a 45°. Dividendo il dataset con questo criterio si ottiene un insieme di training composto da 37920 video e quello di test da 18960.

Capitolo 6

Classificazione

6.1 Struttura della rete

6.2 Tecniche

6.2.1 Semplice

6.2.2 Tecnica dei centri

6.2.3 Tecnica delle differenze

Capitolo 7

Risultati ottenuti

Capitolo 8

Conclusioni

Capitolo 9

Sviluppi futuri

Bibliografia

- [1] PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model - *George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, Kevin Murphy* - 2018
- [2] Coco 2016 keypoint challenge - Lin, T.Y., Cui, Y., Patterson, G., Ronchi, M.R., Bourdev, L., Girshick, R., Dollr,P. - 2016
- [3] PoseNet with TensorFlow.js - <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5>
- [4] Detectron2 - *Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, Ross Girshick* - <https://github.com/facebookresearch/detectron2>, 2019
- [5] Detectron - *Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollàr, Kaiming He* - <https://github.com/facebookresearch/detectron>, 2018
- [6] Caffe2 - <https://caffe2.ai/docs>
- [7] Mask R-CNN - *Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick*, 2017
- [8] Focal Loss for Dense Object Detection - *Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár*, 2017
- [9] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks - *Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun*, 2015
- [10] Fast R-CNN - *Ross Girshick*, 2015

- [11] R-FCN: Object Detection via Region-based Fully Convolutional Networks - *Jifeng Dai, Yi Li, Kaiming He, Jian Sun*, 2016
- [12] Aggregated Residual Transformations for Deep Neural Networks - *Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He*, 2016
- [13] Deep Residual Learning for Image Recognition - *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun*, 2015
- [14] Feature Pyramid Networks for Object Detection - *Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie*, 2016
- [15] Very Deep Convolutional Networks for Large-Scale Image Recognition - *Karen Simonyan, Andrew Zisserman*, 2014
- [16] Cascade R-CNN: High Quality Object Detection and Instance Segmentation - *Zhaowei Cai, Nuno Vasconcelos* - 2019
- [17] Panoptic Feature Pyramid Networks - *Alexander Kirillov, Ross Girshick, Kaiming He, Piotr Dollár* - 2019
- [18] TensorMask: A Foundation for Dense Object Segmentation - *Xinlei Chen, Ross Girshick, Kaiming He, Piotr Dollár* - 2019
- [19] NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis - *Amir Shahroudy, Jun Liu, Tian-Tsong Ng, Gang Wang* - 2016