



UNIVERSITÀ  
DEGLI STUDI  
FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali  
Corso di Laurea Magistrale in Informatica

Tesi di Laurea

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

HUMAN ACTION RECOGNITION USING  
DEEP LEARNING TECHNIQUES FOR POSE  
ESTIMATION

ANDREA MOSCATELLI

Relatore: *Marco Bertini*

Anno Accademico 2018-2019

Andrea Moscatelli: *Riconoscimento di azioni umane usando tecniche di apprendimento profondo per la stima della posa*, Corso di Laurea Magistrale in Informatica, © Anno Accademico 2018-2019

---

## INDICE

---

1	L'apprendimento automatico	11
1.1	Deep learning	13
1.2	Le reti neurali	13
1.3	Le reti neurali ricorrenti	16
1.4	Le reti LSTM	18
2	Lavori precedenti	23
2.1	Recognizing Human Actions as Evolution of Pose Estimation Maps (2018)	24
2.2	Action Machine: Rethinking Action Recognition in Trimmed Videos (2018)	28
2.3	Overview	31
3	Stima della posa	33
3.1	PoseNet	34
3.1.1	Stima dei keypoint	36
3.1.2	Raggruppamento dei keypoint in istanze di persona	37
3.2	Detectron-2	38
3.2.1	Mask R-CNN	41
3.2.2	Predizione della posa con Mask R-CNN	42
4	Metodo proposto	45
4.1	Estrazione delle pose	45
4.2	Assegnazione coerente delle pose	46
4.3	Rimozione degli zeri	48
4.4	Tecniche di rielaborazione	49
4.4.1	Baricentro del frame	50
4.4.2	Baricentro del video	51
4.4.3	Baricentri multipli	52
4.4.4	Tecnica "Next frame"	54
4.4.5	Tecnica delle distanze cumulate	55
4.4.6	Tecnica delle distanze relative	56
4.5	Normalizzazione	57
4.6	Smussamento	58
4.7	Struttura della rete	58
4.8	Criterio di arresto dell'addestramento	60
5	Il dataset NTU RGB+D	61

2 Indice

5.1	NTU RGB+D in dettaglio	63
5.2	Criteri di valutazione	65
5.2.1	Valutazione Cross-subject	65
5.2.2	Valutazione Cross-view	66
6	Esperimenti e risultati	67
6.1	Una porzione rappresentativa del dataset	68
6.2	Fase 1: Test di tutte le tecniche	68
6.3	Fase 2: Numero di livelli	70
6.4	Fase 3: Dropout e regolarizzatori	72
6.5	Fase 4: Allenamento completo tecnica migliore	72
7	Conclusioni e sviluppi futuri	73
7.1	Sviluppi futuri	73

---

## ELENCO DELLE FIGURE

---

- Figura 1 Un neurone biologico (sinistra) ed un neurone artificiale (destra) a confronto. La struttura del neurone artificiale si ispira molto a quella del neurone biologico. 14
- Figura 2 Esempio di una *fully connected network*. In questo esempio di rete abbiamo 2 *hidden layer* (in viola chiaro) composti da 5 *hidden units* ciascuno (in viola scuro). Maggiore è il numero di hidden units e maggiore è la capacità espressiva della rete. 15
- Figura 3 Un esempio di rete convoluzionale per l'analisi di un'immagine. *Conv* = "Convolution layer", moltiplicazione vettoriale della porzione di layer al quale ogni neurone è collegato; *Pool* = "Pooling layer", sintetizzazione della porzione di layer connessa; *FC* = "Fully connected layer"; *Softmax* = operazione conclusiva per la decisione del valore di output. 16
- Figura 4 Un esempio di rete ricorrente con due hidden layers. 17
- Figura 5 Esempio di "srotolamento" di una rete ricorrente, ovvero la trasformazione da una rete ricorrente ad una combinazione equivalente di reti neurali semplici. 18
- Figura 6 Esempio di "srotolamento" di una rete LSTM. In questa figura ogni unità di rete "A" prende in input al tempo  $t$  un elemento  $X_t$  della sequenza temporale  $X$  e restituisce uno stato in output  $h_t$ .  $\tanh$  e  $\sigma$  sono le funzioni di tangente iperbolica e sigmoidale rispettivamente, mentre "+" e "×" sono rispettivamente le operazioni vettoriali di somma e moltiplicazione. 19
- Figura 7 I concetti chiave di una LSTM. 19
- Figura 8 Forget gate di una LSTM 20
- Figura 9 Input gate. 21
- Figura 10 Output gate di una LSTM 21

4 Elenco delle figure

Figura 11	Processo di creazione della heatmap e della posa a partire da un singolo frame del video. Ogni giunto corporeo viene stimato singolarmente per poi essere successivamente combinato agli altri.	25
Figura 12	Sintetizzazione temporale delle heatmap.	25
Figura 13	Sintetizzazione spazio-temporale delle heatmap.	26
Figura 14	Sintetizzazione spazio-temporale delle heatmap.	27
Figura 15	Schematizzazione di <i>Action Machine</i>	28
Figura 16	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.	33
Figura 17	Un esempio di utilizzo in campo medico della stima della posa	34
Figura 18	Generazione con PoseNet delle heatmap per ogni tipologia di keypoint	36
Figura 19	Esempio di stima degli offset a corto raggio con PoseNet	37
Figura 20	Struttura ad albero utilizzata da PoseNet per raggruppare i keypoint appartenenti alla stessa persona	37
Figura 21	Esempio di stima degli offset a medio raggio con PoseNet. L'intento è quello di raggruppare i keypoint appartenenti alla stessa persona.	38
Figura 22	Combinazione delle fasi adottate da PoseNet per il riconoscimento della posa in un'immagine.	38
Figura 23	Tipologie di analisi visive.	39
Figura 24	Instance segmentation con Detectron-2	41
Figura 25	Alcuni esempi di instance segmentation con Mask R-CNN	42
Figura 26	Alcuni esempi di pose prediction con Mask R-CNN	43
Figura 27	Un esempio di assegnazione inconsistente delle pose lungo il video. Nell'esempio, l'ordine delle pose restituite dall'algoritmo è blu, rosso, verde e viola. Quest'ordine è indipendente dai soggetti inquadrati, che possono talvolta essere scambiati fra loro, non essere riconosciuti o addirittura scambiati per dei riflessi.	47

- Figura 28 Esempio di rimozione degli zeri. Nei frame 2-3-4-5 i giunti del braccio sinistro non sono stati identificati. Nei frame 2-3 verranno quindi rimpiazzati con quelli del frame 1, mentre nei frame 4-5 con quelli del frame 6. 49
- Figura 29 Esempio di applicazione della *tecnica del baricentro del frame*. In figura sono rappresentati 3 frame del video di una camminata da sinistra a destra. Ogni punto blu (relativo ad un giunto corporeo) viene sostituito col vettore-distanza (in verde) fra quel punto e il baricentro della posa. 50
- Figura 30 Esempio di applicazione della *tecnica del baricentro del video*. In figura sono rappresentati 3 frame del video di una camminata da sinistra a destra. Ogni punto blu (relativo ad un giunto corporeo) viene sostituito col vettore-distanza (in arancione) fra quel punto e il baricentro del video. 51
- Figura 31 Esempio di applicazione della *tecnica dei 3 baricentri*. In figura sono rappresentati 3 frame del video di una camminata da sinistra a destra. Ogni punto blu (relativo ad un giunto corporeo) viene sostituito col vettore-distanza fra quel punto e il baricentro al quale quel giunto appartiene. 53
- Figura 32 Tecnica del *next frame*. Nell'esempio, il soggetto ripreso alza il braccio sinistro. Ogni posa viene sostituita con l'insieme dei vettori-distanza (frecce rosa) fra ogni giunto e la posizione dello stesso al frame successivo. 54
- Figura 33 Tecnica delle *distanze cumulate*. Nell'esempio sono raffigurati i valori che questa tecnica restituirebbe per l'azione "alzare e abbassare il braccio destro". Ovviamente i valori più alti sono legati al polso e al gomito destro visto che sono le componenti più dinamiche di questo movimento. 55
- Figura 34 Tecnica delle *distanze relative*. In questa figura sono rappresentati solo i vettori-distanza dalla caviglia destra a tutti gli altri giunti. 56

## 6 Elenco delle figure

- Figura 35 Differenza fra normalizzazione *globale* e *separata*. La posa blu e la posa rossa sono identiche, con l'unica differenza di essere posizionate rispettivamente nella metà sinistra e destra del frame. In questo caso stiamo trattando punti nel piano e solo normalizzandoli con la tecnica *separata* si ottengono due pose identiche. 57
- Figura 36 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. 62
- Figura 37 La configurazione dei 25 giunti utilizzati in NTU RGB+D. Le etichette utilizzate per i giunti sono:  
1 - base della colonna vettoriale, 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. 63
- Figura 38 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 telecamere di destra e una verso quella di sinistra. 64
- Figura 39 Risultati ottenuti per ogni tecnica allenando una rete con un solo livello LSTM. CV= Cross View; CS= Cross Subject. 69

Figura 40 Risultati ottenuti facendo variare la profondità della rete. I valori in grassetto sono i migliori per la tecnica corrispondente. 71



*"I computer sono incredibilmente veloci, accurati e stupidi.  
Gli uomini sono incredibilmente lenti, inaccurati e intelligenti.  
L'insieme dei due costituisce una forza incalcolabile."*

— Albert Einstein



# 1

---

## L'APPRENDIMENTO AUTOMATICO

---

Quando parliamo di apprendimento automatico (in inglese *machine learning*), ovvero *l'apprendimento delle macchine*, viene spontaneo domandarsi "Come impara una macchina?".

L'apprendimento viene definito come un processo iterativo che permette di mutare le proprie conoscenze a seconda delle informazioni che vengono raccolte.

Nel machine learning l'apprendimento segue esattamente le stesse dinamiche ma a seconda di come gestiamo le informazioni a nostra disposizione vengono definite 4 macro classi.

- **Apprendimento supervisionato (Supervised Learning)** - questo primo caso, può essere paragonato al processo di apprendimento che si avrebbe con un'insegnante *onnisciente* che supervisiona il suo alunno e che interromperà l'apprendimento solo quando quest'ultimo raggiungerà un livello accettabile del compito da imparare.

L'apprendimento supervisionato lo si ha cioè 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. 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. 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. L'obiettivo che quindi ci si pone in questo tipo di situazione è scovare 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'obiettivo di identificare schemi frequenti, associazioni, correlazioni o strutture casuali fra gli *item* di un database relazionale, cercando 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)** - questo tipo di apprendimento è una sorta di via di mezzo dei primi due, ovvero un apprendimento utilizzato quando solo alcuni dei nostri dati in input  $X$  sono etichettati, mentre la maggior parte di essi non lo è.

In questa categoria di apprendimento ricadono la stragrande maggioranza dei problemi di machine learning e questo perché etichettare dati è un processo lungo e costoso soprattutto in ambito *Big Data*, campo ideale per il machine learning.

- **Apprendimento con rinforzo (Reinforcement Learning)** - questa metodologia di apprendimento simula il processo di apprendimento umano per tentativi ed errori. L'algoritmo si adatta gradualmente tramite un sistema di valutazione basato su ricompense e penalità a seconda della decisione presa. L'obbiettivo è quello di evolvere l'algoritmo massimizzando le ricompense ricevute.

## 1.1 DEEP LEARNING

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

Nel deep learning un computer impara, ad esempio, un task di classificazione direttamente da immagini, testi e suoni e può raggiungere prestazioni allo stato dell'arte superando talvolta persino l'accuratezza umana. I modelli vengono allenati usando grandi dataset etichettati seguendo un'apprendimento di tipo supervisionato ed il sistema centrale sul quale è basato il deep learnig è la *rete neurale*.

## 1.2 LE RETI NEURALI

Nel campo dell'apprendimento automatico, una rete neurale (*neural network* o *NN*) è un modello di calcolo ispirato al sistema di elaborazione delle informazioni tipico del cervello degli esseri viventi, nel quale tanti

piccoli elementi base, denominati *neuroni* (figura 1), sono interconnessi e collaborano tra loro per poter eseguire un determinato compito.

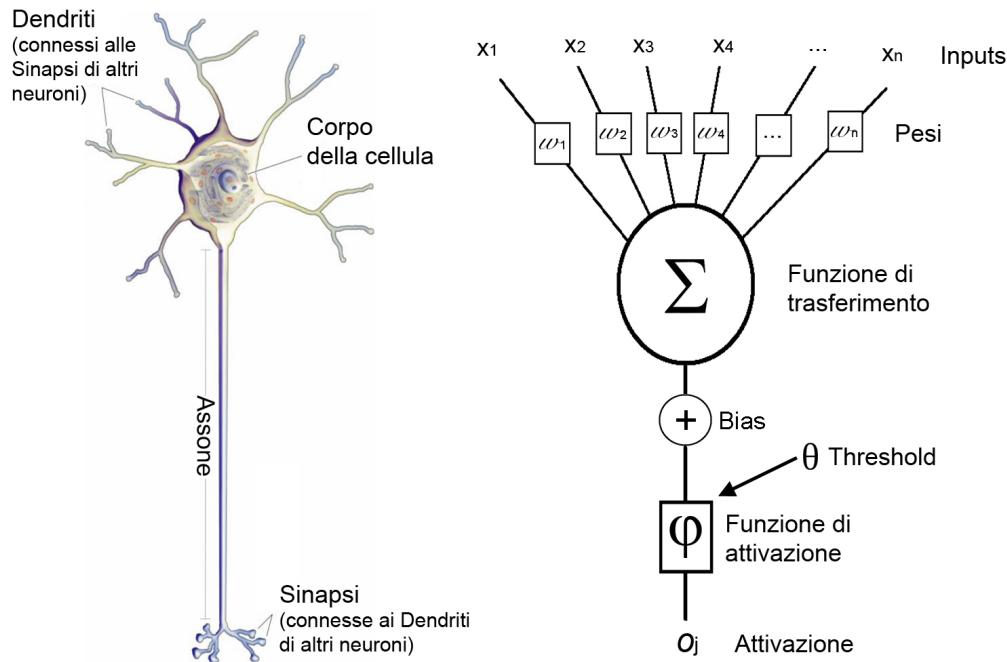


Figura 1: Un neurone biologico (sinistra) ed un neurone artificiale (destra) a confronto. La struttura del neurone artificiale si ispira molto a quella del neurone biologico.

Queste reti costituiscono ad oggi la miglior soluzione per una vasta gamma di problemi grazie alla loro capacità di saper approssimare pressoché qualsiasi funzione  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , a patto di scegliere la giusta quantità di neuroni in fase di progettazione.

La struttura di una rete neurale si sviluppa per livelli ed ogni livello, eccetto quello di *input* e quello di *output*, viene chiamato *hidden layer*. Ogni hidden layer può essere composto da un qualsiasi numero di neuroni, che prendono il nome di *hidden units* e maggiore sarà il numero di hidden units, maggiore sarà la *capacità espressiva* della rete, ovvero la capacità di approssimare correttamente una qualsiasi funzione.

Sono proprio gli hidden layer a dare *profondità* alla rete ed è per questo che si parla sia di *apprendimento profondo* che di *reti neurali profonde* (o *deep neural network* in inglese).

Negli ultimi anni le deep neural network hanno avuto una diffusione sorprendente grazie non solo alla loro adattabilità ad una vasta gamma di problemi, ma anche alla rapida evoluzione vista nel mondo delle GPU,

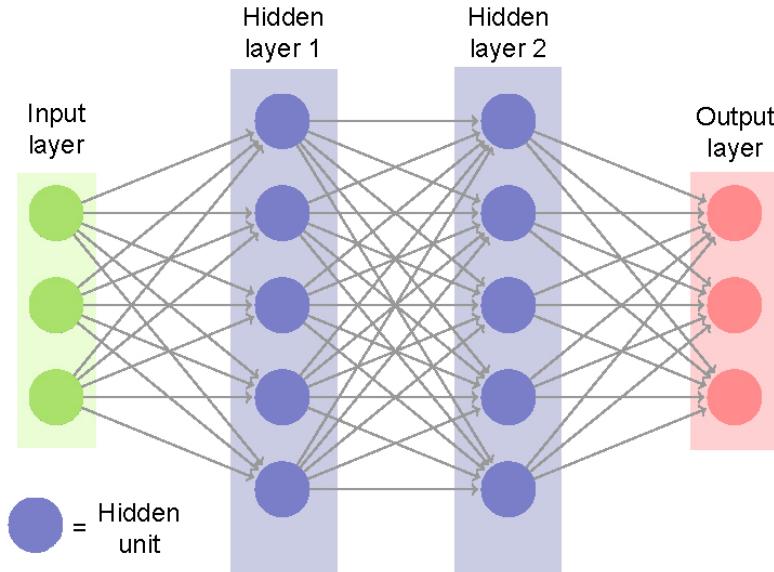


Figura 2: Esempio di una *fully connected network*. In questo esempio di rete abbiamo 2 *hidden layer* (in viola chiaro) composti da 5 *hidden units* ciascuno (in viola scuro). Maggiore è il numero di hidden units e maggiore è la capacità espressiva della rete.

che ha reso parallelizzabile e velocizzato notevolmente l’addestramento di queste reti.

I livelli che possiamo trovare all’interno di una rete neurale sono di diverso tipo, come diverse possono essere le connessioni dei neuroni fra un livello e l’altro. Sono proprio queste diversità a determinare il tipo di rete neurale e le più famose tipologie sono:

- **Fully connected network** - in questo tipo di reti ogni neurone è connesso con tutti quelli del livello precedente (come in figura 2). Come si può facilmente intuire, il numero dei parametri (ovvero la capacità espressiva) della rete cresce esponenzialmente al crescere della sua struttura ed il rischio che si corre è quello di creare una rete talmente efficiente nell’ottimizzare la funzione generatrice dei dati di allenamento da essere troppo specifica per essa, non riuscendo a predire correttamente i valori al di fuori di quel dataset. Questa problematica, ben nota nel mondo del machine learning, prende il nome di *overfitting*.
- **Convolutional neural network (CNN)** - queste reti sono composte principalmente da *layer convoluzionali*, ovvero layer nei quali ogni

neurone è connesso solo ad una piccola regione localizzata del livello precedente (figura 3). Questa particolare tipologia di connessione è estremamente efficiente nell'ambito della visione computerizzata ed è in grado di *sintetizzare* correttamente un'immagine senza perdita d'informazione. Questa sintetizzazione viene indicata in gergo col termine di *features extraction*, ovvero l'estrazione delle caratteristiche principali di un certo dato in input.

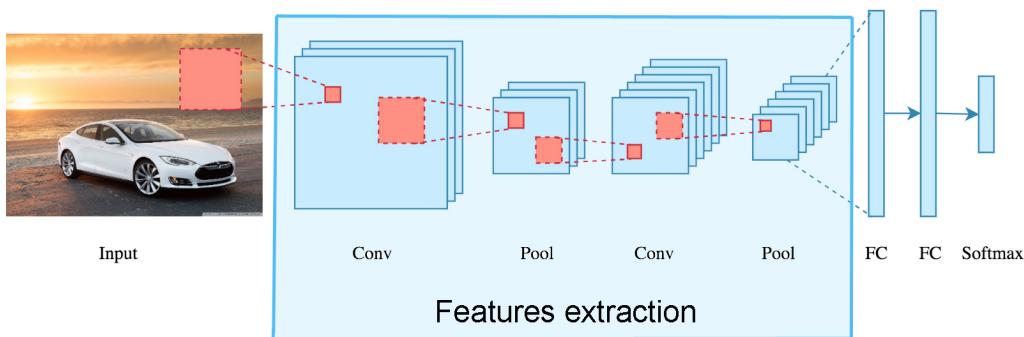


Figura 3: Un esempio di rete convoluzionale per l'analisi di un'immagine. *Conv* = “Convolution layer”, moltiplicazione vettoriale della porzione di layer al quale ogni neurone è collegato; *Pool* = “Pooling layer”, sintetizzazione della porzione di layer connessa; *FC* = “Fully connected layer”; *Softmax* = operazione conclusiva per la decisione del valore di output.

Le reti neurali sopra citate, seppur molto efficienti nei loro ambiti, presentano però un problema di *memoria* ovvero, non sono state ideate per processare una serie temporale di dati. L'unica soluzione che hanno è quella di trattare tutta la serie temporale come un unico input e processarla interamente in un unico ciclo. Questa primordiale soluzione non ha ovviamente preso campo vista la mancanza di generalità.

Esistono però delle reti neurali che, a differenza dalle precedenti, riescono ad analizzare una sequenza temporale in maniera simile al ragionamento umano, ovvero mantenendo una memoria del “passato” ed in base a quest’ultima adattare le scelte e la valutazione dello stato corrente. Queste particolari reti vengono chiamate *reti neurali ricorrenti* (o *recurrent neural networks*).

### 1.3 LE RETI NEURALI RICORRENTI

Le reti neurali ricorrenti (*recurrent neural network* o *RNN*) sono una classe di rete neurali artificiali caratterizzate da una struttura ciclica, ovvero

una struttura dove i layer possono essere connessi con loro stessi o con layer precedenti (figura 4).

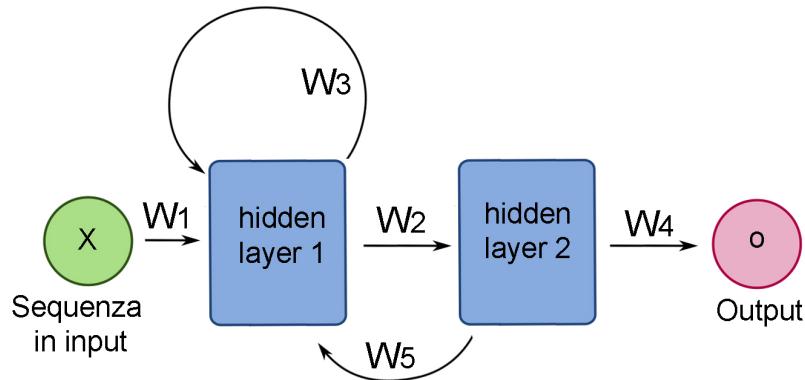


Figura 4: Un esempio di rete ricorrente con due hidden layers.

Questa struttura ciclica permette di mantenere una sorta di *stato di memoria* della rete, utilizzato per processare ogni elemento della sequenza in input. Questa struttura si è rivelata ideale per compiti di analisi predittiva su sequenze temporali, quali possono essere ad esempio il riconoscimento della grafia, l’analisi di video, il riconoscimento vocale o l’analisi di testi.

Per quanto riguarda l’addestramento di queste reti, esso avviene con la stessa modalità delle reti neurali standard, ovvero i pesi dei neuroni che compongono la rete vengono regolati tramite un processo iterativo con lo scopo di minimizzare l’errore di classificazione prodotto sul dataset di allenamento.

Tipicamente, l’algoritmo che regola l’apprendimento delle RNN viene chiamato *Backpropagation Through Time (BPTT)* e non è altro che la versione “ricorrente” del classico *backpropagation algorithm* usato per le reti neurali standard.

L’applicazione del BPTT può essere vista come l’applicazione del backpropagation algorithm allo “srotolamento” della RNN (figura 5).

La lunghezza delle sequenze che queste reti sono in grado di analizzare è teoricamente infinita ma di fatto, per sequenze troppo lunghe, si presenta un problema legato alla precisione finita dei calcolatori, ovvero la *scomparsa del gradiente* (in inglese *vanishing gradient problem*).

Il concetto alla base del backpropagation algorithm è infatti quello di aggiornare iterativamente ogni parametro del modello in maniera proporzionale alla derivata parziale della *loss function* (ovvero la funzione d’errore di classificazione) rispetto al parametro stesso.

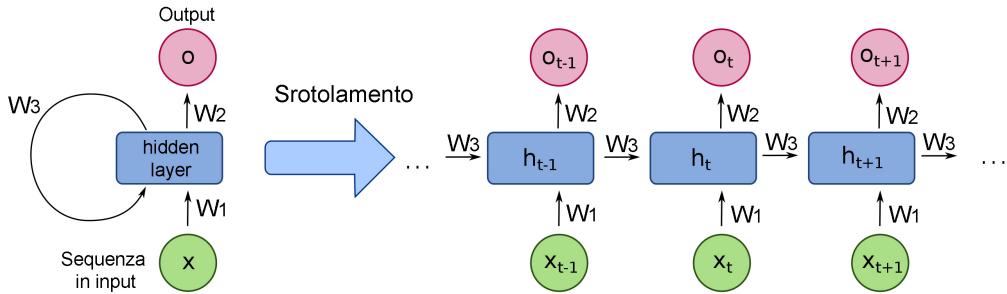


Figura 5: Esempio di “srotolamento” di una rete ricorrente, ovvero la trasformazione da una rete ricorrente ad una combinazione equivalente di reti neurali semplici.

Poiché ogni gradiente sarà un valore compreso fra 0 e 1 e l’aggiornamento dei parametri ai vari livelli verrà propagato tramite la *regola della catena*, ovvero

$$\frac{\partial a}{\partial b} = \frac{\partial a}{\partial c} \cdot \frac{\partial c}{\partial b},$$

il prodotto dei gradienti decrescerà esponenzialmente al crescere della profondità della rete, col rischio di sparire se dovesse diventare più piccolo della precisione minima del calcolatore.

La profondità delle reti ricorrenti è strettamente legata alla lunghezza della sequenza analizzata e raggiunge velocemente un numero di livelli intrattabile per l’algoritmo di backpropagation, rendendole di fatto inutilizzabili per i casi nel mondo reale.

Per ovviare a questo problema, nel 1997 Sepp Hochreiter e Jürgen Schmidhuber proposero un nuovo tipo di rete ricorrente chiamato *Long short-term memory (LSTM)* [21].

#### 1.4 LE RETI LSTM

Le reti Long Short-Term Memory, normalmente chiamate LSTM, sono delle particolari reti ricorrenti in grado di imparare non solo le dipendenze *a lunga distanza* presenti in una sequenza in input ma anche di risolvere il problema della scomparsa del gradiente.

Come ogni altra rete ricorrente anche per le LSTM è possibile srotolarne la struttura ed ottenere una catena di “pezzi di rete” chiamati anche *unità* (figura 6).

Il funzionamento di un’unità ricorrente LSTM ruota tutto intorno ai concetti di *cell state* (figura 7a) e di *gate* (figura 7b), ovvero:

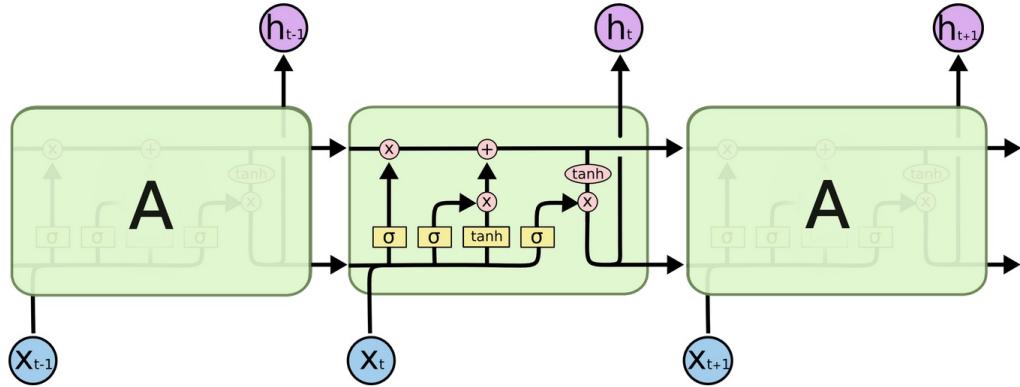


Figura 6: Esempio di “srotolamento” di una rete LSTM. In questa figura ogni unità di rete “A” prende in input al tempo  $t$  un elemento  $X_t$  della sequenza temporale  $X$  e restituisce uno stato in output  $h_t$ .  $\tanh$  e  $\sigma$  sono le funzioni di tangente iperbolica e sigmoidale rispettivamente, mentre “+” e “ $\times$ ” sono rispettivamente le operazioni vettoriali di somma e moltiplicazione.

- cell state: funziona come un nastro trasportatore in grado di memorizzare informazioni, rendendole disponibili durante tutta l’analisi della sequenza;
- gate: sono delle NN più piccole che stabiliscono, attraverso una funzione sigmoidale, quali informazioni possono continuare ad essere memorizzate nel cell state e quali invece devono essere “dimenticate”.

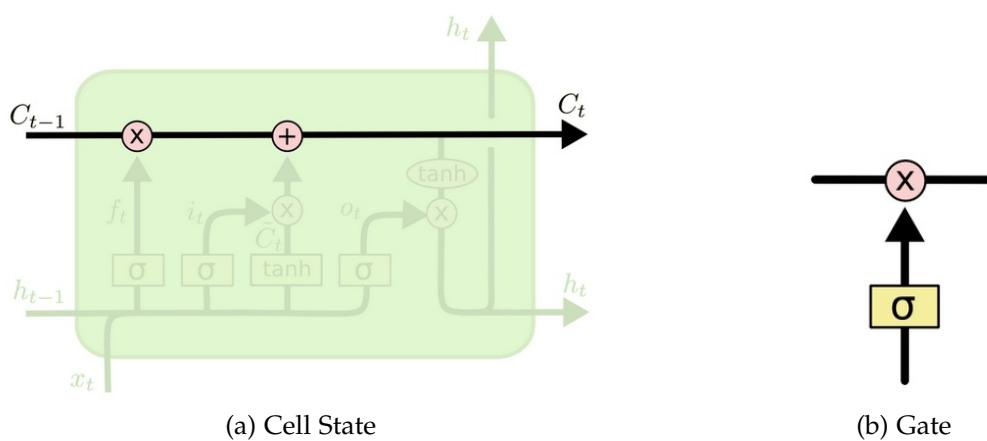


Figura 7: I concetti chiave di una LSTM.

Esistono principalmente tre diverse tipologie di gate che regolano il flusso informativo all'interno di una unità LSTM: *forget gate*, *input gate* e *output gate*.

**Forget gate** - Il forget gate (figura 8) decide quali informazioni mantenere all'interno del cell state e quali invece saranno "dimenticate".

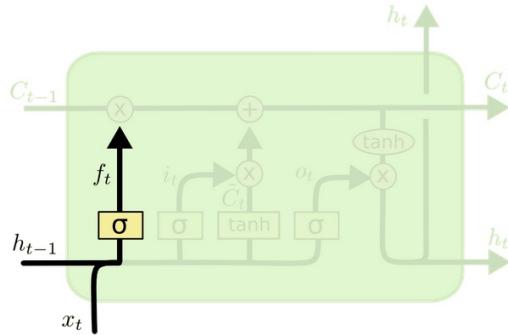


Figura 8: Forget gate di una LSTM

In formula, al t-esimo passo, il forget gate sarà

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

dove:

- $h_{t-1}$  è lo stato, nonché l'output, della LSTM al passo precedente
- $x_t$  è il t-esimo elemento della sequenza in input
- $W_f$  e  $b_f$  sono rispettivamente la matrice dei pesi e il bias del forget gate
- $\sigma$  è la funzione sigmoidale.

**Input gate** - Questo gate decide quali nuove informazioni vogliamo registrare nel nostro cell state e la sua funzione può essere suddivisa in due parti (figura 9a):

1. attraverso un'operazione sigmoidale, seleziona cosa dell'input corrente deve essere mantenuto. In formula,

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i).$$

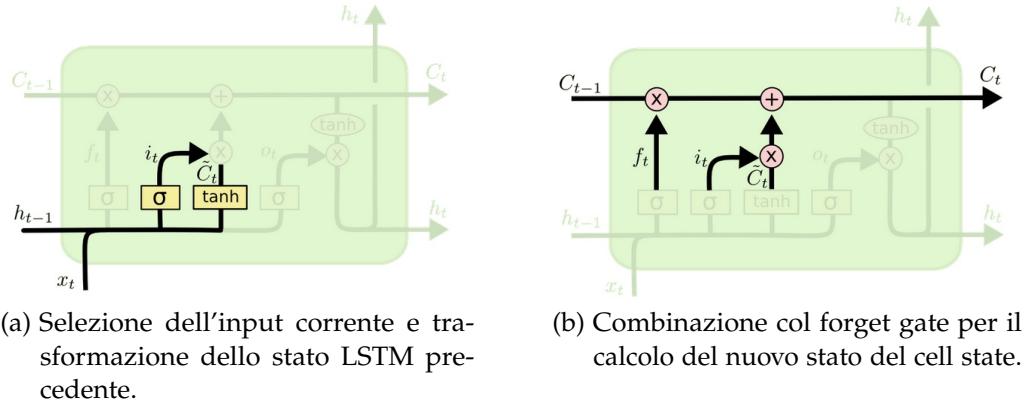


Figura 9: Input gate.

2. trasforma il precedente stato della LSTM con un'operazione  $\tanh$ , ovvero

$$\tilde{C} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

I due vettori risultanti vengono dapprima moltiplicati fra loro e successivamente sommati al cell state combinandoli col forget gate (figura 9b). In formula

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t.$$

**Output gate** - Questo ultimo gate si occupa infine di decidere cosa vogliamo restituire in output ad ogni passo t (figura 10). Il suo compito è quello di combinare una versione filtrata dello stato precedente con una trasformazione del cell state.

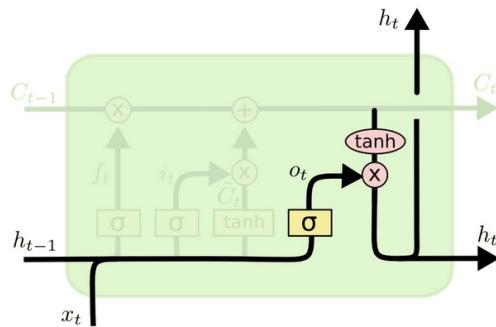


Figura 10: Output gate di una LSTM

L'operazione applicata al filtraggio dello stato precedente è quella sigmoidale, di modo da selezionarne solo i componenti desiderati, ovvero

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

mentre quella applicata al cell state è  $\tanh$  di modo da proiettarne i valori nel range (-1,1), ottenendo

$$h_t = o_t * \tanh(C_t)$$

Il valore  $h_t$  così calcolato sarà l'output al passo t della rete LSTM e l'input del passo successivo, insieme allo stato corrente del cell state  $C_t$ .

Quella appena descritta è una classica LSTM ma in alcuni articoli scientifici è possibile trovare delle leggere varianti, come le "peephole" LSTM [22] o le *Gated Recurrent Unit (GRU)* [23]. Ad ogni modo il concetto alla base di queste varianti resta lo stesso.

# 2

---

## LAVORI PRECEDENTI

---

Il riconoscimento di azioni corporee tramite l’analisi della *posa* umana o *skeleton* (vedremo nel prossimo capitolo più in dettagli di cosa si tratta) sta attraendo recentemente una notevole attenzione nel mondo della visione computerizzata. Il suo successo è senz’altro dovuto non solo agli ottimi risultati ottenuti, ma anche alla sua efficiente semplificazione della struttura umana riducendo di fatto i costi computazionali e le risorse necessarie allo stoccaggio dati. Normalmente la posa viene calcolata utilizzando sensori 2D o 3D posizionati in prossimità dei giunti corporei (gomiti, ginocchi, spalle,...), oppure con algoritmi appositi in grado di estrarla da normali video RGB.

Normalmente ci sono due approcci per il riconoscimento di azioni umane facendo uso dalla posa: gli approcci “*manuali*” e quelli *deep learning*.

Negli approcci *manuali* si cerca di categorizzare le azioni umane seguendo pattern fisici intuitivi, come le posizioni degli arti o le correlazioni temporali fra essi, ad esempio il movimento ondulatorio delle gambe durante una camminata o una corsa.

Per quanto riguarda invece gli approcci *deep learning* le tipologie di azioni corporee vengono categorizzate in maniera automatica direttamente dai dati stessi. Le reti neurali ricorrenti, come le LSTM, o quelle di tipo *convoluzionali-temporali*, come quella in [29], si sono rivelate estremamente efficaci per il riconoscimento di pattern temporali in un video, ottenendo ottimi risultati nella categorizzazione delle azioni.

Esiste infine un terzo approccio che recentemente sta riscuotendo un discreto interesse, ovvero quello basato su *grafi* che punta ad esprimere contemporaneamente sia le relazioni temporali che quelle spaziali dei giunti corporei. Questa tecnica punta a rappresentare la sequenza di pose in un video come un grafo composto da archi temporali e spaziali, considerando cioè rispettivamente le relazioni inter ed intra-frame.

Ad oggi i migliori risultati sulla categorizzazione dei movimenti corporali possono essere riassunti nei seguenti articoli scientifici:

- **Recognizing Human Actions as the Evolution of Pose Estimation Maps** - Mengyuan Liu, Junsong Yuan - 2018 [24]
- **Action Machine: Rethinking Action Recognition in Trimmed Videos** - Jiagang Zhu, Wei Zou, Liang Xu, Yiming Hu, Zheng Zhu, Manyu Chang, Junjie Huang, Guan Huang, Dalong Du - 2018 [25]

Vediamo adesso più nel dettaglio come questi lavori hanno affrontato il problema del riconoscimenti di azioni umane.

## 2.1 RECOGNIZING HUMAN ACTIONS AS EVOLUTION OF POSE ESTIMATION MAPS (2018)

L'idea alla base di questo lavoro è quella di creare delle *mappe di stima della posa* umana dalle quali estrarre delle *heatmap* globali e delle *predizioni di pose* e una volta ottenute queste due componenti, combinare le loro caratteristiche temporali e spaziali di modo da identificare l'azione svolta.

Più precisamente, indicando con  $\mathcal{Y}_k \in \{x, y\}$  le coordinate del k-esimo giunto corporeo, possiamo definire un'intera struttura corporea come  $\mathcal{Y} = \{\mathcal{Y}_1, \dots, \mathcal{Y}_k, \dots, \mathcal{Y}_K\}$ , dove K è il numero totale dei tipi di giunti corporei considerati.

Addestrando un classificatore multiclass  $g_t^k$  per predire la posizione del k-esimo giunto corporeo al passo t, otteniamo per ogni punto  $z$  dell'immagine una mappa di stima relativa al giunto k,

$$B_t^k(\mathcal{Y}_k = z) = g_t^k \left( f_z; \bigcup_{i=1, \dots, K} \psi(z, B_{t-1}^i) \right)$$

dove  $f_z$  è la *color feature* per il punto  $z$ ,  $B_{t-1}^i$  è la mappa stimata da  $g_{t-1}^i$  al passo precedente,  $\bigcup$  è l'operatore di concatenazione vettoriale e  $\psi$  è la funzione di estrazione della *feature* interessata partendo dalla mappa ottenuta al passo precedente. Dopo T passi, le mappe ottenute vengono utilizzate per stimare la posizione dei giunti.

Partendo dalle K mappe così ottenute per ogni frame n del video, ovvero  $\{B_T^{1,n}, \dots, B_T^{K,n}\}$ , vengono calcolate una heatmap  $G_n$  ed una posa  $\mathcal{L}_n$  che rappresentano globalmente la figura umana nell'immagine (figura 11). La heatmap  $G_n$  sarà

$$G_n = \frac{1}{K} \sum_{k=1}^K B_T^{k,n},$$

## 2.1 RECOGNIZING HUMAN ACTIONS AS EVOLUTION OF POSE ESTIMATION MAPS (2018)

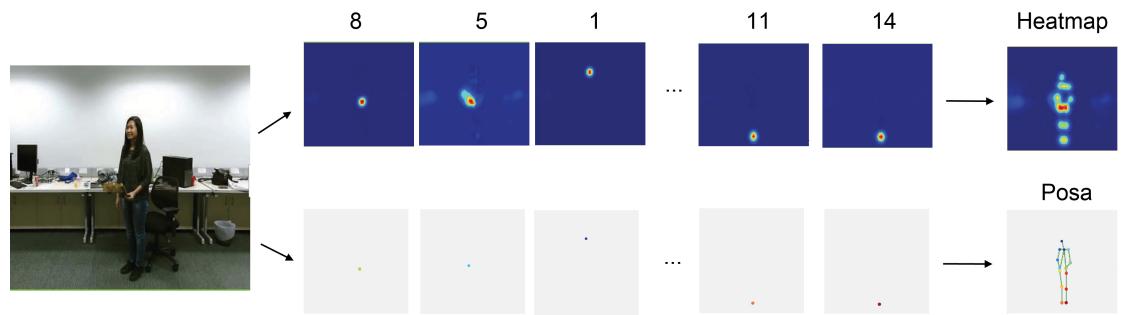


Figura 11: Processo di creazione della heatmap e della posa a partire da un singolo frame del video. Ogni giunto corporeo viene stimato singolarmente per poi essere successivamente combinato agli altri.

mentre la posa  $\mathcal{L}_n$  sarà quell'insieme di punti dell'immagine  $\{z^{k,n}\}_{k=1}^K$  per i quali

$$z^{k,n} = \underset{z \in \mathcal{Z}}{\operatorname{argmax}} \{B_{\gamma}^{k,n}(y_k = z)\},$$

dove  $\mathcal{Z} \in \mathbb{R}^2$  è l'insieme di tutti i punti dell'immagine.

Quello che abbiamo fatto è stato cioè trasformare un video in un sequenza di heatmap e pose, a questo punto dobbiamo però sintetizzarle in un unico elemento rappresentativo di più facile manipolazione.

Gli autori del lavoro mettono a confronto 3 diversi tipi di sintetizzazione:

- **sintetizzazione temporale delle heatmap** - Data una sequenza di  $N$  heatmap  $\mathcal{V}_G = \{G_1, \dots, G_n, \dots, G_N\}$  dove  $G_n \in \mathbb{R}^{P \times Q}$  è quella relativa

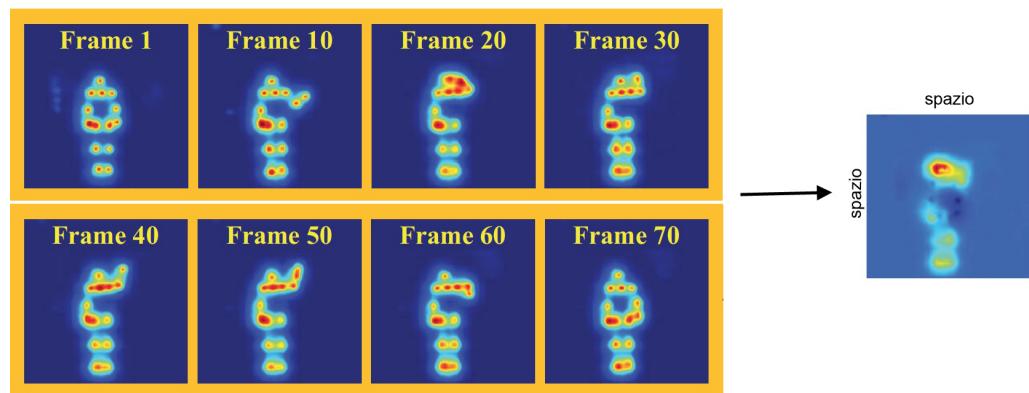


Figura 12: Sintetizzazione temporale delle heatmap.

all' $n$ -esimo frame con  $P$  righe e  $Q$  colonne, allora la sequenza  $\mathbf{G}_{1:n}$  può essere mappata in un vettore

$$\mathbf{v}_n = V\left(\frac{1}{n} \sum_{i=1}^n \mathbf{G}_i\right),$$

dove la funzione  $V$  trasforma una matrice in un vettore  $\mathbf{v}_n \in \mathbb{R}^{(P,Q) \times 1}$  (figura 12).

- **sintetizzazione spazio-temporale delle heatmap** - Partizionando ogni heatmap  $\mathbf{G}_n$  in  $P$  righe, ovvero  $\mathbf{G}_n = [(\mathbf{p}_1^\top, \dots, \mathbf{p}_s^\top, \dots, \mathbf{p}_P^\top)^\top]$  o in  $Q$  colonne, ovvero  $\mathbf{G}_n = [(\mathbf{q}_1, \dots, \mathbf{q}_s, \dots, \mathbf{q}_Q)]$  ed usando la precedente funzione  $V$  per mappare  $(\mathbf{p}_{1:s})^\top$  e  $\mathbf{q}_{1:s}$  in  $\mathbf{v}_s^P$  e  $\mathbf{v}_s^Q$  rispettivamente, si può ottenere due vettori  $\mathbf{u}^P, \mathbf{u}^Q$  rappresentativi del frame attraverso la seguente funzione obiettivo:

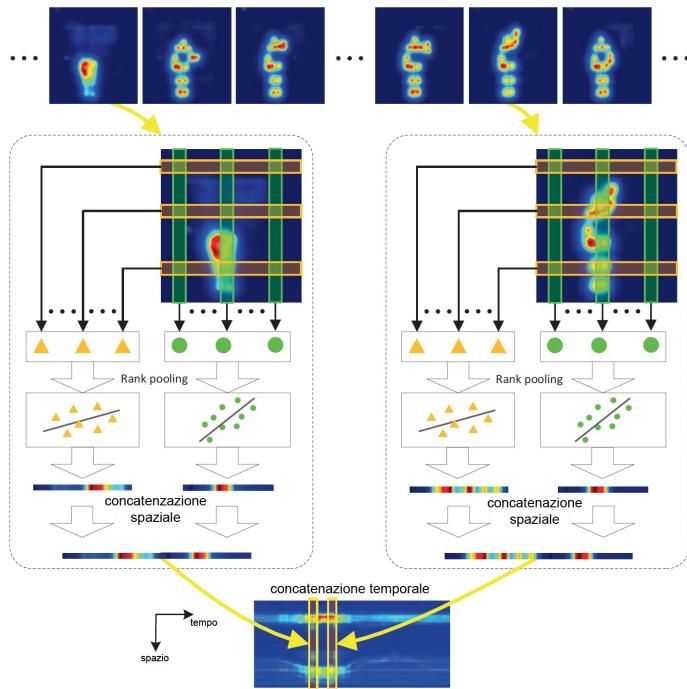


Figura 13: Sintetizzazione spazio-temporale delle heatmap.

$$\begin{aligned} & \arg \min_{\mathbf{u}^n} \frac{1}{2} \|\mathbf{u}^n\|^2 + W \sum_{\forall i,j} \epsilon_{ij}, \\ & \text{s.t. } (\mathbf{u}^n)^\top \cdot (\mathbf{v}_{s_i}^n - \mathbf{v}_{s_j}^n) \geq 1 - \epsilon_{ij} \\ & \epsilon_{ij} \geq 0 \end{aligned}$$

dove  $\eta \in \{\mathbf{p}, \mathbf{q}\}$ ,  $v_{s_i}^\eta \succ v_{s_j}^\eta$  indica la successione temporale fra  $v_{s_i}^\eta$  e  $v_{s_j}^\eta$ ,  $\mathbf{u}^p \in \mathbb{R}^Q$  e  $\mathbf{u}^q \in \mathbb{R}^P$ . Per gli  $N$  frame vengono concatenati in ordine temporale i vettori così calcolati, ottenendo  $\mathbf{U}^p \in \mathbb{R}^{Q \times N}$  e  $\mathbf{U}^q \in \mathbb{R}^{P \times N}$ . La matrice finale  $\mathbf{U}$  sarà l'unione di queste concatenazioni, ovvero  $[(\mathbf{U}^p)^\top, (\mathbf{U}^q)^\top]^\top$ , cioè una matrice che manterrà sia le caratteristiche temporali che quelle spaziali del movimento ripreso nel video. L'intero processo è schematizzato in figura 13.

- **sintetizzazione spazio-temporale delle pose** - Per comprendere come viene creata questa sintetizzazione, supponiamo di avere una sequenza di pose  $\mathcal{V}_L = \{\mathcal{L}_1, \dots, \mathcal{L}_n, \dots, \mathcal{L}_N\}$  dove  $\mathcal{L}_n = \{z^{k,n}\}_{k=1}^K$  e  $z^{k,n} = (x^{k,n}, y^{k,n})$ , ovvero le coordinate orizzontali e verticali del  $k$ -esimo giunto corporeo. Per codificare le caratteristiche di tale sequenza, ogni posa viene trasformata in una successione di distanze fra giunti consecutivi, seguendo l'ordine rappresentato in figura 14. Ripetendo questo processo per ogni frame e concatenando i risultati ottenuti, si ottiene un'immagine rappresentativa delle caratteristiche spazio-temporali dell'evoluzione della posa nel video. L'intero processo è rappresentato in figura 14.

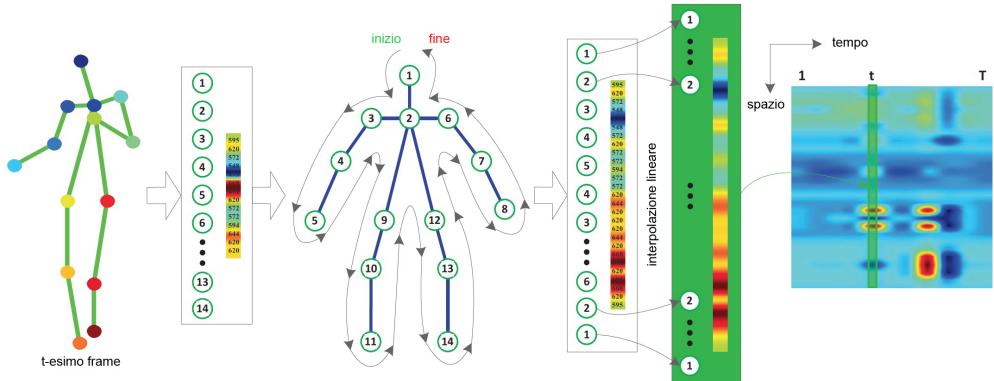


Figura 14: Sintetizzazione spazio-temporale delle heatmap.

Gli esperimenti effettuati dagli autori sono molteplici e una delle caratteristiche principali sulla quale è necessario porre attenzione è la tipologie delle pose utilizzate: in alcuni casi hanno utilizzato delle pose 3D ottenute con delle telecamere di profondità, in altri casi hanno utilizzato pose 2D.

Uno dei dataset utilizzati in questo lavoro è NTU-RGB+D [19] che, come vedremo più in dettaglio nel capitolo 5, propone due metodi di valutazione distinti: *cross-subject* e *cross-view*. I migliori risultati ottenuti sono riassunti in tabella 1.

Pose	Accuratezza cross-view	Accuratezza cross-subject
2D	84.21%	78.80
3D	95.26%	91.71%

Tabella 1: Risultati ottenuti da “Recognizing Human Actions as the Evolution of Pose Estimation Maps - (2018)”

## 2.2 ACTION MACHINE: RETHINKING ACTION RECOGNITION IN TRIMMED VIDEOS (2018)

In questo lavoro gli autori categorizzano le azioni riprese nei video combinando un’analisi sul video e una sulla posa estratta da quest’ultimo.

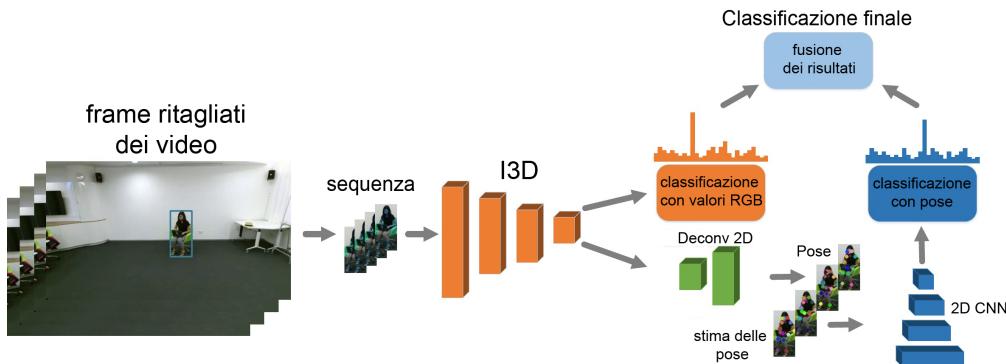


Figura 15: Schematizzazione di *Action Machine*

Come mostrato in figura 15, *Action machine* (questo è il nome che gli autori hanno dato a questo algoritmo) si suddivide in fasi:

1. **Input** - Tutti i video del dataset vengono analizzati con *Deformable CNN* [27], un algoritmo per il riconoscimento di persone in immagini in grado di delinearne la relativa *bounding box*. La threshold in questa fase è stata settata a 0.99 per scartare immediatamente la maggior parte dei falsi positivi riconosciuti. Successivamente viene selezionata la più piccola *bounding box* che contiene tutte le *bounding box* identificate nel video ed utilizzata come maschera di ritaglio per ogni frame. Utilizzare questa tecnica ha un doppio vantaggio: riesce a risolvere molti dei falsi negativi lungo il video e allinea le features sulla dimensione temporale.
2. **Backbone** - I ritagli dei frame vengono passati all’*Inflated 3D ConvNet* (I3D, proposto nel lavoro di J. Carreira e A. Zisserman nel

2017 [26]) implementato con ResNet-50 [13] per l'estrazione delle features. Per aiutare la stima della posa in ogni frame viene rimosso il *temporal max pooling* dopo la prima fase di I<sub>3</sub>D. La mappa delle features ottenute ha dimensione  $2048 \times 8 \times 7 \times 7$  è verrà utilizzata sia per la categorizzazione dell'azione basata solo sui valori RGB che per la stima della posa.

3. **Categorizzazione con valori RGB** - Dopo l'ultimo layer convoluzionale di I<sub>3</sub>D viene effettuato un *global average pooling* per ottenere il vettore delle features composto da 2048 valori, che indicheremo con  $P_{rgb}$ . Se definiamo un dataset di N video con n categorie  $\{(X_i, y_i)\}_{i=1}^N$ , dove  $y_i \in \{1, \dots, n\}$  è l'etichetta corrispondente alle features  $X_i$ , allora la predizione dell'azione può essere ottenuta direttamente come segue:

$$Y_{rgb} = \varphi(W_c P_{rgb} + b_c),$$

dove  $\varphi$  è l'operazione *softmax*,  $Y_{rgb} \in \mathbb{R}^n$  e  $W_c$  e  $b_c$  sono i parametri del livello completamente connesso. In fase di training la loss finale viene combinata alla *cross-entropy loss* ottenendo

$$L_r = - \sum_{i=1}^N \log(Y_{rgb}(y_i)),$$

dove  $Y_{rgb}(y_i)$  è il valore dell' $y_i$ -esima posizione di  $Y_{rgb}$ .

4. **Stima della posa** - Una volta ottenuto l'output dalla I<sub>3</sub>D, la stima della posa viene effettuata sulla dimensione temporale. Ispirandosi a Mask R-CNN [7], viene aggiunta all'ultimo livello della I<sub>3</sub>D una *head* di deconvoluzione 2D, ovvero 2 livelli di deconvoluzione con la normalizzazione dei batch e un'attivazione di tipo RELU. Ogni layer ha 256 filtri con un kernel  $4 \times 4$  ed uno stride di 2. Viene infine aggiunto un livello convoluzionale  $1 \times 1$  per generare la heatmap corrispondente ai K keypoint (un canale per ogni keypoint) e gli offsets (2 canali per ogni keypoint, per le direzioni sull'asse  $x$  e  $y$ ) per un totale di 3K canali in output, dove  $K = 17$ .

Data un'immagine, definiamo  $f_k(x_i) = 1$  se il k-esimo keypoint è situato in posizione  $x_i$ , altrimenti  $f_k(x_i) = 0$ , dove  $i \in 1, \dots, Q$  è l'indice dei pixel nell'immagine. Per ogni punto  $x_i$  e per ogni keypoint k viene calcolata la probabilità  $h_k(x_i) = 1$  per  $\|x_i - l_k\| \leq M$  ovvero la probabilità che il punto  $x_i$  si trovi entro un raggio  $M$

dall'effettivo punto  $l_k$  del keypoint  $k$ . Queste heatmap vengono allenate risolvendo un problema di classificazione binaria per ogni posizione e per ogni keypoint in maniera indipendente. Per ogni posizione  $x_i$  e per ogni keypoint  $k$  viene inoltre stimato il vettore-distanza  $F_k(x_i) = l_k - x_i$  fra  $x_i$  e l'effettivo punto del keypoint corrispondente.

Questa parte dell'algoritmo fornisce quindi le heatmap delle probabilità  $h_k(x_i)$  per ogni keypoint  $k$  e per ogni punto  $x_i$ . Il *training target* per ogni heatmap è  $\bar{h}_k(x_i)$  ovvero una mappa di zeri e uno tale che  $\bar{h}_k(x_i) = 1$  se  $\|x_i - l_k\| \leq M$ , 0 altrimenti. La corrispondente loss function  $L_h(\theta)$  è la somma delle loss L1 per ogni punto e per ogni keypoint in maniera indipendente, ovvero

$$L_h(\theta) = \frac{1}{K} \sum_{k=1}^K \sum_{i=1}^N R(h_k(x_i), \bar{h}_k(x_i)),$$

dove  $R$  è la loss L1.

Per quanto riguarda invece la stima degli offset, quest'ultimi vengono calcolati solo per tutti quei punti  $x_i$  entro un raggio  $M$  da ogni keypoint e le differenze fra i valori stimati e quelli reali vengono penalizzati con una loss L1, ovvero

$$L_o(\theta) = \frac{1}{K} \sum_{k=1}^K \sum_{i: \|l_k - x_i\| \leq M} R(F_k(x_i), (l_k - X_i)).$$

La loss finale per la stima della posa sarà quindi

$$L_p = \lambda_h L_h(\theta) + \lambda_o L_o(\theta),$$

dove  $\lambda_h = 0.5$  e  $\lambda_o = 0.5$ .

In fase di test, viene effettuata un'operazione *argmax* su ognuna della  $K$  heatmap per ottenere la stima del punto corrispondente, ovvero

$$x_k = \arg \max_{x_i} (h_k(x_i), i \in 1, \dots, Q).$$

La posizione accurata del  $k$ -esimo keypoint viene infine ottenuta sommando il relativo offset  $F_k(x_k)$  a  $x_k$ .

5. **Stima dell'azione utilizzando la posa** - Le coordinate 2D delle pose vengono dapprima trasformate in un tensore di dimensione  $2 \times T \times$

$K$ , dove  $T$  è il numero di frame in input e successivamente passate come input a ResNet-18 [13] per la classificazione dell'azione. A causa dell'insufficiente dimensione spaziale di questo input, tutte le operazioni di pooling in ResNet-18 sono state rimosse e tutti gli stride di 2 nei livelli convoluzionali sono stati rimpiazzati con stride di 1. Infine un *global average pooling* al termine di ResNet-18 restituisce in output un vettore di features lungo 512. Per ogni sequenza di pose viene quindi stimata l'azione  $Y_{\text{action}}$  attraverso una cross-entropy loss

$$L_{\text{action}} = - \sum_{i=1}^N \log(Y_{\text{action}}(y_i)).$$

6. **Allenamento multi-task** - Action machine ha 3 obiettivi: classificazione dell'azione utilizzando i valori RGB, stima della posa e classificazione dell'azione utilizzando le sequenze di pose. Questi 3 task sono ottimizzati parallelamente utilizzando la seguente funzione loss

$$L = \lambda_1 L_r + \lambda_2 L_p + \lambda_3 L_{\text{action}},$$

dove  $\lambda_1, \lambda_2$  e  $\lambda_3$  sono dei bilanciatori del peso che vogliamo dare ad ogni task. Gli autori di action machine settano questi bilanciatori tutti a 1.

7. **Fusione dei risultati RGB con quelli della posa** - Nella fase di test, le probabilità provenienti dalla stima con i valori RGB vengono sommate a quelle provenienti dalla stima con le pose, di modo da combinare i punti di forza entrambe le tecniche.

I risultati ottenuti con questo lavoro sono, ad oggi, i migliori su diversi dataset, tra i quali NTU-RGB+D dove Action machine è attualmente il miglior algoritmo con la seguente accuratezza:

- **97.2%** per il criterio *Cross-view*
- **94.3%** per il criterio *Cross-subject*.

### 2.3 OVERVIEW

I migliori risultati fino ad oggi ottenuti utilizzando il dataset NTU-RGB sono schematizzati in tabella 2 ordinati per risultato e suddivisi a seconda delle informazioni utilizzate dato che, come vedremo nel capitolo 5, il

dataset NTU-RGB mette a disposizione diverse tipologie di dati, tra i quali i video RGB delle azioni svolte, le pose dei soggetti inquadrati e le riprese ad infrarossi.

Articolo	Posa	RGB	IR	Accuratezza cross-view (%)	Accuratezza cross-subject (%)
Lie Group [31]	✓			52.8	50.1
H-RNN [32]	✓			64.0	59.1
Deep LSTM [33]	✓			67.3	60.7
PA-LSTM [33]	✓			70.3	62.9
ST-LSTM+TS [34]	✓			77.7	69.2
Temporal Conv [35]	✓			83.1	74.3
DSSCA-SSLM [41]		✓		-	74.9
Skelemotion [45]	✓			84.7	76.5
C-CNN+MTLN [36]	✓			84.8	79.6
VA-LSTM [37]	✓			87.6	79.4
Chained [40]		✓		-	80.8
ST-GCN [38]	✓			88.3	81.5
SR-TSL [39]	✓			92.4	84.8
2D-3D-Softargmax [42]		✓		-	85.5
Glimpse Clouds [43]		✓		93.2	86.6
PB-GCN [46]	✓			93.2	87.5
MFAS [47]	✓	✓		-	90.04
FUSION [28]	✓		✓	94.5	91.6
PoseMap [24]	✓	✓		95.2	91.7
MMTM [44]	✓	✓		-	91.99
Action Machine [25]		✓		97.2	94.3

Tabella 2: Risultati dei lavori precedenti sul dataset NTU-RGB. Le risorse utilizzate sono: *Posa* = skeleton dei soggetti fornita da NTU-RGB; *RGB* = video RGB; *IR* = video ad infrarossi.

# 3

---

## STIMA DELLA POSA

---

Cos è la stima della posa?

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

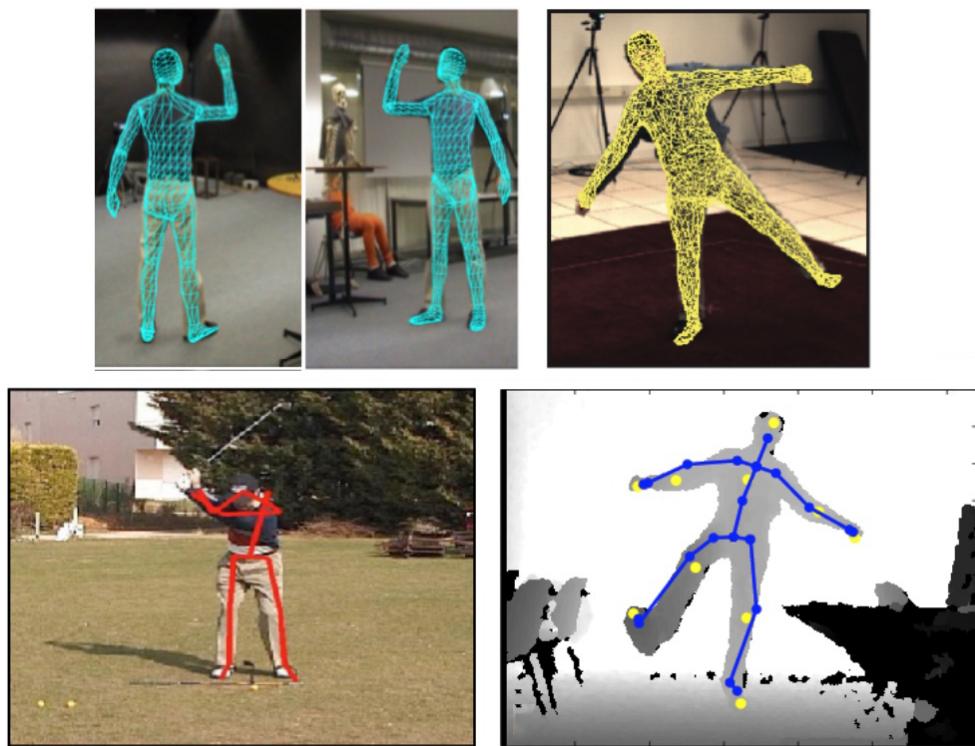


Figura 16: Esempi di stima della posa. In alto tre esempi di stima della posa utilizzando modelli di tipo volumetrico. In basso due esempi di stima della posa ottenuti utilizzando modelli di tipo scheletrici.

Questa tecnica non va assolutamente confusa con tecniche di riconoscimento di persone, infatti la stima della posa è in grado solo di riconoscere dove sono situate le parti del corpo di un individuo 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 estremamente difficile.

Spinti dal crescente interesse degli 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 17: 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.

### 3.1 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, compreso quelli "affollati", ed essere in grado di identificare ogni persona stimandone i suoi *punti chiave* (o *keypoint*).

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

- **Approccio top-down** - l'approccio *top-down* inizia con una fase di identificazione e localizzazione approssimativa della posizione delle persone, delimitando il riquadro dell'immagine entro il quale sono contenute e continua con una fase di stima della posa o di separazione "primo piano-sfondo" nell'area identificata.
- **Approccio bottom-up** - l'approccio *bottom-up* inizia localizzando *entità semantiche individuali*, come ad esempio gambe, braccia, mani, etc, e procede raggruppandole logicamente in istanze di persone complete. *PoseNet* adotta questo secondo approccio.

La rete neurale utilizzata in *PoseNet* è di tipo convoluzionale ed il costo computazionale per il riconoscimento delle pose è essenzialmente indipendente dal numero di persone raffigurate nella scena ma dipende esclusivamente dalla scelta dei filtri 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 questo approccio potrebbe sembrare più "disordinato", i risultati empirici ne hanno dimostrato l'efficacia.

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 dopo essere stato allenato col famoso dataset COCO-2016 [2] (che contiene anche l'annotazione dei 17 keypoint di migliaia di persone), è riuscito a migliorarne l'*AP* (average-precision) da 0,655 a 0,687, diventando il miglior risultato.

Questo metodo essendo molto semplice è anche quindi molto rapido, poiché non richiede alcuna fase supplementare di raffinamento dei risul-

tati 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.

### 3.1.1 Stima dei keypoint

L'obiettivo di questa fase è quello di rilevare, in modo indipendente dall'istanza, tutti i keypoint visibili appartenenti a qualsiasi persona dell'immagine. A tale scopo, per ogni tipologia di keypoint, viengono prodotte delle *heatmap* e degli *offset*.

Sia  $x_i$  la coordinata 2D del pixel  $i$  nell'immagine, dove  $i = 1, \dots, N$  e  $N$  è il numero totale di pixel, allora definiamo con  $D_R(y) = \{x : \|x - y\| \leq R\}$  un disco di raggio  $R$  centrato nel punto  $y$  e  $y_{j,k}$  la coordinata 2D del  $k$ -esimo keypoint della  $j$ -esima persona, con  $j = 1, \dots, M$ , dove  $M$  è il numero di persone nell'immagine.

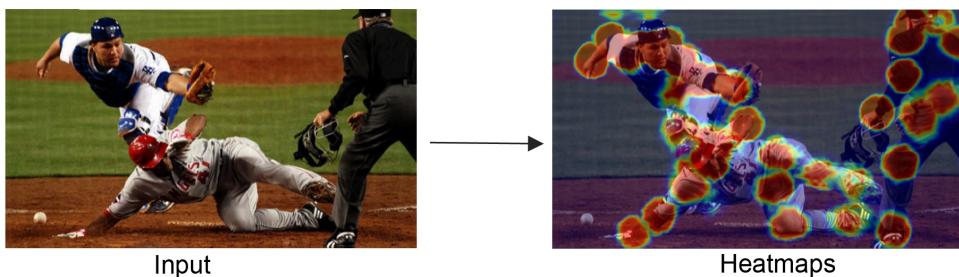


Figura 18: Generazione con PoseNet delle heatmap per ogni tipologia di keypoint

Come abbiamo visto nel capitolo precedente, per ogni tipo di keypoint  $k = 1, \dots, K$ , viene impostato un task di classificazione binaria come segue: la heatmap  $p_k(x) = 1$  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 keypoint) e ciascuno di loro equivale a stimare un disco di raggio  $R$  attorno a un tipo di keypoint specifico appartenente a qualsiasi persona nell'immagine.

Oltre alle heatmap, vengono anche stimati vettori di *offset a corto raggio* (*short-range offset*)  $S_k(x)$  il cui scopo è quello di migliorare l'accuratezza della localizzazione dei keypoint. Per ogni punto  $x$  all'interno dei dischi ricavati al passo precedente, il vettore di offset 2D a corto rag-

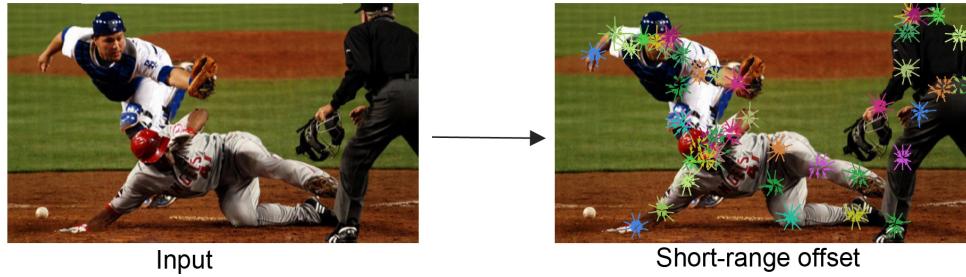


Figura 19: Esempio di stima degli offset a corto raggio con PoseNet

gio  $S_k(x) = y_{j,k} - x$  rappresenta la distanza fra il punto  $x$  e il  $k$ -esimo keypoint della  $j$ -esima persona più vicina. Vengono quindi generati  $K$  vettori per ogni punto  $x$  che verranno successivamente combinati fra loro in una *trasformata di Hough* per migliorare l'accuratezza della posizione stimata di ogni keypoint. Rimandiamo all'articolo [1] per i dettagli d'implementazione.

### 3.1.2 Raggruppamento dei keypoint in istanze di persona

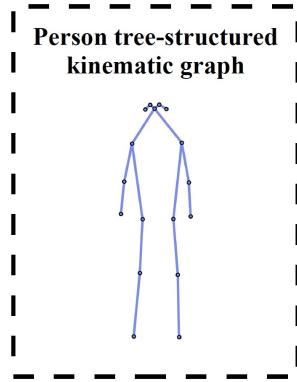


Figura 20: Struttura ad albero utilizzata da PoseNet per raggruppare i keypoint appartenenti alla stessa persona

A questo punto è però necessario capire come associare ogni keypoint alle persone raffigurate nell'immagine (nel caso ce ne sia più di una).

Seguendo lo schema delle connessioni fra tipi di keypoint rappresentati in figura 20 la rete viene allenata per restituire in output anche i cosiddetti *offset a medio raggio* (*mid-range offsets*), ovvero le probabilità di connessioni fra keypoint, col lo scopo di raggruppare quelli appartenenti alla stessa persona.

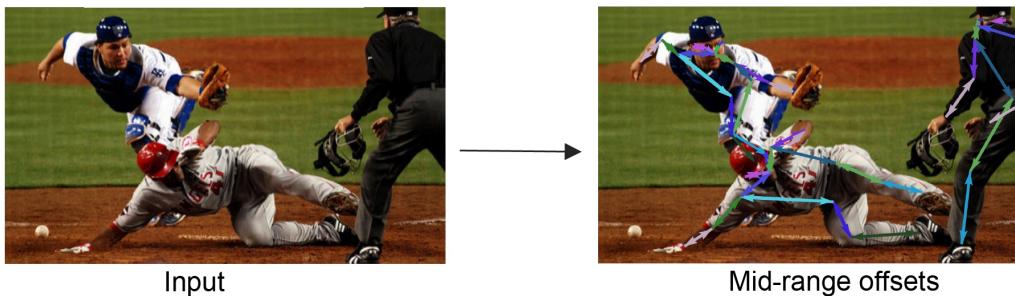


Figura 21: Esempio di stima degli offset a medio raggio con PoseNet. L'intento è quello di raggruppare i keypoint appartenenti alla stessa persona.

Un esempio di questa stima è raffigurato in figura 21 mentre una raffigurazione completa del sistema adottato da PoseNet è rappresentato in figura 22.

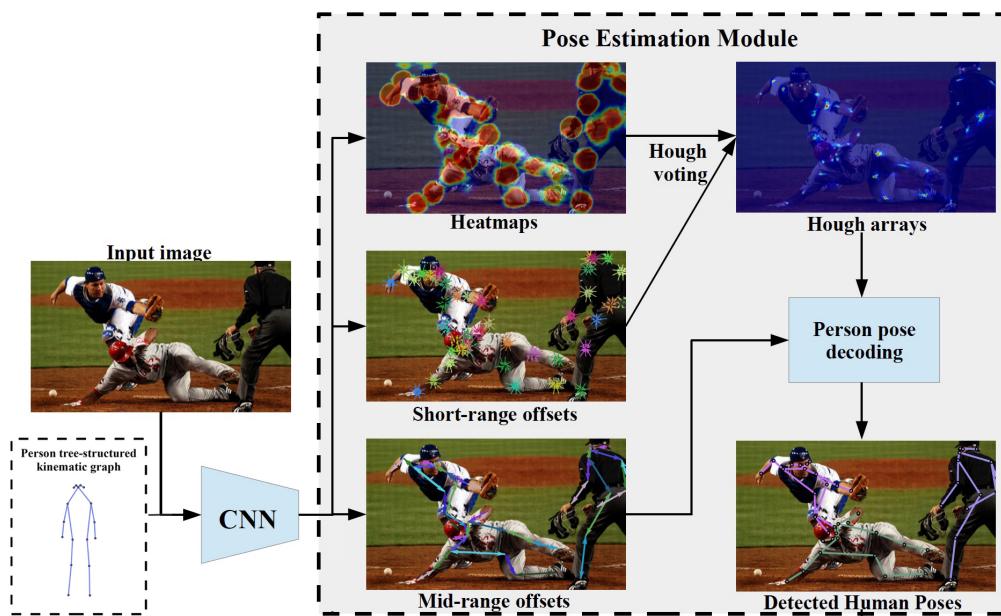


Figura 22: Combinazione delle fasi adottate da PoseNet per il riconoscimento della posa in un'immagine.

### 3.2 DETECTRON-2

Detectron-2 è un progetto open-source lanciato da *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 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'*International Conference on Computer 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.

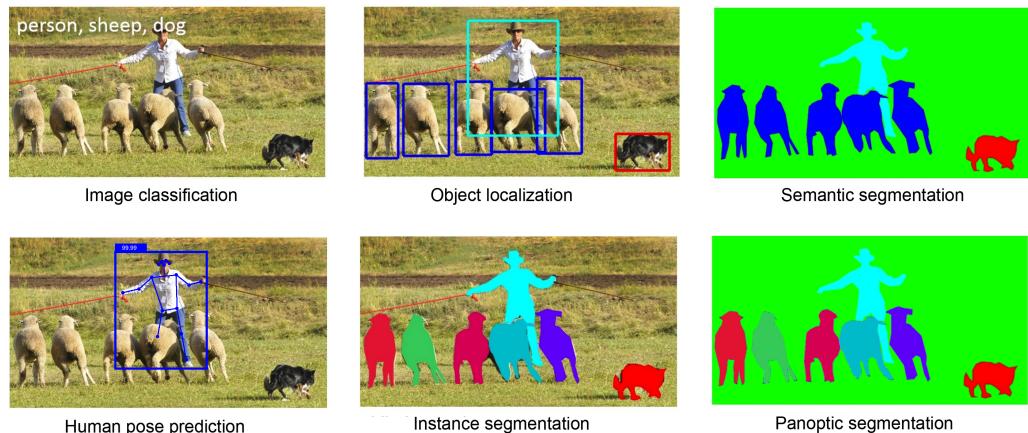


Figura 23: Tipologie di analisi visive.

Detectron-2 è adesso basato su *Pytorch*, una libreria open-source dedicata al machine learning ed ampiamente usata nel campo della computer-vision, che ha inglobato in se anche il precedente framework Caffe2. Più nello specifico Detectron-2 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 grazie alla struttura modulare di Pytorch, che permette di separare facilmente il nuovo modello dai precedenti algoritmi di Detectron-2.

Essendo stato interamente riscritto in Pytorch, Detectron-2 è più rapido del suo predecessore nei compiti di *object-detection*, *instance segmentation* e *human-pose prediction* ed in più è in grado di gestire i nuovi task di *semantic segmentation* e *panoptic segmentation*, ovvero la combinazione fra instance-segmentation e semantic-segmentation (figura 23).

Oltre che nel mondo della ricerca, questa piattaforma viene usata anche per l'addestramento di nuovi modelli in svariati campi della *computer vision*, come ad esempio la *realità 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 invece la stima della posa umana in un'immagine, Detectron-2 utilizza una Mask R-CNN [7] riadattata all'estrazione dei keypoint. Vediamo adesso più nel dettaglio come è stato strutturato questo algoritmo.

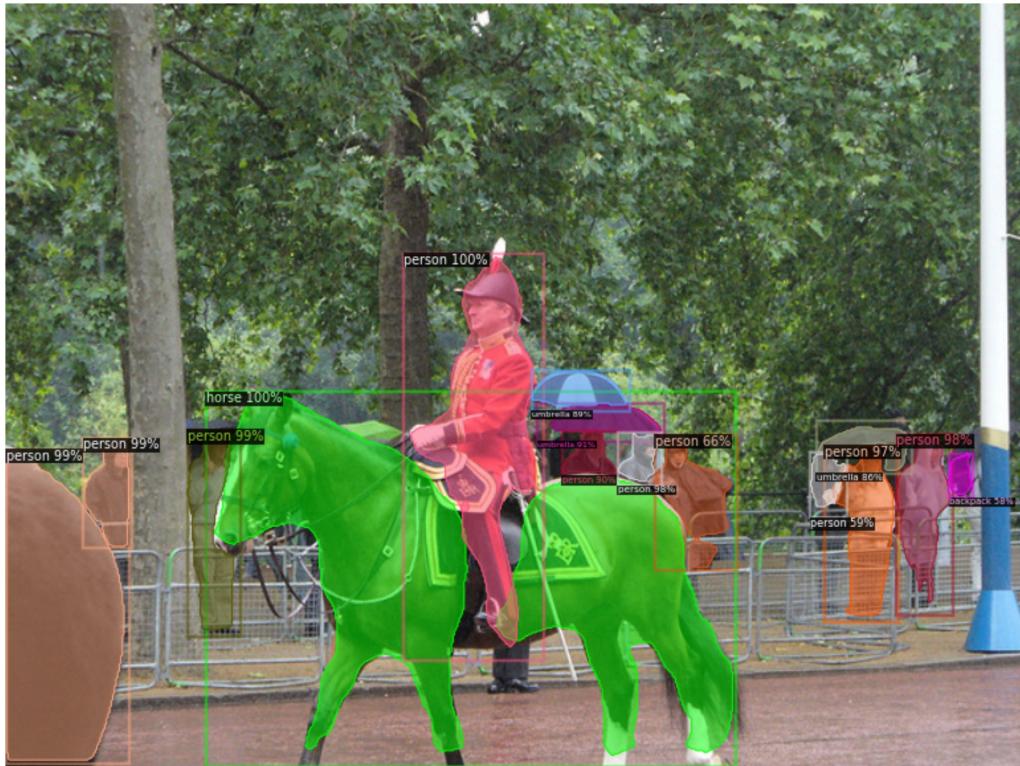


Figura 24: Instance segmentation con Detectron-2

### 3.2.1 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.

La procedura adottata da Mask R-CNN è suddivisa in due fasi. 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 di persona. Nella seconda fase vengono stimate *in parallelo*: la classe, il delineamento preciso (*bounding box*) e la maschera binaria di ogni *regione d'interesse* (RoI)



Figura 25: Alcuni esempi di instance segmentation con Mask R-CNN

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 bounding box sono identiche a quelle definite per Fast R-CNN [10].

Per quanto riguarda invece  $L_{mask}$ , decodifica  $K$  maschere binarie di dimensione  $m \times m$ , dove  $K$  è il numero di classi, quindi la sua dimensione sarà  $Km^2$  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.

### 3.2.2 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.

Le coordinate dei keypoint 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



Figura 26: Alcuni esempi di pose prediction con Mask R-CNN

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 L<sub>2</sub> (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.



# 4

---

## METODO PROPOSTO

---

In questo capitolo vedremo sia quali sono state le tecniche adottate in questo lavoro di tesi che le decisioni prese durante l'iter operativo.

Come vedremo nel capitolo seguente, il dataset utilizzato è NTU-RGB+D [19] e nonostante esso fornisca anche i dati relativi alla posa dei soggetti inquadrati (quello che viene spesso indicato come *skeleton*) abbiamo preferito non farne assolutamente uso ed utilizzare invece i soli video RGB, dai quali successivamente stimare la posa e servirci di quest'ultima per il riconoscimento delle azioni svolte. Questa scelta è stata presa con lo scopo di favorire una maggiore generalizzazione e applicazione dell'algoritmo, adoperabile in contesti più diffusi dove la stima della posa non è data, ma abbiamo a disposizione solo un comunissimo video RGB.

Passiamo adesso ad analizzare una ad una le varie fasi affrontate nel lavoro di tesi.

### 4.1 ESTRAZIONE DELLE POSE

Per prima cosa è necessario convertire tutti i video del dataset in sequenze di pose, in modo da sintetizzarne efficacemente il movimento umano senza un'eccessiva perdita d'informazione. Gli algoritmi scelti per questa fase sono stati Detectron-2 [4] per la sua rimarcabile accuratezza e PoseNet [1] per la sua velocità d'inferenza e portabilità. I modelli proposti da queste due tecniche sono molteplici. Per Detectron-2 abbiamo:

- $R50-FPN-1x$
- $R50-FPN-3x$
- $R101-FPN-3x$
- $X101-FPN-3x$

mentre per PoseNet abbiamo:

- *PoseNet-50*
- *PoseNet-75*
- *PoseNet-100*
- *PoseNet-101*

Ognuno di questi modelli ha valori diversi in accuratezza e velocità di inferenza. Nel caso di Detectron-2 questa comparazione viene fornita dagli autori stessi e schematizzata in tabella 3, mentre per quanto riguarda PoseNet non viene purtroppo fornito un simile confronto.

Nome	Tempo di inferenza (s/imm)	Box AP	KP AP
R50-FPN-1x	0.072	53.6	64.0
R50-FPN-3x	0.066	55.4	65.5
R101-FPN-3x	0.076	56.4	66.1
<b>X101-FPN-3x</b>	0.121	57.3	66.0

Tabella 3: Diversi modelli in Detectron-2 per l'estrazione della posa da un'immagine. Il tempo di inferenza viene misurato in *secondi/immagine*; Box AP = "Bounding box average precision" ; KP AP = "Keypoint average precision"

È stato quindi scelto il modello *X101-FPN* per Detectron-2, in quanto quello con il miglior valore di accuratezza globale e PoseNet-101, per una più facile comparazione dei risultati.

#### 4.2 ASSEGNAZIONE COERENTE DELLE POSE

Quello che otteniamo per ogni video dalla fase precedente è una sequenza di pose umane estratte da ogni frame. Sia Detectron-2 che PoseNet sono in grado di riconoscere più persone all'interno di un'immagine, producendo una lista di pose  $P_1, P_2, \dots, P_n$  ordinata secondo le regole dell'algoritmo, ad esempio Detectron-2 le ordina per valore di *score* decrescente, ovvero quel valore che indica quanto è sicuro l'algoritmo di aver correttamente stimato la posa di quella persona.

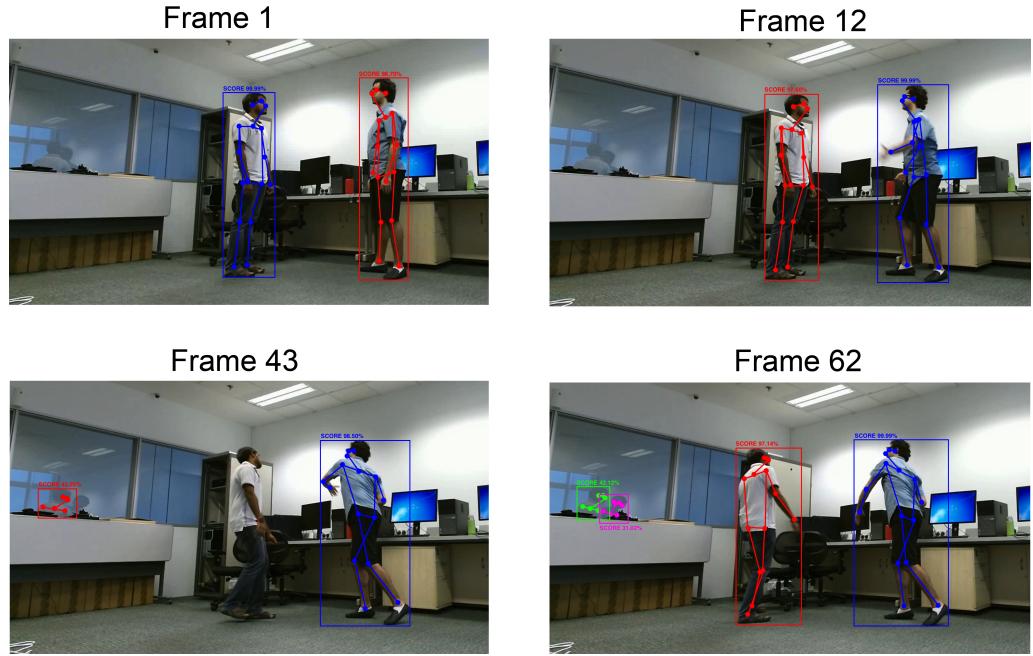


Figura 27: Un esempio di assegnazione inconsistente delle pose lungo il video. Nell'esempio, l'ordine delle pose restituite dall'algoritmo è blu, rosso, verde e viola. Quest'ordine è indipendente dai soggetti inquadrati, che possono talvolta essere scambiati fra loro, non essere riconosciuti o addirittura scambiati per dei riflessi.

Quello che però questi algoritmi non hanno è l'informazione relativa al concetto di video, ovvero alla successione logica dei frame e trattano ognuno di essi come un'immagine indipendente. Quello che ne segue è una totale indipendenza fra l'ordine delle pose riconosciute e la loro assegnazione coerente coi soggetti del video. Ad esempio, ammettiamo di avere due soggetti A e B e che per l' $i$ -esimo frame vengano identificate due pose  $\mathcal{P}_{1,i}$  e  $\mathcal{P}_{2,i}$  appartenenti ad A e B rispettivamente. Quest'ordine non è però garantito nel frame successivo dove  $\mathcal{P}_{1,i+1}$  potrebbe essere la posa di B e  $\mathcal{P}_{2,i+1}$  quella di A.

Per ovviare a questo tipo di problema, la tecnica adottata è stata quella di calcolare iterativamente ad ogni frame la distanza fra l'ultima posa assegnata ad ogni soggetto e tutte quelle riconosciute al frame successivo per assegnare poi la posa più corretta. Nel dettaglio, definiamo la distanza fra due pose come la somma delle distanze euclidee fra i loro punti in comune, ovvero se  $\mathcal{P}_i = \{p_{i,k}\}_{k=1}^K$  è l'insieme dei punti che definiscono la

posa  $i$ , dove ogni  $p_{i,k} = (x_{i,k}, y_{i,k})$  è la coordinata del  $k$ -esimo giunto e  $K$  il numero di tipi di giunto, allora la distanza fra  $P_i$  e  $P_j$  sarà:

$$\mathcal{D}_{ij} = \sum_{k=1}^K I(p_{i,k}) \cdot I(p_{j,k}) \cdot \text{eucl}(p_{i,k}, p_{j,k})$$

dove  $I$  è la funzione identità, ovvero  $I(p) = 1$  se  $p$  è definito, 0 altrimenti.  $\text{eucl}(p_1, p_2)$  è la funzione di distanza euclidea fra i punti  $p_1$  e  $p_2$ . Se le due pose non hanno punti in comune, ovvero  $\sum_{k=1}^K I(p_{i,k}) \cdot I(p_{j,k}) = 0$  allora  $\mathcal{D}_{ij} = \infty$ .

Una volta ottenuta la lista delle distanze fra pose si può procedere al loro abbinamento, assegnando ad ogni soggetto la posa più vicina e ripetere così il procedimento per il frame successivo.

Abbiamo però un ultimo problema da fronteggiare in questa fase, ovvero la variabilità del numero di pose stimate in ogni frame del video. Questo accade quando per uno o più frame, uno dei soggetti non viene riconosciuto dall'algoritmo o quando l'improvviso riflesso su una finestra o uno specchio viene interpretato come una nuova persona (figura 27).

Questo problema è stato risolto selezionando solo le pose con il miglior *score medio* lungo tutto il video, si presuppone infatti che un riflesso sia mediamente meno convincente di una persona vera e propria. Più precisamente sono state mantenute solo le migliori due pose e scartate le altre, visto che in questo lavoro di tesi siamo interessati ad analizzare solo azioni individuali o di coppia.

### 4.3 RIMOZIONE DEGLI ZERI

Come abbiamo appena visto, PoseNet e Detectron-2 talvolta in qualche frame possono non riconoscere un'intera persona o non riuscire a stimare la posizione di qualche giunto, se ad esempio coperti o sfocati. Entrambi gli algoritmi trattano questi due casi nella stessa maniera, ovvero assegnano il punto  $(0, 0)$  ad ogni giunto non riconosciuto.

Questo valore però è fortemente fuorviante considerando che le informazioni a nostra disposizione sono solo pochi punti corporei. Detectron-2 e PoseNet stimano una posa di soli 17 punti, quindi per ogni giunto non riconosciuto, non solo perdiamo  $1/17$  dell'informazione ma confonderemo anche il nostro algoritmo in fase di classificazione dell'azione facendogli analizzare per quel giunto un movimento inesistente verso il punto  $(0,0)$ . È necessario quindi adottare una buona tecnica di rimozione degli zeri,

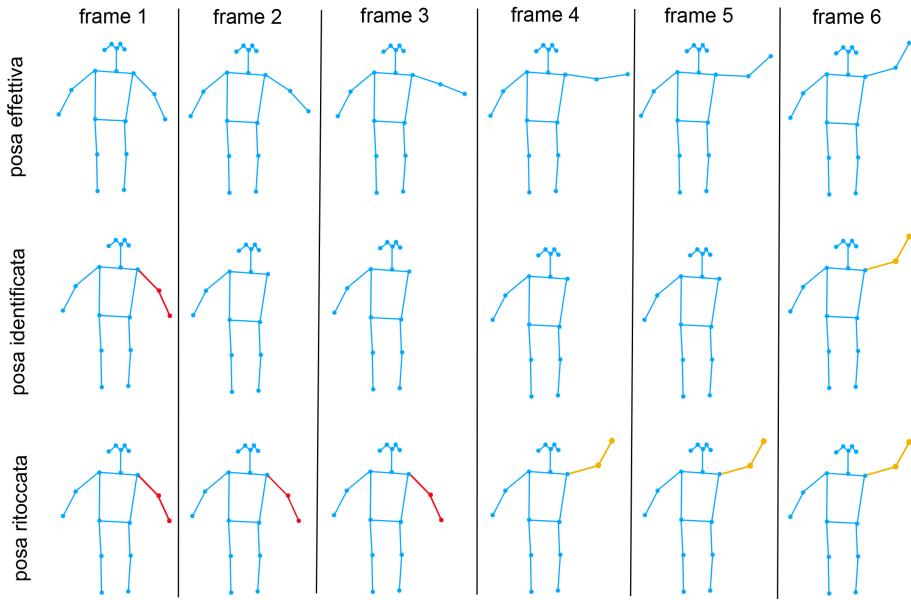


Figura 28: Esempio di rimozione degli zeri. Nei frame 2-3-4-5 i giunti del braccio sinistro non sono stati identificati. Nei frame 2-3 verranno quindi rimpiazzati con quelli del frame 1, mentre nei frame 4-5 con quelli del frame 6.

che limiti la perdita d'informazione, mantenendo al contempo coerenza con le pose stimate.

La cosa più semplice da fare sarebbe assegnare ad ogni giunto non riconosciuto la sua ultima posizione identificata, ma così facendo utilizzerebbero solo il passato del video in questione e per lunghe sequenze di valori mancanti potremmo perdere molta informazione utile. È stato quindi scelto di attribuire ad ogni giunto non identificato il primo punto riconosciuto più vicino nel tempo, passato o futuro, per quel giunto.

L'intento di questa tecnica è quello di ridurre la distanza fra ogni giunto non identificato e la sua effettiva posizione nell'immagine. Un esempio di questa tecnica è rappresentato in figura 28

#### 4.4 TECNICHE DI RIELABORAZIONE

Adesso che siamo riusciti ad ottenere delle sequenze di pose coerenti e complete possiamo procedere alla loro classificazione.

Potremmo passare i valori fin qui ottenuti ad una rete neurale, senza alcuna lavorazione aggiuntiva, ma come è facilmente prevedibile questo porterà a risultati non eccellenti.

In questo lavoro di tesi sono state analizzate una serie di tecniche di rielaborazione delle sequenze di pose, al fine di capire quale fra queste aiuta maggiormente la loro classificazione.

Nelle sezioni seguenti indicheremo con  $K$  il numero totale di giunti di ogni posa (come abbiamo visto, sia per PoseNet che per Detectron-2,  $K = 17$ ), con  $\mathcal{P}_f$  la posa estratta dal frame  $f$ , dove  $\mathcal{P}_f = \{\mathbf{p}_{kf}\}_{k=1}^K$  è l'insieme dei punti nell'immagine corrispondenti ai giunti  $k$ , ovvero  $\mathbf{p}_{kf} = (x_{kf}, y_{kf})$ .

Vediamo adesso più nel dettaglio le tecniche testate.

#### 4.4.1 Baricentro del frame

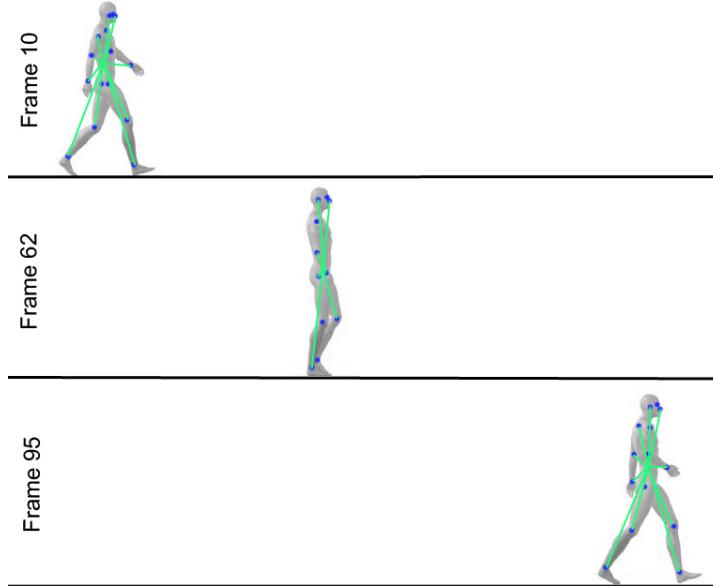


Figura 29: Esempio di applicazione della *tecnica del baricentro del frame*. In figura sono rappresentati 3 frame del video di una camminata da sinistra a destra. Ogni punto blu (relativo ad un giunto corporeo) viene sostituito col vettore-distanza (in verde) fra quel punto e il baricentro della posa.

Applicando questa tecnica, ogni posa  $\mathcal{P}_f$  è stata sostituita con l'insieme  $\mathcal{T}_f = \{\mathbf{t}_{1f}, \dots, \mathbf{t}_{Kf}\}$ , dove ogni  $\mathbf{t}_{if}$  è il *vettore-distanza* fra  $\mathbf{p}_{if}$  e il baricentro di  $\mathcal{P}_f$ , indicato con  $\overline{\mathcal{P}}_f$ , ovvero

$$\mathbf{t}_{if} = \mathbf{p}_{if} - \overline{\mathcal{P}}_f \quad \text{e} \quad \overline{\mathcal{P}}_f = \frac{\sum_{j=1}^K \mathbf{p}_{jf}}{K}.$$

Questa tecnica ha il vantaggio di relativizzare la posa ovunque essa si trovi all'interno del frame e sintetizzarne efficientemente il movimento circoscritto al frame. Allo stesso tempo però, utilizzando un punto di vista sempre e solo focalizzato sul singolo frame, questa tecnica potrebbe non riuscire a mappare correttamente il movimento globale della posa lungo tutto il video. Un esempio di questa tecnica è schematizzato in figura 29.

Nel caso di più persone nel video, ovvero quando analiziamo azioni di coppia, questa tecnica stima un unico baricentro per ogni frame, adattando le due pose di conseguenza. A tal proposito è stata quindi testata anche una seconda variante, ovvero quella del *baricentro del frame per persona*, dove cioè viene settato un baricentro diverso per ogni persona rappresentata nel frame.

#### 4.4.2 Baricentro del video

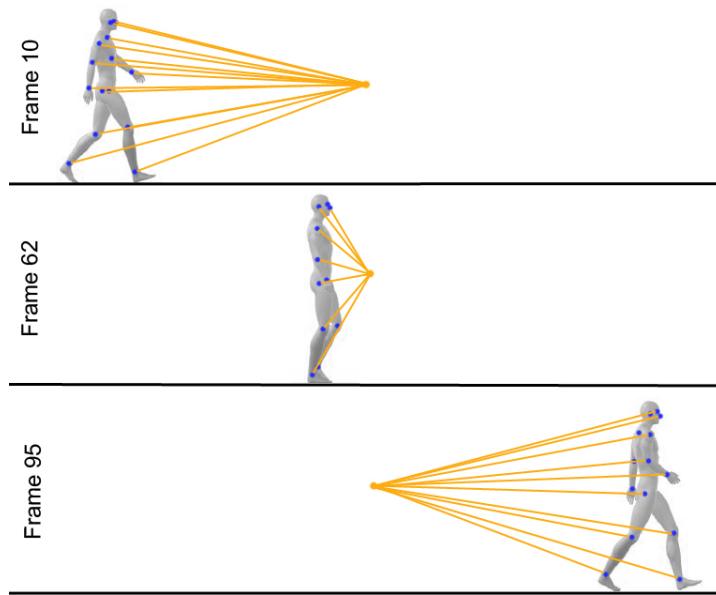


Figura 30: Esempio di applicazione della *tecnica del baricentro del video*. In figura sono rappresentati 3 frame del video di una camminata da sinistra a destra. Ogni punto blu (relativo ad un giunto corporeo) viene sostituito col vettore-distanza (in arancione) fra quel punto e il baricentro del video.

Questa tecnica, molto simile alla precedente, utilizza però come baricentro quello relativo all'intero video anziché quello di ogni frame ovvero,

se indichiamo con  $\mathcal{V} = [F_1, \dots, F_f, \dots, F_L]$  la sequenza di frame del video, allora il baricentro del video  $\bar{\mathcal{V}}$  sarà

$$\bar{\mathcal{V}} = \frac{\sum_{f=1}^L \overline{\mathcal{P}_f}}{L}.$$

Come per la tecnica precedente, anche qui sostituiremo ogni posa con l'insieme dei vettori-distanze dal baricentro, ovvero  $\mathcal{T}_f = \{t_{1f}, \dots, t_{Kf}\}$  dove  $t_{if} = p_{if} - \bar{\mathcal{V}}$ .

Il vantaggio aggiuntivo di questa tecnica rispetto alla precedente è la capacità di saper immagazzinare anche l'informazione sul movimento globale della posa lungo tutto il video. Un esempio di questa tecnica è rappresentato in figura 30.

Come per la tecnica precedente, anche per questa è stata testata la variante *baricentro del video per persona*, dove viene definito un baricentro diverso per ogni persona rappresentata nel video.

#### 4.4.3 Baricentri multipli

Come vedremo nel capitolo 6, la tecnica del *baricentro del video* avrà risultati leggermente migliori rispetto a quella del *baricentro del frame*. La domanda che a questo punto sorge spontanea è se aumentando il numero dei baricentri si ottenga un miglioramento dei risultati.

Se suddividiamo il corpo umano in  $B$  gruppi e definiamo ogni giunto corporeo come appartenente ad uno (e uno solo) di questi gruppi, allora possiamo trasformare ogni posa in un insieme di vettori-distanza fra ogni giunto ed il baricentro del gruppo corrispondente.

Più precisamente, se suddividiamo l'insieme  $\mathcal{K} = \{\mathcal{K}_1, \dots, \mathcal{K}_B\}$  tale che:

- $\bigcup_{b=1}^B \mathcal{K}_b = \mathcal{K}$
- $\mathcal{K}_i \cap \mathcal{K}_j = \emptyset, \quad \forall \mathcal{K}_i, \mathcal{K}_j \in \mathcal{K}$

allora ogni posa  $\mathcal{P}_f$  può essere sostituita con la sua trasformazione  $\mathcal{T}_f = \{t_{1f}, \dots, t_{Kf}\}$  dove, come per le tecniche precedenti, ogni  $t_{if} = p_{if} - \bar{\mathcal{B}}_b$  e

$$\bar{\mathcal{B}}_b = \frac{\bigcup_{i \in \mathcal{B}_b} \sum_{f=1}^L \mathcal{P}_{if}}{L}.$$

In questo lavoro di tesi sono state testate 3 varianti di questa tecnica, ovvero:

- **B=3** - testa, busto e gambe

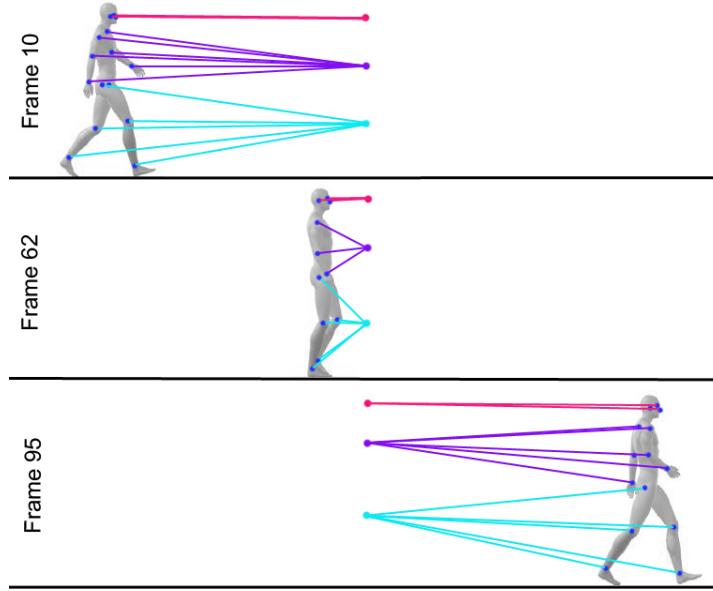


Figura 31: Esempio di applicazione della *tecnica dei 3 baricentri*. In figura sono rappresentati 3 frame del video di una camminata da sinistra a destra. Ogni punto blu (relativo ad un giunto corporeo) viene sostituito col vettore-distanza fra quel punto e il baricentro al quale quel giunto appartiene.

- **B=5** - tronco, braccio sinistro, braccio destro, gamba sinistra e gamba destra (questa suddivisione viene suggerita dagli autori del dataset NTU-RGB+D col modello "Part-Aware LSTM" [19] )
- **B=17** - un baricentro per ogni giunto

Queste 3 diverse tecniche hanno il vantaggio di frammentare la posa, attribuendo valori più alti alle componenti più dinamiche del video, ovvero quelle componenti che si presuppone caratterizzino maggiormente l'azione ripresa nel video. In figura 31 è rappresentato un esempio della tecnica dei 3 baricentri.

Considerando che un'azione può essere svolta in maniera speculare (ad esempio una camminata può essere fatta sia da sinistra a destra che viceversa), ognuna di queste tecniche è stata testata anche considerando solo i valori assoluti dei vettori ottenuti così da astrarre il modello dalla direzione verso la quale viene svolta l'azione. Nei risultati esposti nel capitolo 6 questa variante verrà indicata col suffisso "ASS".

#### 4.4.4 *Tecnica "Next frame"*

Un altro approccio nella classificazione delle sequenze di pose è quello di concentrarci su ciò che accade fra un frame e l'altro, dato che è proprio in questo lasso di tempo che abbiamo il movimento del soggetto o dei soggetti inquadrati.

Seguendo questa idea trasformeremo tutte le nostre pose in un insieme di vettori-distanza fra la posizione dei giunti al frame  $f$  e quelli al frame  $f + 1$ . La nostra posa trasformata  $\mathcal{T}_f$  sarà quindi,  $\mathcal{T}_f = \mathcal{P}_{f+1} - \mathcal{P}_f$ , ovvero

$$\mathcal{T}_f = \bigcup_{k \in \mathcal{K}} p_{k,f+1} - p_{k,f}$$

Un esempio visivo di questa tecnica è rappresentato in figura 32.

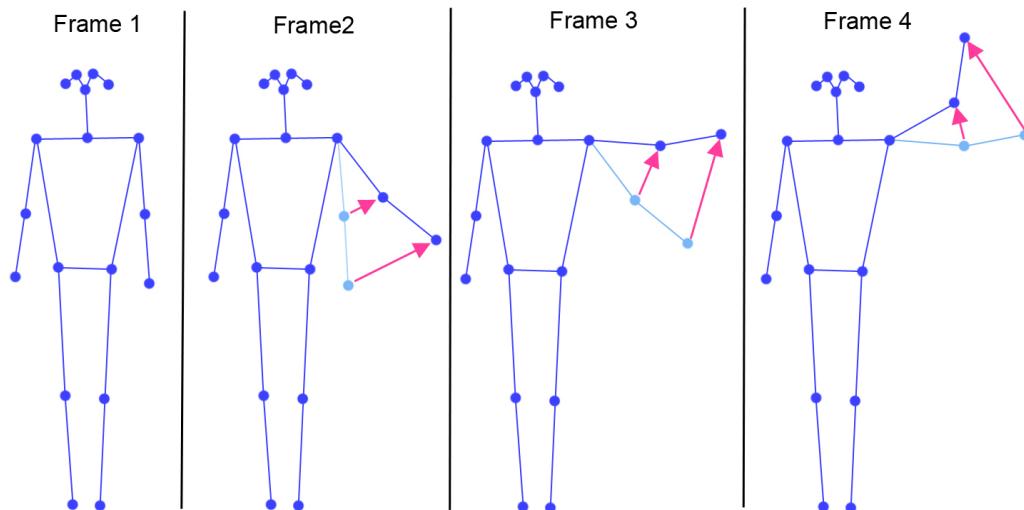


Figura 32: Tecnica del *next frame*. Nell'esempio, il soggetto ripreso alza il braccio sinistro. Ogni posa viene sostituita con l'insieme dei vettori-distanza (frecce rosa) fra ogni giunto e la posizione dello stesso al frame successivo.

Questa tecnica analizza cioè le differenze di posizione dei giunti fra frame consecutivi, questo però porta spesso a dover memorizzare variazioni minime, specialmente se il nostro video ha un FPS (Frame Per Second) elevato. Inoltre potrebbe non essere semplice distinguere le effettive variazioni di movimento dalle approssimazioni derivanti dall'algoritmo di riconoscimento della posa.

Quello che allora possiamo fare è memorizzare le differenze dei giunti fra frame a distanza  $S > 1$ , ovvero

$$\mathcal{T}_f = \bigcup_{k \in \mathcal{K}} p_{k,f+S} - p_{k,f}$$

In questo lavoro di tesi sono stati testati i valori di  $S = \{1, 3, 7, 15\}$  per capire se e quali miglioramenti porta questa variazione di distanza.

#### 4.4.5 *Tecnica delle distanze cumulate*

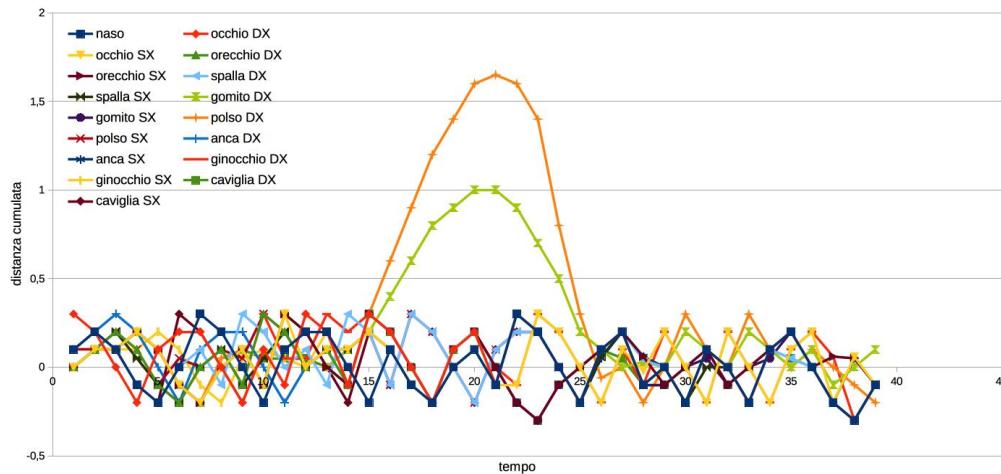


Figura 33: *Tecnica delle distanze cumulate*. Nell'esempio sono raffigurati i valori che questa tecnica restituirebbe per l'azione "alzare e abbassare il braccio destro". Ovviamente i valori più alti sono legati al polso e al gomito destro visto che sono le componenti più dinamiche di questo movimento.

Come abbiamo visto nella tecnica precedente, la distinzione fra valori spesso troppo piccoli e il rumore derivante dall'approssimazione della posizione del giunto non è un compito semplice. Per tentare di ovviare a questo problema, un'altra tecnica testata è quella delle *distanze cumulate*, ovvero trasformare ogni posa  $\mathcal{P}_f$  nella somma dei vettori distanza calcolati fino a quel punto con la tecnica del *next frame*, ovvero

$$\mathcal{T}_f = \bigcup_{k \in \mathcal{K}} \sum_{j=1}^{f-1} p_{k,j+1} - p_{k,j}$$

Quello che ci aspettiamo di ottenere sono variazioni evidenti per i giunti più dinamici nel video, mentre per quelli statici, che quindi si pre-

suppone non caratterizzino eccessivamente il movimento, ci aspettiamo variazioni minori. In figura 33 abbiamo una schematizzazione di ciò che otterremmo con la tecnica delle distanze cumulate se analizzassimo una porzione di video dove il soggetto inquadrato alza e abbassa il braccio destro.

#### 4.4.6 *Tecnica delle distanze relative*

Questa tecnica riprende quella del *centro del frame* con la differenza che ogni posa viene trasformata in una matrice di distanze fra ogni coppia di giunti. Più precisamente per ogni posa  $\mathcal{P}_f$ , che ricordiamo essere appartenente all'insieme  $\mathbb{R}^{K \times 2}$ , otterremo la rispettiva trasformazione  $\mathcal{T}_f \in \mathbb{R}^{K \times K}$ , composta dai valori

$$t_{ij} = \text{eucl}(p_{if}, p_{jf})$$

con  $i = 0, \dots, K$ ,  $j = 0, \dots, K$  ed  $\text{eucl}(a, b)$  la funzione della distanza euclidea fra i punti  $a$  e  $b$ .

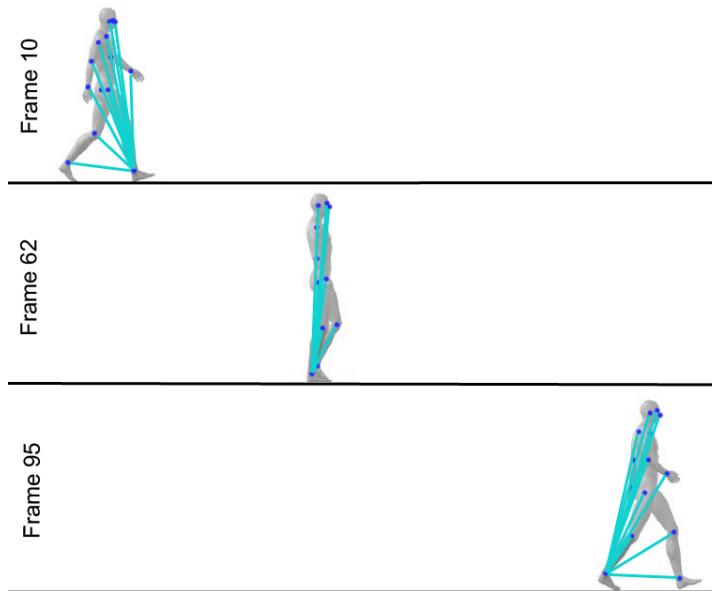


Figura 34: Tecnica delle *distanze relative*. In questa figura sono rappresentati solo i vettori-distanza dalla caviglia destra a tutti gli altri giunti.

Questa tecnica, seppur focalizzata sul singolo frame, ha lo scopo di estrapolare dalla posa anche quei collegamenti fra giunti corporei distanti fra loro, come ad esempio la correlazione fra il movimento delle braccia e quello delle gambe durante una camminata. In figura 34 vediamo un esempio della tecnica appena descritta.

#### 4.5 NORMALIZZAZIONE

Ognuna delle tecniche appena viste ha risultati differenti l'uno dall'altro, con intervalli e variabilità dei valori estremamente diversi per ognuna di esse, rendendo quindi necessaria una normalizzazione dei dati prima di passarli alla rete neurale.

La normalizzazione dei risultati svolge in ogni algoritmo di machine learning un ruolo estremamente importante ed è per questo che anche in questo lavoro di tesi le è stata data particolare attenzione.

Nel nostro caso la maggior parte delle tecniche adottate restituiscono in output un dataset di vettori, ad esempio le coordinate di punti sul piano o distanze vettoriali, mentre qualche tecnica, ad esempio quella delle *distanze relative*, restituisce un dataset di valori semplici.

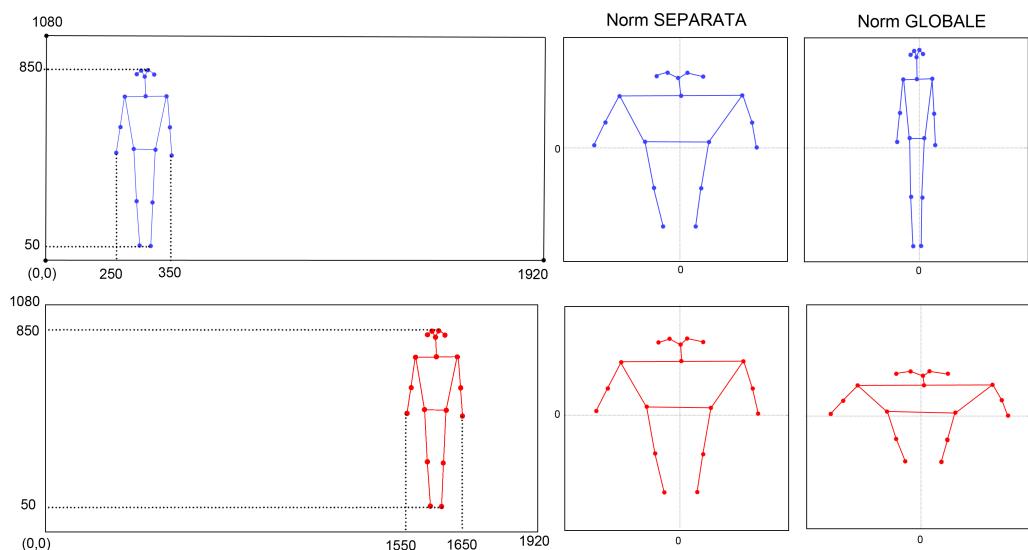


Figura 35: Differenza fra normalizzazione *globale* e *separata*. La posa blu e la posa rossa sono identiche, con l'unica differenza di essere posizionate rispettivamente nella metà sinistra e destra del frame. In questo caso stiamo trattando punti nel piano e solo normalizzandoli con la tecnica *separata* si ottengono due pose identiche.

Per questo motivo occorre fare una distinzione fra due tipi di normalizzazione dei dati:

- **Normalizzazione globale** - Questa normalizzazione tratta tutti i valori in input globalmente e nel caso di un insieme di vettori, tratta i componenti di quest'ultimi come fossero tutti appartenenti ad un unico grande vettore.

- **Normalizzazione separata** - Questa tecnica, se applicata ad un dataset di valori semplici, funziona esattamente come una normalizzazione globale, se invece viene applicata ad un dataset di vettori allora ne normalizza i valori dimensione per dimensione, trattando cioè quest'ultime in maniera indipendente l'una dall'altra. Questa tecnica è concettualmente più corretta nel caso in cui si analizzi un dataset di punti nel piano, dove vogliamo separare la normalizzazione dei valori dell'asse x da quella dell'asse y (figura 35).

Specifichiamo comunque che in entrambi i casi la normalizzazione avviene video per video, di modo da renderli totalmente indipendenti l'uno dall'altro.

#### 4.6 STRUTTURA DELLA RETE

Per quanto riguarda la rete neurale adottata per la classificazione delle azioni è stato scelto di usare le LSTM. Queste reti, nel mondo del machine learning, hanno già dimostrato la loro grande capacità espressiva quando si tratta di analizzare sequenze temporali in input.

La struttura della rete è stata fatta variare a seconda della fase degli esperimenti svolti. Nella prima fase è stato scelto di testare ogni tecnica utilizzando una rete neurale più piccola e semplice. Successivamente, solo per le tecniche più promettenti, è stato deciso di testare anche strutture di rete più complesse.

Vediamo più nel dettaglio ogni componente scelto per la rete.

- **Livelli** - In prima battuta è stato scelto di utilizzare una rete neurale composta da un solo livello LSTM ed un livello completamente connesso per l'inferenza finale. Per le tecniche più promettenti è stato incrementato progressivamente il numero dei soli livelli LSTM fino al raggiungimento ottimale dei risultati.
- **Hidden units** - Per quanto riguarda il numero di unità nascoste di ogni livello LSTM della rete, è stata utilizzata la seguente regola generale

$$N_h = \frac{N_{samples}}{\alpha \cdot (N_{in} + N_{out})}$$

dove  $N_{samples}$  è il numero di esempi del *training set*,  $N_{in}$  e  $N_{out}$  sono il numero di unità nascoste in entrata e in uscita rispettivamente e  $\alpha$  è un moltiplicatore arbitrario, generalmente fra 2 e 10, che indica

quanto generale vogliamo sia il nostro modello o, in altre parole, quanto vogliamo evitare l'*overfitting*. In questo lavoro di tesi è stato scelto  $\alpha \approx 5$  e di conseguenza ogni livello LSTM è stato strutturato con 64 hidden units.

- **Loss function** - Visto che siamo in un contesto di classificazione multi-classe, dove però ogni azione appartiene ad una e una sola categoria, la *loss function* scelta è la “*categorical crossentropy*” ovvero l'unione fra un'attivazione *soft-max* e una *cross-entropy loss*,

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \quad \text{CCE} = -\sum_i^C t_i \cdot \log(f(s)_i)$$

dove C è il numero totale di classi,  $s_i$  è lo score uscente dalla rete neurale per la classe i e  $t_i$  è il valore target per la classe i.

Considerando che il vettore target  $t$  è un vettore one-hot, composto cioè da tutti zero eccetto un solo  $t_p = 1$ , allora solo la classe corretta  $C_p$  definirà il valore della loss, che sarà quindi

$$\text{CCE} = -\log \left( \frac{e^{s_p}}{\sum_j^C e^{s_j}} \right)$$

dove  $s_p$  è il valore che la rete neurale restituisce per la classe p.

- **Batch size** - La batch size, vista la dimensione del dataset ed il numero di categorie, è stata scelta pari a  $10 \times N$ , dove N è il numero di categorie del dataset, di modo da consentire ad ogni iterazione una sufficiente variabilità del batch ed una corretta generalizzazione nell'aggiornamento dei pesi. Per l'allenamento della rete sul dataset completo la batch size è stata quindi fissata a 600.
- **Ottimizzatore** - Per la fase iniziale degli esperimenti, ogni tecnica è stata testata utilizzando *RMSProp* [30] come ottimizzatore, con un valore  $\rho = 0.9$  ed un learning rate iniziale di 0.01 decrementato gradualmente fino a 0 dopo 100 epoche, di modo da testare velocemente ognuna di esse. Per le tecniche più promettenti è stato rimosso il vincolo delle 100 epoche ed utilizzato un learning rate costante di 0,001 interrompendo l'addestramento secondo il criterio scelto (spiegato nella prossima sezione).
- **Dropout e Dropout ricorrente** - Per una maggiore semplicità di strutturazione degli esperimenti, il *dropout* ed il *dropout ricorrente*

sono sempre stati impostati entrambi allo stesso valore. Inizialmente, durante la fase di esplorazione delle tecniche, è stato assegnato a entrambi un valore pari a 0, successivamente, per le tecniche più promettenti, sono stati fatti variare a 0.5, 0.4, 0.3, 0.2, 0.15, 0.1 e 0.05.

- **Regolarizzatore e Regolarizzatore ricorrente** - Come per il dropout, anche il *regolarizzatore* ed il *regolarizzatore ricorrente* sono sempre stati settati entrambi allo stesso valore, inizialmente entrambi a 0 e successivamente, per le tecniche più promettenti, sono stati fatti variare a  $10^{-4}$ ,  $10^{-5}$  e  $10^{-6}$ .

#### 4.7 CRITERIO DI ARRESTO DELL'ADDESTRAMENTO

Sapere quando interrompere l'addestramento della rete per testarla sull'insieme di *test* non è un compito facile.

Le scuole di pensiero più diffuse sono sostanzialmente due e hanno come idea quella di arrestare l'addestramento:

1. al valor minimo della **loss di validazione**
2. al valore massimo dell'**accuratezza di validazione**

In questo lavoro di tesi sono state testate, per ogni tecnica, entrambe le ipotesi di modo da capire quale delle due è la più adatta al nostro caso di studio. Per fare ciò il dataset di allenamento è stato suddiviso in due parti:

- l'80% come effettivo dataset di allenamento
- il 20% come dataset di validazione.

# 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 considerando che un’azione dipende fortemente dall’età, il sesso, la cultura e le condizioni fisiche di chi la svolge, 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 azioni, sarà facile distinguerle e classificarle fra loro, magari identificando un semplice pattern di movimento o addirittura la struttura di un oggetto coinvolto. 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, rendendo la classificazione estremamente più difficile.

Il terzo punto da considerare per 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 perché sono state utilizzate più telecamere contemporaneamente.

Come ultima cosa, probabilmente la più importante, l’insufficientza 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.



Figura 36: 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.

NTU RGB+D si compone 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) ed 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 vari.

### 5.1 NTU RGB+D IN DETTAGLIO

Ma vediamo adesso più nel dettaglio quali sono le informazioni a disposizione in questo dataset.

**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.

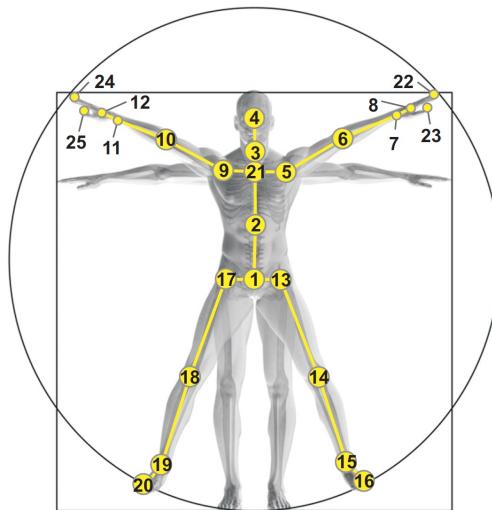


Figura 37: La configurazione dei 25 giunti utilizzati in NTU RGB+D. Le etichette utilizzate per i giunti sono: 1 - base della colonna vettoriale, 2 - punto medio della colonna vertebrale, 3 - collo, 4 - testa, 5 - spalla sinistra, 6 - gomito sinistro, 7 - polso sinistro, 8 - mano sinistra, 9 - spalla 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  $512 \times 424$  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 saltato il corrispondente pixel nel video RGB e la relativa mappa di profondità. I 25 giunti corporei utilizzati sono raffigurati in figura 37.

I video RGB sono normali video registrati con una risoluzione  $1920 \times 1080$ .

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

**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 36 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 ripresa da 3 telecamere di modo da generare 3 prospettive diverse della stessa azione contemporaneamente. Le 3 telecamere sono state posizionate tutte alla stessa altezza e sempre con la stessa angolazione di ripresa, ovvero  $-45^\circ$ ,  $0^\circ$  e  $45^\circ$ .

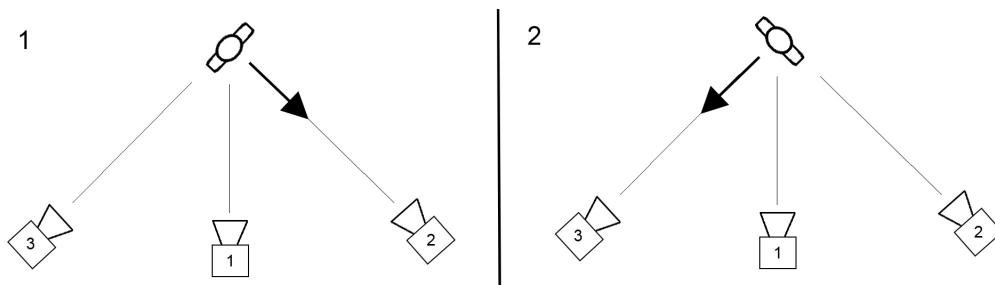


Figura 38: Disposizione delle telecamere per ogni setup. La telecamera 1 ha ripreso ogni azione da una posizione centrale mentre le telecamere 2 e 3 hanno ripreso le azioni con un'angolazione di  $45^\circ$ . Ogni attore ha ripetuto l'azione 2 volte: una rivolto verso la telecamera di destra e una verso quella di sinistra.

Ad ogni attore è stato inoltre richiesto di ripetere l'azione due volte: una volta verso la telecamera 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  $45^\circ$  da sinistra e quella ad una angolazione di  $45^\circ$  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  $0^\circ$ , la telecamera 2 da  $45^\circ$  e la 3 da

$-45^\circ$ . Uno schema della disposizione delle telecamere e della procedura di ripresa è raffigurato in figura 38

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. In tabella 4 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 4: 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  $45^\circ$  e  $-45^\circ$ . Dividendo il dataset con questo criterio si ottiene un insieme di training composto da 37920 video e quello di test da 18960.

# 6

---

## ESPERIMENTI E RISULTATI

---

In questo capitolo verranno mostrati non solo i risultati ottenuti per i vari esperimenti effettuati ma anche il procedimento logico adottato al fine di raggiungere la miglior classificazione possibile.

La legenda di ogni tabella in questo capito sarà la seguente

- **rimoz 0** - rimozione degli zeri
- **VIDEO/FRAME** - tecnica del centro del video/frame
- **VIDEO/FRAME PERS** - tecnica del centro del video/frame per ogni posa
- **3-5-17 BAR** - tecnica dei 3-5-17 baricentri
- **3-5-17 BAR ASS** - tecnica dei 3-5-17 baricentri in valore assoluto
- **DIST REL** - tecnica delle distanze relative
- **DIST CUM** - tecnica delle distanze cumulate
- **NEXT** - tecnica del next frame con S=1
- **NEXT 3-5-15** - tecnica del next frame con S=3-5-15
- **norm** - normalizzazione globale
- **normXY** - normalizzazione separata
- **MIN LOSS** - i risultati sono stati ottenuti interropendo l'allenamento secondo il criterio del *minor valore della loss di validazione*, se non specificato allora viene sottinteso del *maggior valore dell'accuratezza di validazione*.
- **semplice** - non è stata utilizzata alcuna tecnica aggiuntiva dopo l'estrazione della posa.

### 6.1 UNA PORZIONE RAPPRESENTATIVA DEL DATASET

Come prima cosa, considerata la dimensione del dataset (56880 video), è stato scelto un sottoinsieme di azioni di modo da poter lavorare inizialmente su una porzione più piccola, sufficientemente rappresentativa e di difficile categorizzazione, che permettesse una più rapida esplorazione delle tecniche.

A questo scopo sono state scelte 8 azioni con le seguenti caratteristiche:

- 2 azioni di coppia simili fra loro
- 3 coppie di azioni individuali simili fra loro

Le azioni scelte, fra quelle proposte da NTU-RGB+D, sono state:

1. dare uno pacca sulla schiena a qualcuno (azione di coppia)
2. toccare la tasca di qualcuno (azione di coppia)
3. leggere
4. giocare col telefonino/tablet
5. applaudire
6. strofinarsi le mani
7. scrivere
8. usare una tastiera

Questa porzione di dataset così creata è sicuramente di difficile categorizzazione, dato che molte delle azioni scelte si differenziano solo per piccoli movimenti delle dita non distinguibili col solo della posa generata da PoseNet e Detectron2.

### 6.2 FASE 1: TEST DI TUTTE LE TECNICHE

Per questa prima fase è stato scelto di testare ogni tecnica del capitolo 4 allenando una semplice rete neurale composta da un livello LSTM ed un livello completamente connesso per l'inferenza finale, ignorando per il momento dropout e regolarizzazione.

I risultati ottenuti sono illustrati in tabella 39.

TECNICA	1 Livello LSTM					
	DETECTRON			POSENET		
	Media	CV	CS	Media	CV	CS
rimoz 0 + VIDEO + norm	63,53	66,14	60,92	53,88	59,22	48,55
rimoz 0 + FRAME + norm	63,29	65,98	60,60	57,96	59,89	56,02
rimoz 0 + FRAME PERS + norm	62,48	64,04	60,92	58,11	58,78	57,43
rimoz 0 + FRAME + norm (MIN LOSS)	62,17	65,23	59,10	57,18	59,02	55,34
rimoz 0 + VIDEO PERS + norm	61,00	62,22	59,78	59,63	62,74	56,52
rimoz 0 + FRAME PERS + norm (MIN LOSS)	60,92	63,05	58,79	55,63	56,69	54,57
rimoz 0 + DIST REL + norm	60,22	64,56	55,89	51,28	48,89	53,67
rimoz 0 + 3BAR + norm	60,17	66,81	53,53	57,39	53,40	61,37
rimoz 0 + 3BAR ASS + norm (MIN LOSS)	59,60	63,85	55,34	62,03	62,46	61,59
rimoz 0 + 3BAR ASS + norm	59,47	64,64	54,30	63,04	61,67	64,40
rimoz 0 + VIDEO + norm (MIN LOSS)	59,31	61,27	57,34	52,53	59,22	45,83
rimoz 0 + 5BAR + norm	59,06	56,29	61,82	60,59	62,34	58,83
rimoz 0 + DIST CUM + norm	58,64	61,75	55,53	32,93	12,50	53,35
rimoz 0 + DIST REL + norm (MIN LOSS)	58,46	61,71	55,21	51,61	49,64	53,58
rimoz 0 + VIDEO PERS + norm (MIN LOSS)	58,12	59,14	57,11	57,87	60,52	55,21
rimoz 0 + 3BAR + norm (MIN LOSS)	57,60	64,44	50,77	55,76	49,96	61,55
rimoz 0 + 17BAR ASS + norm	57,03	62,30	51,77	54,71	58,19	51,22
rimoz 0 + 5BAR + norm (MIN LOSS)	55,41	53,13	57,70	54,83	55,58	54,08
normXY	55,04	60,13	49,96	57,15	55,10	59,19
normXY (MIN LOSS)	54,69	59,65	49,73	52,89	54,55	51,22
norm	53,96	55,66	52,26	53,88	55,46	52,31
norm (MIN LOSS)	53,76	55,62	51,90	53,46	55,10	51,81
rimoz 0 + norm (MIN LOSS)	53,56	53,44	53,67	52,64	53,24	52,04
rimoz 0 + NEXT 3 + norm	52,65	57,12	48,19	14,99	12,54	17,44
rimoz 0 + DIST CUM + norm (MIN LOSS)	52,41	59,26	45,56	30,16	12,54	47,78
rimoz 0 + norm	52,29	54,98	49,59	51,98	53,24	50,73
rimoz 0 + 5BAR ASS + norm	51,17	47,23	55,12	53,42	53,80	53,03
rimoz 0 + 17BAR ASS + norm (MIN LOSS)	49,93	55,34	44,52	50,52	56,21	44,84
rimoz 0 + 5BAR ASS + norm (MIN LOSS)	48,80	46,56	51,04	47,78	45,69	49,86
rimoz 0 + NEXT 3 + norm (MIN LOSS)	47,16	46,72	47,60	12,57	12,54	12,59
Semplice	39,49	43,47	35,51	42,09	43,51	40,67
Semplice (MIN LOSS)	39,38	43,16	35,60	42,21	42,88	41,53
rimoz 0 + 17BAR + norm	34,96	12,54	57,38	33,18	12,50	53,85
rimoz 0 + 17BAR + norm (MIN LOSS)	33,99	12,54	55,44	32,86	12,54	53,17
rimoz 0 + NEXT 7 + norm	12,54	12,58	12,50	29,50	13,85	45,15
rimoz 0 + NEXT 15 + norm	12,52	12,54	12,50	21,40	30,30	12,50
rimoz 0 + NEXT 15 + norm (MIN LOSS)	12,52	12,54	12,50	21,48	30,46	12,50
rimoz 0 + NEXT 7 + norm (MIN LOSS)	12,52	12,54	12,50	25,40	13,85	36,96
rimoz 0 + NEXT + norm (MIN LOSS)	12,52	12,54	12,50	12,52	12,50	12,55
rimoz 0 + NEXT + norm	12,47	12,54	12,41	12,54	12,54	12,55

Figura 39: Risultati ottenuti per ogni tecnica allenando una rete con un solo livello LSTM. CV= Cross View; CS= Cross Subject.

Da questa prima classifica dei risultati, apparte avere un'idea su quali sono le tecniche più efficienti, si evince anche qualche altro aspetto importante:

- Categorizzare le azioni utilizzando semplicemente i valori della stima della posa ottenuti da Detectron-2 e Posenet è decisamente non sufficiente, ottenendo un'accuratezza massima del 42,21% (in media)
- Quanto trattiamo punti sul piano la *normalizzazione separata* ha risultati nettamente migliori rispetto a quella *globale* ( $\approx +2\%$  in media).
- In quasi tutti i casi è preferibile interrompere l'allenamento della rete al *massimo valore di accuratezza di validazione* invece che al *minor valore della loss di validazione*.

### 6.3 FASE 2: NUMERO DI LIVELLI

A questo punto è interessante capire se aumentare il numero la profondità della rete porta o meno dei benefici.

Sono state quindi usate le migliori tecniche della fase precedente per allenare 3 diverse reti neurali, così strutturate:

- **1 livello** LSTM da 64 hidden units ed un livello completamente connesso ( $\approx 35k$  parametri)
- **2 livelli** LSTM da 64 hidden units ciascuno ed un livello completamente connesso ( $\approx 68k$  parametri)
- **3 livelli** LSTM da 64 hidden units ciascuno ed un livello completamente connesso ( $\approx 101k$  parametri).

Come si può evincere dai risultati in tabella 40, incrementare la profondità della rete a due livelli spesso aiuta il processo di classificazione delle azioni, mentre aggiungerne un terzo sembrerebbe peggiorare i risultati, probabilmente a causa di un maggiore overfitting sui dati di allenamento.

TECNICA	1 Livello LSTM						2 Livelli LSTM						3 Livelli LSTM					
	DETECTRON			POSENET			DETECTRON			POSENET			DETECTRON			POSENET		
	Media	CV	CS	Media	CV	CS	Media	CV	CS	Media	CV	CS	Media	CV	CS	Media	CV	CS
rimoz 0 + VIDEO + norm	<b>63,53</b>	66,14	60,92	53,88	59,22	48,55	60,43	61,71	59,15	<b>57,00</b>	57,83	56,16	<b>59,24</b>	62,82	55,66	<b>56,86</b>	60,09	53,62
rimoz 0 + FRAME + norm	63,29	65,98	60,60	<b>57,96</b>	59,89	56,02	<b>63,43</b>	64,72	62,14	56,91	58,03	55,80	33,02	12,50	53,53	12,50	12,50	12,50
rimoz 0 + FRAME PERS + norm	<b>62,48</b>	64,04	60,92	<b>58,11</b>	58,78	57,43	57,52	59,65	55,39	57,88	59,65	56,11	36,16	59,81	12,50	55,30	56,80	53,80
rimoz 0 + FRAME + norm (MIN LOSS)	<b>62,17</b>	65,23	59,10	<b>57,18</b>	59,02	55,34	61,28	62,10	60,46	55,28	57,20	53,35	32,77	12,50	53,03	12,50	12,50	12,50
rimoz 0 + VIDEO PERS + norm	61,00	62,22	59,78	<b>59,63</b>	62,74	56,52	60,83	64,91	56,75	58,52	58,35	58,70	<b>61,14</b>	61,91	60,37	32,81	12,50	53,13
rimoz 0 + FRAME PERS + norm (MIN LOSS)	<b>60,92</b>	63,05	58,79	55,63	56,69	54,57	56,71	59,61	53,80	<b>56,76</b>	56,05	57,47	33,90	55,30	12,50	54,77	56,69	52,85
rimoz 0 + DIST REL + norm	<b>60,22</b>	64,56	55,89	51,28	48,89	53,67	59,88	59,97	59,78	<b>55,96</b>	57,16	54,76	59,00	62,66	55,34	<b>58,64</b>	62,66	54,62
rimoz 0 + 3BAR + norm	60,17	66,81	53,53	57,39	53,40	61,37	<b>62,63</b>	61,71	63,54	<b>60,35</b>	62,74	57,97	59,44	61,27	57,61	<b>58,64</b>	58,03	59,24
rimoz 0 + 3BAR ASS + norm (MIN LOSS)	59,60	63,85	55,34	<b>62,03</b>	62,46	61,59	59,11	62,07	56,16	61,70	61,99	61,41	<b>61,63</b>	60,72	62,55	59,18	60,84	57,52
rimoz 0 + 3BAR ASS + norm	59,47	64,64	54,30	<b>63,04</b>	61,67	64,40	61,86	59,42	64,31	60,89	61,59	60,19	<b>62,22</b>	61,35	63,09	61,22	60,52	61,91
rimoz 0 + VIDEO + norm (MIN LOSS)	59,31	61,27	57,34	52,53	59,22	45,83	<b>59,49</b>	60,64	58,33	<b>55,54</b>	55,38	55,71	59,32	60,52	58,11	55,04	57,99	52,08
rimoz 0 + 5BAR + norm	59,06	56,29	61,82	<b>60,59</b>	62,34	58,83	<b>63,92</b>	68,87	58,97	57,86	56,21	59,51	34,85	12,58	57,11	33,13	53,76	12,50
rimoz 0 + DIST CUM + norm	<b>58,64</b>	61,75	55,53	32,93	12,50	53,35	56,70	58,82	54,57	<b>56,33</b>	57,67	54,98	36,00	59,49	12,50	54,45	54,91	53,99
rimoz 0 + DIST REL + norm (MIN LOSS)	58,46	61,71	55,21	51,61	49,64	53,58	<b>59,84</b>	59,53	60,15	56,25	57,16	55,34	57,34	59,06	55,62	<b>57,60</b>	60,80	54,39
rimoz 0 + VIDEO PERS + norm (MIN LOSS)	58,12	59,14	57,11	57,87	60,52	55,21	60,83	64,91	56,75	<b>58,52</b>	58,35	58,70	<b>61,14</b>	61,91	60,37	32,81	12,50	53,13
rimoz 0 + 3BAR + norm (MIN LOSS)	57,60	64,44	50,77	55,76	49,96	61,55	<b>60,10</b>	60,32	59,87	<b>58,37</b>	62,03	54,71	56,30	56,88	55,71	58,06	58,47	57,65

Figura 40: Risultati ottenuti facendo variare la profondità della rete. I valori in grassetto sono i migliori per la tecnica corrispondente.

#### 6.4 FASE 3: DROPOUT E REGOLARIZZATORI

Adesso che abbiamo un'idea di quali sono le tecniche più efficienti possiamo ridurre il raggio d'azione e testare solo quest'ultime sul tutto il dataset facendo variare per ognuna di esse il valore del regolarizzatore e del dropout (ricordiamo che per semplicità di strutturazione degli esperimenti il *regolarizzatore ricorrente* ed il *dropout ricorrente* sono stati sempre impostati allo stesso valore del *regolarizzatore* e del *dropout del kernel* rispettivamente). I risultati ottenuti in questa fase sono elencati in tabella ??

#### 6.5 FASE 4: ALLENAMENTO COMPLETO TECNICA MIGLIORE

Quest'ultima fase è dedicata a finalizzare la miglior tecnica trovata, rimuovendo il vincolo delle 100 epoche, fissando il learning rate a 0,001 e interrompendo l'allenamento al raggiungimento del miglior valore di accuratezza sul dataset. Ci sono volute circa [...] epoche per raggiungere il miglior valore e i risultati ottenuti al termine di questo lungo procedimento sono i seguenti:

# 7

---

## CONCLUSIONI E SVILUPPI FUTURI

---

In questo lavoro di tesi ci siamo dedicati al riconoscimento e alla classificazione di azioni umane tramite tecniche di apprendimento profondo per la stima della posa.

A tale scopo abbiamo deciso di ideare un algoritmo che non avesse bisogno di informazioni iniziali complesse, come ad esempio la posizione dei giunti dei soggetti inquadrati, ma che attraverso l'uso dei soli video RGB fosse in grado di estrapolare tutte le informazioni necessarie.

Al fine di ottenere il miglior algoritmo abbiamo eseguito un serie di esperimenti strutturati secondo un procedimento ben preciso, che permettesse l'esplorazione rapida di ogni tecnica ideata affinando progressivamente i risultati per quelle più promettenti.

Quello che abbiamo ottenuto alla fine del processo sono due algoritmi con una discreta capacità di classificazione delle azioni umane e per la loro semplicità anche un'elevata portabilità.

Nello specifico, gli algoritmi che proponiamo sono:

- il classificatore "*3BAR-Detectron*" con un accuratezza CS: 90% e CV 80%
- il classificatore "*3BAR-Posenet*" con un accuratezza CS: -% e CV -%

Questi due algoritmi, pur facendo uso dei soli video RGB, superano le performance di diversi lavori nel mondo scientifico dedicati al medesimo scopo, inoltre essendo non complessi sono facilmente migliorabili.

Una visione globale dei risultati ottenuti con i due algoritmi, comparati a quelli dello stato dell'arte, è sintetizzata in tabella 5.

### 7.1 SVILUPPI FUTURI

Gli sviluppi futuri per gli algoritmi proposti sono molteplici. In questo lavoro di tesi ci siamo concentrati principalmente sulla trasformazione

Articolo	Posa	RGB	IR	Accuratezza cross-view (%)	Accuratezza cross-subject (%)
Lie Group [31]	✓			52.8	50.1
H-RNN [32]	✓			64.0	59.1
Deep LSTM [33]	✓			67.3	60.7
PA-LSTM [33]	✓			70.3	62.9
ST-LSTM+TS [34]	✓			77.7	69.2
Temporal Conv [35]	✓			83.1	74.3
DSSCA-SSLM [41]		✓		-	74.9
Skelemotion [45]	✓			84.7	76.5
C-CNN+MTLN [36]	✓			84.8	79.6
VA-LSTM [37]	✓			87.6	79.4
Chained [40]		✓		-	80.8
ST-GCN [38]	✓			88.3	81.5
<b>3-BAR-PoseNet</b>		✓		-	-
<b>3-BAR-Detectron</b>		✓		<b>89.39</b>	<b>80.88</b>
SR-TSL [39]	✓			92.4	84.8
2D-3D-Softargmax [42]		✓		-	85.5
Glimpse Clouds [43]		✓		93.2	86.6
PB-GCN [46]	✓			93.2	87.5
MFAS [47]	✓	✓		-	90.04
FUSION [28]	✓		✓	94.5	91.6
PoseMap [24]	✓	✓		95.2	91.7
MMTM [44]	✓	✓		-	91.99
Action Machine [25]		✓		97.2	94.3

Tabella 5: Risultati di *3-BAR-Detectron* e *3-BAR-Posenet* comparati allo stato dell'arte.

dei video RGB in una sequenza di pose successivamente rielaborate ed abbiamo ignorato completamente l'informazione utilizzabile dalla classificazione delle azioni utilizzando direttamente i valori RGB dei video. Potrebbe essere interessante combinare l'informazione proveniente dalle due tipologie di classificazione.

Un altro possibile sviluppo di questo lavoro è testare gli algoritmi proposti anche sui dataset *NTU-RGB+D120* [20], *MSR Daily Activity3D* [48] e *Northwestern-UCLA Multiview Action 3D (NUCLA)* [49] per validare le prestazioni ottenute su NTU-RGB+D oppure analizzare le eventuali differenze.



---

## 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-7ddobc881cd5>  
(Cited on pages 34, 37, and 45.)
- [4] Detectron2 - *Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, Ross Girshick* - <https://github.com/facebookresearch/detectron2> - 2019  
(Cited on page 35.)
- [5] Detectron - *Ross Girshick, Ilya Radosavovic, Georgia Gkioxari, Piotr Dollár, Kaiming He* - <https://github.com/facebookresearch/detectron> - 2018
- [6] Caffe2 - <https://caffe2.ai/docs> (Cited on pages 34 and 45.)
- [7] Mask R-CNN - *Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick* - 2017 (Cited on page 39.)
- [8] Focal Loss for Dense Object Detection - *Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár* - 2017 (Cited on page 39.)
- [9] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks - *Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun* - 2015 (Cited on pages 29, 39, and 40.)
- [10] Fast R-CNN - *Ross Girshick* - 2015 (Cited on pages 39 and 40.)
- [11] R-FCN: Object Detection via Region-based Fully Convolutional Networks - *Jifeng Dai, Yi Li, Kaiming He, Jian Sun* - 2016 (Cited on pages 40 and 41.)

- [12] Aggregated Residual Transformations for Deep Neural Networks - *Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He* - 2016  
(Cited on pages 40 and 42.)
- [13] Deep Residual Learning for Image Recognition - *Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun* - 2015 (Cited on page 40.)
- [14] Feature Pyramid Networks for Object Detection - *Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie* - 2016 (Cited on page 40.)
- [15] Very Deep Convolutional Networks for Large-Scale Image Recognition - *Karen Simonyan, Andrew Zisserman* - 2014 (Cited on pages 29, 31, and 40.)
- [16] Cascade R-CNN: High Quality Object Detection and Instance Segmentation - *Zhaowei Cai, Nuno Vasconcelos* - 2019 (Cited on page 40.)
- [17] Panoptic Feature Pyramid Networks - *Alexander Kirillov, Ross Girshick, Kaiming He, Piotr Dollár* - 2019 (Cited on page 40.)
- [18] TensorMask: A Foundation for Dense Object Segmentation - *Xinlei Chen, Ross Girshick, Kaiming He, Piotr Dollár* - 2019 (Cited on page 39.)
- [19] NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis - *Amir Shahroudy, Jun Liu, Tian-Tsong Ng, Gang Wang* - 2016 (Cited on page 39.)
- [20] NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding - *Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan and Alex C. Kot* - 2019 (Cited on page 40.)
- [21] Long short-term memory - *Sepp Hochreiter; Jürgen Schmidhuber* - 1997  
(Cited on pages 27, 45, 53, 62, and 65.)
- [22] Recurrent Nets that Time and Count - *Felix A. Gers, Jürgen Schmidhuber* - 2000 (Cited on page 75.)  
(Cited on page 18.)
- [23] Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation - *Kyunghyun Cho; Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio* - 2014 (Cited on page 22.)

- [24] Recognizing Human Actions as the Evolution of Pose Estimation Maps - *Mengyuan Liu, Junsong Yuan* - 2018 (Cited on page 22.)
- [25] Action Machine: Rethinking Action Recognition in Trimmed Videos - *Jiagang Zhu, Wei Zou, Liang Xu, Yiming Hu, Zheng Zhu, Manyu Chang, Junjie Huang, Guan Huang, Dalong Du* - 2018 (Cited on pages 24, 32, and 74.)
- [26] Quo vadis, action recognition? A new model and the kinetics dataset. *J. Carreira, A. Zisserman* - 2017 (Cited on pages 24, 32, and 74.)
- [27] Deformable convolutional networks. *J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei.* - 2017 (Cited on page 29.)
- [28] Infrared and 3D skeleton feature fusion for RGB-D action recognition - *Alban Main de Boissiere e Rita Noumeir* - 2020 (Cited on page 28.)
- [29] Interpretable 3D human action analysis with temporal convolutional networks - *T. S. Kim, A. Reiter* - 2017 (Cited on pages 32 and 74.)
- [30] RMSProp and equilibrated adaptive learning rates for non-convex optimization - *Yann N. Dauphin, Harm de Vries, Junyoung Chung, Yoshua Bengio* - 2015 (Cited on page 23.)
- [31] Human action recognition by representing 3d skeletons as points in a lie group - *R. Vemulapalli, F. Arrate e R. Chellappa* - 2014 (Cited on page 59.)
- [32] Hierarchical recurrent neural network for skeleton based action recognition - *Y. Du, W. Wang e L. Wang.* - 2015 (Cited on pages 32 and 74.)
- [33] Ntu rgb+d: A large scale dataset for 3d human activity analysis- *A. Shahroudy, J. Liu, T.-T. Ng e G. Wang* - 2016 (Cited on pages 32 and 74.)
- [34] Spatio-temporal lstm with trust gates for 3d human action recognition - *J. Liu, A. Shahroudy, D. Xu e G. Wang* - 2016 (Cited on pages 32 and 74.)
- [35] Interpretable 3d human action analysis with temporal convolutional networks - *T. S. Kim e A. Reiter* - 2017 (Cited on pages 32 and 74.)

- [36] A new representation of skeleton sequences for 3d action recognition - *Q. Ke, M. Bennamoun, S. An, F. Sohel e F. Boussaid* - 2017 (Cited on pages 32 and 74.)
- [37] View adaptive recurrent neural networks for high performance human action recognition from skeleton data - *P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue e N. Zheng* - 2017 (Cited on pages 32 and 74.)
- [38] Spatial temporal graph convolutional networks for skeleton-based action recognition - *S. Yan, Y. Xiong e D. Lin* - 2018 (Cited on pages 32 and 74.)
- [39] Skeletonbased action recognition with spatial reasoning and temporal stack learning - *C. Si, Y. Jing, W. Wang, L. Wang e T. Tan* - 2018 (Cited on pages 32 and 74.)
- [40] Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection - *M. Zolfaghari, G. L. Oliveira, N. Sedaghat e T. Brox* - 2017 (Cited on pages 32 and 74.)
- [41] Deep multimodal feature analysis for action recognition in rgb+d videos - *A. Shahroury, T. Ng, Y. Gong e G. Wang* - 2018 (Cited on pages 32 and 74.)
- [42] 2d/3d pose estimation and action recognition using multitask deep learning - *D. C. Luvizon, D. Picard e H. Tabia.* - 2018 (Cited on pages 32 and 74.)
- [43] Glimpse clouds: Human activity recognition from unstructured feature points - *F. Baradel, C. Wolf, J. Mille e G. W. Taylor* - 2018 (Cited on pages 32 and 74.)
- [44] MMTM: Multimodal Transfer Module for CNN Fusion - *H. R. V. Joze, A. Shaban, M. L. Iuzzolino, K. Koishida* - 2019 (Cited on pages 32 and 74.)  
(Cited on pages 32 and 74.)
- [45] SkeleMotion: A New Representation of Skeleton Joint Sequences Based on Motion Information for 3D Action Recognition - *C Caetano, J Sena, F Brémond, J. A. dos Santos, W. R. Schwartz* - 2019 (Cited on pages 32 and 74.)
- [46] Part-based Graph Convolutional Network for Action Recognition - *K Thakkar, P. J. Narayanan* - 2018 (Cited on pages 32 and 74.)

- [47] MFAS: Multimodal Fusion Architecture Search - *J. Perez-Rua, V. Vielzeuf, S. Pateux, M. Baccouche, F. Jurie* - 2019 (Cited on pages 32 and 74.)
- [48] Mining actionlet ensemble for action recognition with depth cameras - *J. Wang, Z. Liu, Y. Wu, and J. Yuan.* - 2012 (Cited on page 75.)
- [49] Cross-view action modeling, learning and recognition - *J. Wang, X. Nie, Y. Xia, Y. Wu, and S. Zhu.* - 2014 (Cited on page 75.)