POLITECNICO DI TORINO

Corso di Laurea Magistrale in
Ingegneria Informatica

Tesi di Laurea Magistrale

Action recognition per il tennis basato su scheletri 2D

Analisi delle prestazioni al variare dei punti dello scheletro presi in considerazione

Relatore
Prof. Paolo Garza

Candidato
Gabriele Cedrino

Correlatore esterno
Dott. Mirko Zaffaroni

Aprile 2021

Sommario

Negli ultimi anni sono stati fatti grandi passi in avanti nell'utilizzo delle reti convoluzionali per l'*action recognition*, e i campi di applicazione si sono moltiplicati andando ad interessare tra gli altri anche videosorveglianza, sorveglianza sanitaria e analisi sportiva. Questa tesi si va a inserire proprio all'interno di quest'ultima categoria, in particolare mette le basi per la realizzazione di un'applicazione per il riconoscimento delle azioni nel gioco del tennis, che possa essere utilizzata in tempo reale durante gli allenamenti. La rete convoluzionale presa in considerazione basa le sue predizioni sull'analisi dei movimenti dello scheletro umano e in questo studio si vuole analizzare come varino le prestazioni in termini di correttezza delle predizioni ed efficienza temporale, al diminuire del numero di giunzioni dello scheletro utilizzate. Inoltre, si proverà a sfruttare una rappresentazione della racchetta per effettuare la classificazione delle azioni, sempre con l'obbiettivo di identificare una combinazione di punti che permetta al sistema di essere veloce e sufficientemente accurato. Gli esperimenti di questo elaborato affronteranno prima lo studio dell'accuratezza e di altre metriche per stabilire la bontà del modello utilizzato e successivamente quello sui tempi di esecuzione per poi concludere con un'analisi complessiva che andrà ad individuare la miglior combinazione di punti rispetto a questi parametri. I risultati ottenuti indicano che sfruttare solamente lo scheletro delle braccia o di parte di esse, come polsi e gomiti, permette di mantenere livelli di accuratezza soddisfacenti, in alcuni casi anche superiori all'utilizzo dell'intero scheletro, con tempi di esecuzione minori. In particolare, l'utilizzo dei soli punti di polsi e gomiti garantisce valori di accuratezza leggermente superiori rispetto all'utilizzo dei *keypoint* dell'intero corpo, ma con un considerevole guadagno temporale che si attesta sul 17%. Per quanto riguarda l'utilizzo della racchetta, i risultati più significativi sono stati raggiunti con la combinazione di un punto che identifica il centro di essa, insieme a polsi e gomiti., con la quale si sono ottenuti livelli di accuratezza molto simili all'utilizzo dello scheletro intero ma con tempi di esecuzione inferiori.

Ringraziamenti

Vorrei dedicare questo spazio a tutte le persone che mi hanno supportato e sopportato durante l'intero percorso di studi che vede la sua conclusione con la realizzazione di questo lavoro di tesi.
Innanzitutto, ringrazio il mio relatore Prof. Paolo Garza per la sua disponibilità e tempestività nel rispondere a tutte le mie domande e per gli indispensabili consigli per la stesura dell'elaborato.
Un ringraziamento speciale va allo staff della Fondazione LINKS, per avermi permesso di svolgere questo approfondimento su un tema che mi appassiona e per aver dispensato utili consigli volti a migliorare la mia ricerca. In particolare, ringrazio il Dott. Mirko Zaffaroni per avermi seguito in ogni fase del mio percorso di tesi, per la sua enorme disponibilità e per tutte le conoscenze che mi ha trasmesso.
Grazie a mia madre, mio padre e mio fratello senza i quali oggi non potrei essere quello che sono e non avrei mai potuto raggiungere questo importante traguardo della mia vita.
E ultimi ma non per importanza ringrazio tutti i miei amici, il gruppo dei Festaioli, con i quali ho passato tanti momenti indimenticabili che mi hanno permesso di vivere al meglio questi lunghi anni di studio.

Indice

Lista immagini

Lista tabelle

Capitolo 1

Introduzione

Questo studio nasce dalla necessità di realizzare un'applicazione di supporto per gli allenatori di tennis, applicazione che renda possibile analizzare in tempo reale i dati relativi ai colpi effettuati dal giocatore. Per raggiungere questo obbiettivo è necessario avere un sistema che reagisca in tempi brevi ma allo stesso tempo con precisione. Proprio su questi ultimi due aspetti verrà posta l'attenzione, e per farlo sarà utilizzata una rete neurale per il riconoscimento delle azioni umane, che sfrutta una rappresentazione dello scheletro umano, cercando di individuare la miglior combinazione di elementi dello scheletro da utilizzare.

1.1 Human Action Recognition

Con human action recognition si indica, nel contesto della computer vision, la funzione di classificare le azioni umane contenute all'interno di un video, utilizzando etichette di classe predefinite. Negli ultimi dieci anni, gli studi sulla human action recognition hanno acquisito sempre maggiore importanza ed hanno raggiunto risultati promettenti in diversi campi di applicazione tra cui, videosorveglianza, human-computer interaction, sorveglianza sanitaria e analisi sportiva. Questo sviluppo è stato favorito dall'introduzione di nuove tecniche e strutture nel campo delle reti neurali ed alla crescente disponibilità di risorse computazionali. Nonostante ciò, non possiamo ancora affermare che l'attività della Human Action Recognition sia stata affrontata in modo esaustivo, come si può fare, ad esempio, per la classificazione delle immagini. Le difficoltà principali da affrontare quando si lavora con i video sono principalmente due: la potenza di calcolo necessaria per processare grandi quantità di essi e la difficoltà nell'estrarre in modo efficiente le features spazio-temporali.
Infatti, a differenza dell'analisi delle immagini, quando abbiamo a che fare con delle sequenze video, dobbiamo andare a considerare anche la dimensione tempo ed è necessario trovare un modo efficace per estrarre e combinare le features temporali. Un esempio significativo per dimostrare l'importanza di questa nuova dimensione è quello dell'analisi di un uomo che si siede su una sedia. Se considerassimo solamente una singola immagine di un uomo che si trova a metà, tra la posizione seduta e quella eretta, non saremo in grado di distinguere se si sta alzando oppure sedendo. Per farlo avremo bisogno di analizzare una sequenza di immagini successive nel tempo per capire la direzione del suo movimento.
Negli ultimi anni sono state sviluppate diverse tipologie di reti neurali in grado di tenere in considerazione anche la dimensione temporale. Tra le tecniche principali troviamo l'utilizzo di due stream di input differenti all'interno della stessa rete, uno per l'analisi delle caratteristiche spaziali e l'altro per quelle temporali [1][2], l'impiego di convoluzioni tridimensionali [3] oppure la combinazione di queste due tecniche [4]. E poi ancora le reti neurali con struttura Long Short-Term Memory (LSTM) [5] che utilizzano una struttura particolare dei singoli nodi, in grado di mantenere una memoria di un certo numero di frame del passato e combinarli con il nuovo frame da analizzare.

1.2 Obbiettivo e struttura di questa tesi

La mia tesi affronta il task della human action recognition nell'ambito del gioco del tennis, utilizzando una rete neurale, che sfrutta lo scheletro del giocatore per determinare quale tipo di giocata ha compiuto. Lo scheletro è rappresentato attraverso dei keypoint corrispondenti alle principali giunzioni e parti del corpo. L'obbiettivo è quello di effettuare un'analisi prestazionale, in termini di accuratezza e tempi di esecuzione, al variare del numero di punti chiave presi in considerazione. In questo modo si vuole capire quale sia la miglior combinazione di punti da utilizzare, per avere un sistema che sia veloce, ma allo stesso tempo sufficientemente accurato. In un secondo momento, in aggiunta allo scheletro umano, proverò ad utilizzare alcuni punti che rappresentano la racchetta per verificare se si ottiene un miglioramento nelle prestazioni.
Nei prossimi capitoli si farà una breve introduzione al machine learning per passare poi a descrivere gli aspetti principali delle reti neurali ed in particolare di quelle convoluzionali con i vari livelli che le compongono. Terminata questa parte introduttiva di teoria verranno presi in considerazione alcuni paper che rappresentano l'attuale stato dell'arte e che sono strettamente collegati con questo lavoro. Infine, nel capitolo conclusivo sarà descritto il lavoro svolto, partendo da una descrizione dei datasets e delle reti utilizzate e proseguendo con l'analisi dei risultati ottenuti.

Capitolo 2

Machine Learning

Negli ultimi anni abbiamo assistito ad una diffusione sempre maggiore dell'utilizzo delle tecniche di machine learning, favorito dalla crescente disponibilità di risorse computazionali e di dati. Le applicazioni del machine learning sono in continua evoluzione e cominciano ad affiancarci in molti aspetti della nostra vita quotidiana come sistemi a guida autonoma, filtri antispam, social media e anche in molti processi produttivi aziendali. La maggior parte delle aziende sta investendo in questo settore, che sicuramente avrà un ruolo fondamentale anche nel futuro prossimo. In questo capitolo provo a spiegare cosa è il machine learning e quali sono le conoscenze di base necessarie per poter comprendere al meglio il mio elaborato.

2.1 Introduzione al Machine Learning

Machine learning, in italiano "apprendimento automatico", è un sottoinsieme dell'intelligenza artificiale (AI) che fornisce alle macchine la capacità di apprendere e migliorarsi in modo automatico attraverso l'esperienza, senza bisogno di essere esplicitamente programmate. Con l'utilizzo del machine learning cambia radicalmente l'approccio del programmatore: se prima era necessario scrivere righe di codice molto dettagliate per indicare alla macchina come comportarsi in ogni situazione, adesso si hanno a disposizione algoritmi in grado di apprendere da soli e di conseguenza compiere determinate azioni. Per riuscire ad acquisire questa conoscenza, gli algoritmi analizzano insiemi di dati dai quali ricavano correlazioni e successivamente vere e proprie regole. Maggiori sono la quantità e la qualità dei dati e migliori saranno le previsioni fatte dall'algoritmo. Proprio per questo motivo un ruolo fondamentale per l'efficacia di un algoritmo di machine learning è ricoperto dai big data, che vanno però gestiti, preparati ed integrati in modo opportuno. I dati utilizzati nel mondo del machine learning possono essere suddivisi in due categorie: dati etichettati e dati non etichettati. Sicuramente disporre di dati etichettati è meglio e permette di raggiungere risultati migliori, ma allo stesso tempo richiede anche un enorme sforzo manuale per assegnare tutte le etichette.
A seconda del problema che si deve affrontare e della tipologia di dati che abbiamo a disposizione si possono distinguere diverse tipologie di algoritmi, al giorno d'oggi quelle maggiormente utilizzate sono tre: apprendimento supervisionato, apprendimento non supervisionato e apprendimento di rinforzo.
L'**apprendimento supervisionato** è la tipologia di algoritmi maggiormente utilizzati e trova applicazione in vari campi come riconoscimento delle immagini, elaborazione del testo e riconoscimento delle azioni. In questo caso l'addestramento viene fatto utilizzando dati etichettati. È necessario fornire un numero sufficientemente grande e vario di esempi per permettere all'algoritmo di generalizzare i criteri di classificazione. Ad esempio, se si vuole creare un classificatore per le classi cane e gatto, sarà opportuno procurarsi un vasto assortimento di campioni che includano il maggior numero possibile di razze distinte di cani

e gatti, altrimenti l'algoritmo potrebbe avere difficoltà di classificazione ogni volta che incontra una razza sconosciuta. Nella Figura 2.1 possiamo vedere uno schema generalizzato di apprendimento supervisionato. Abbiamo un training set, l'insieme di esempi etichettati, che viene sottoposto all'algoritmo, il quale osservando i campioni è in grado di estrarre un modello che potrà successivamente essere utilizzato per classificare campioni privi di etichetta.

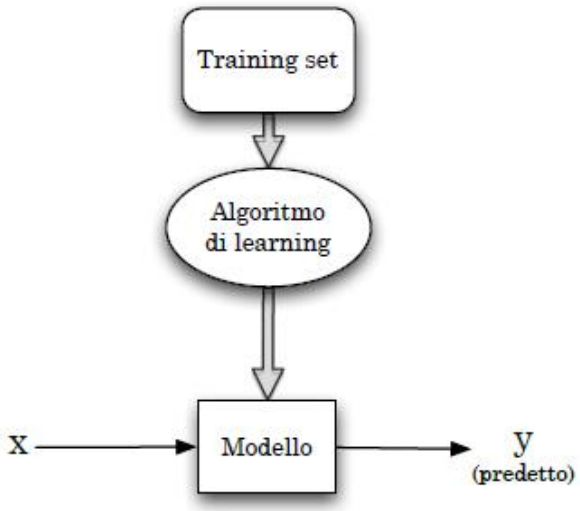

Figura 2.1: *Schema generale apprendimento supervisionato*

In questa categoria di algoritmo possiamo distinguere tra due tecniche principali: di classificazione se l'output ottenuto è una classe, una categoria come nell'esempio del cane e del gatto, o di regressione se l'output ottenuto è un numero reale come potrebbe essere l'esempio di un algoritmo che prevede il prezzo di vendita di una casa. Alcuni esempi di apprendimento supervisionato sono classificatori binari, classificatori multi-classe, regressione lineare e logistica e le macchine a vettori di supporto (SVM).
L'**apprendimento non supervisionato** è in grado di lavorare e apprendere da dati non etichettati, ciò permette di evitare l'enorme sforzo umano richiesto per organizzare ed assegnare le etichette a tutti i dati come nel caso dell'apprendimento supervisionato. Questa tecnica è in grado di esaminare i dati, individuare alcune caratteristiche comuni e suddividerli sulla base di queste caratteristiche. Facendo nuovamente l'esempio dei cani e dei gatti, in questo caso ci troveremo di fronte ad un insieme di immagini di cani e gatti non etichettati, l'algoritmo sarà questa volta in grado di dirci se due immagini di un cane sono simili tra loro ma diverse da quella di un gatto. Sfruttando qualche forma di similarità sarà così in grado di raggruppare i dati anche se non conosce quale gruppo rappresentano. Le due tipologie principali di apprendimento non supervisionato sono il clustering e la riduzione della dimensionalità dei dati. Con clustering si indicano gli algoritmi che analizzano un dataset e lo suddividono in gruppi contenenti elementi simili tra loro, mentre con riduzione della dimensionalità si indicano tecniche per semplificare i dati senza perdere troppe informazioni. Sono molto utili in fase di preelaborazione dei dati per ridurne la dimensionalità prima di

applicare altre tecniche. Supponiamo di voler predire il valore di una casa e di disporre delle seguenti caratteristiche: dimensione, numero di bagni, numero di camere da letto, distanza dalla scuola più vicina e distanza dall'ospedale più vicino. Se si vuole semplificare il dataset si possono raggruppare le classi simili in un'unica classe più generica. Ad esempio, le prime tre potrebbero rientrare sotto la categoria dimensione della casa e le ultime due in distanza dai servizi. Queste tecniche si occupano di svolgere questo compito minimizzando il più possibile il numero di informazioni perse.
L'**apprendimento di rinforzo** è un tipo differente di algoritmo di machine learning, nel quale non sono utilizzati dati, ma si hanno un ambiente e una macchina. La macchina si muove all'interno dell'ambiente e attraverso dei sensori riceve una serie di feedback dall'ambiente, che possono essere positivi o negativi. Inizialmente i movimenti della macchina saranno casuali, e verranno man mano modificati in base ai feedback ricevuti, con la tendenza a replicare quelli che sono stati valutati positivamente e a evitare quelli che hanno ricevuto un feedback negativo.

2.2 Perceptron

L'antenato del machine learning è sicuramente il Perceptron [6]. Introdotto per la prima volta da Frank Rosenblatt nel 1958, è un algoritmo classificatore binario che utilizza il metodo di apprendimento supervisionato. La sua strutturata prende ispirazione dai neuroni del cervello umano. Come si può vedere dalla Figura 2.2, il Perceptron prende in input un vettore che rappresenta le feuture che vengono utilizzate per distinguere le classi e un vettore dei pesi che indica l'importanza da assegnare a ciascuna features. Pensiamo ad esempio di voler distinguere tra moto e macchine e di usare come features altezza, larghezza e colore. Se nelle immagini che usiamo per il suo addestramento abbiamo sia una moto che una macchina rosse, al colore sarà associato un peso molto basso perché non è riconosciuta come una features discriminante ai fini della scelta tra moto e macchina. Al suo interno il Perceptron effettua la somma dei prodotti tra features e relativi pesi. Il risultato ottenuto viene poi passato attraverso una funzione di attivazione, nel caso più semplice la funzione step, che se il risultato è maggiore di zero lo assegna alla classe 1, altrimenti alla classe -1. Uno dei limiti più grandi di questo sistema, è che è in grado di riconoscere solamente funzioni linearmente separabili.

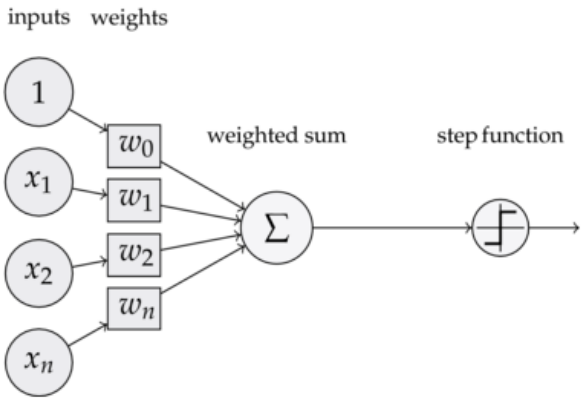

Figura 2.2: *Struttura Perceptron*

2.3 Deep Learning e Reti Neurali

Continuando ad ispirarsi alla struttura del cervello umano, dopo il Perceptron che si ispirava al singolo neurone, vennero introdotte le reti neurali. Con rete neurale si intende una struttura composta da più livelli interconnessi tra loro, e ciascuno formato da un certo numero di neuroni artificiali (Perceptron), come si può vedere dalla Figura 2.3. Il modello più semplice che può essere considerato è quello formato da due soli strati, uno di input e uno di output, ma solitamente si utilizzano modelli più complessi, con ulteriori strati intermedi chiamati hidden layers. Più il problema che dobbiamo risolvere è complesso, più è gande il numero di layers necessari. Quando consideriamo reti composte da più livelli, parliamo di Deep Learning.

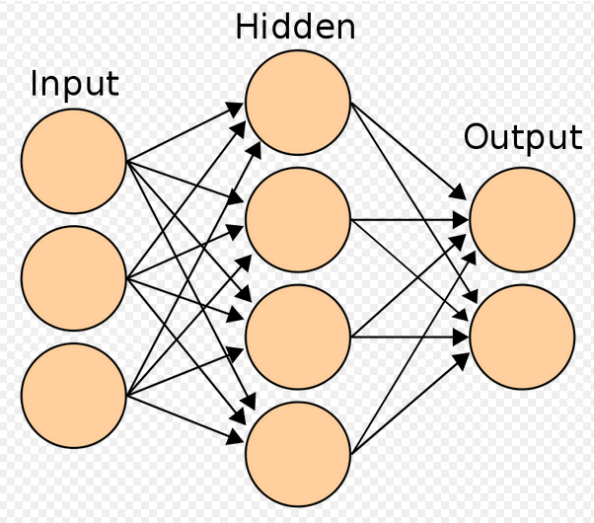

Figura 2.3: *Struttura rete neurale*

Per poter funzionare al meglio, questo tipo di tecnologia ha bisogno di grandi quantità di dati da analizzare e di un elevato quantitativo di risorse computazionali. Negli ultimi anni la disponibilità di questi due elementi è notevolmente aumentata e di conseguenza anche l'utilizzo e lo sviluppo delle reti neurali. I casi di applicazione classici sono il riconoscimento di testo, immagini e voce ma anche sequenza video come nel caso che andrò ad analizzare successivamente.
Sono disponibili diverse strutture di reti neurali, a seconda dello scopo per cui vengono utilizzate:

- Reti feed forward [7], possono condurre le informazioni in una sola direzione di elaborazione. Non presentano cicli e non hanno memoria di input avvenuti a tempi precedenti.

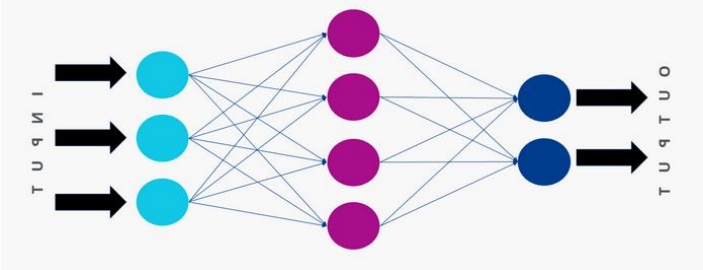

Figura 2.4: *Rete feed forward*

- Reti ricorrenti (RNN) [8], i valori di uscita di uno strato di un livello superiore vengono utilizzati come ingresso ad uno strato di livello inferiore. Questo tipo di struttura consente al sistema di creare una memoria. Esistono diversi tipi di reti ricorrenti, tra le più significative abbiamo le Long Short-Term Memory (LSTM), che sono in grado di apprendere da lunghe sequenze temporali e conservarne la memoria

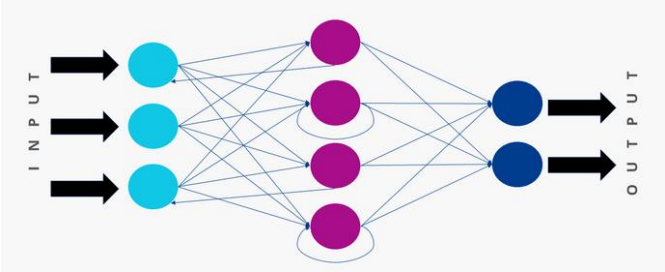

Figura 2.5: *Rete ricorrente*

- Reti convoluzonali [9], sono reti feed forward, composte da un livello di input, uno di output e alcuni hidden layers che comprendono particolari livelli come quelli convoluzionali, di attivazione o di pooling. Sono molto utilizzate nell'ambito della computer vision e rappresentano lo stato dell'arte per molte applicazioni come la classificazione delle immagini. È anche il tipo di rete che viene utilizzata in questo studio e per questo nel prossimo capitolo gli sarà dedicata una spiegazione più completa.

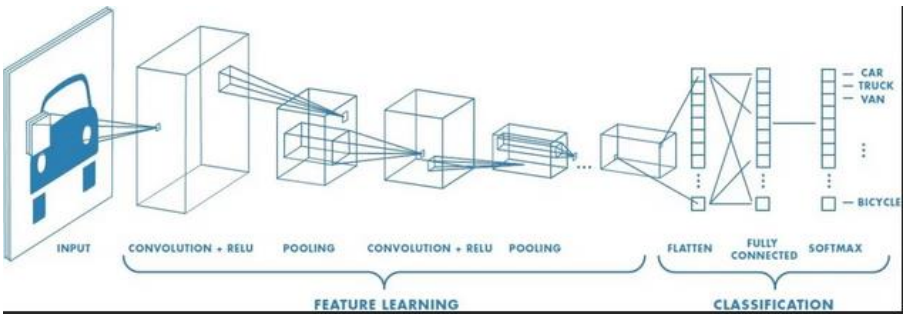

Figura 2.6: *Rete convoluzionale*

2.4 Reti convoluzionali

Le reti convoluzionali sono tipicamente molto profonde, questo perché ogni livello che le compone, è in grado di riconoscere solo una specifica caratteristica dell'immagine. I primi livelli si occupano delle features di più basso livello come angoli e linee, mentre proseguendo con i livelli si arriva a distinguere parti intere dell'immagine come ruote o parti del corpo umano.

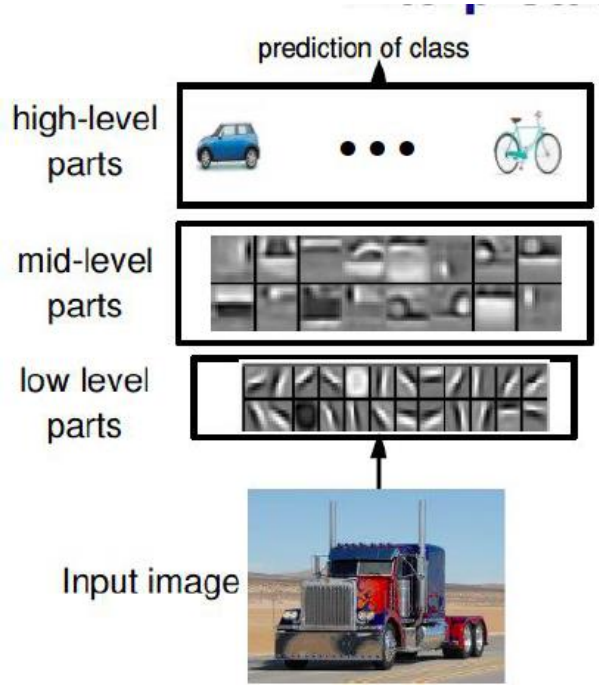

Figura 2.7: *Esempio estrazione features*

Le reti convoluzionali eseguono principalmente quattro operazioni: convoluzioni con l'utilizzo di filtri, applicazioni di non linearità, pooling e normalizzazioni. Perché tutto ciò possa avvenire correttamente, è necessario che ci sia una sequenza di livelli di diverso tipo: convolutional layer, pooling layer, batch normalization layer e fully connected layer.

2.4.1 Convolutional layer

Un convolutional layer applica piccoli filtri ad una piccola porzione dell'immagine. Questa operazione viene iterata in modo tale da coprire l'intera superficie dell'immagine. Un filtro è chiamato così perché filtra l'immagine basandosi su specifiche features. Ad esempio, un filtro per gli angoli sarà in grado di individuare tutti gli angoli all'interno dell'immagine. Una delle dimensioni più utilizzate per i filtri è 3x3. Ciascun filtro viene fatto scorrere sull'immagine in modo da analizzarla tutta. C'è un parametro chiamato stride, che indica di quanto deve essere mosso il filtro prima di essere nuovamente applicato. Il risultato dell'applicazione di un filtro sarà un numero, tutti questi numeri messi insieme andranno a

comporre una matrice di output chiamata feature map. Nella Figura 2.8 possiamo osservare come viene applicato un filtro 3x3.

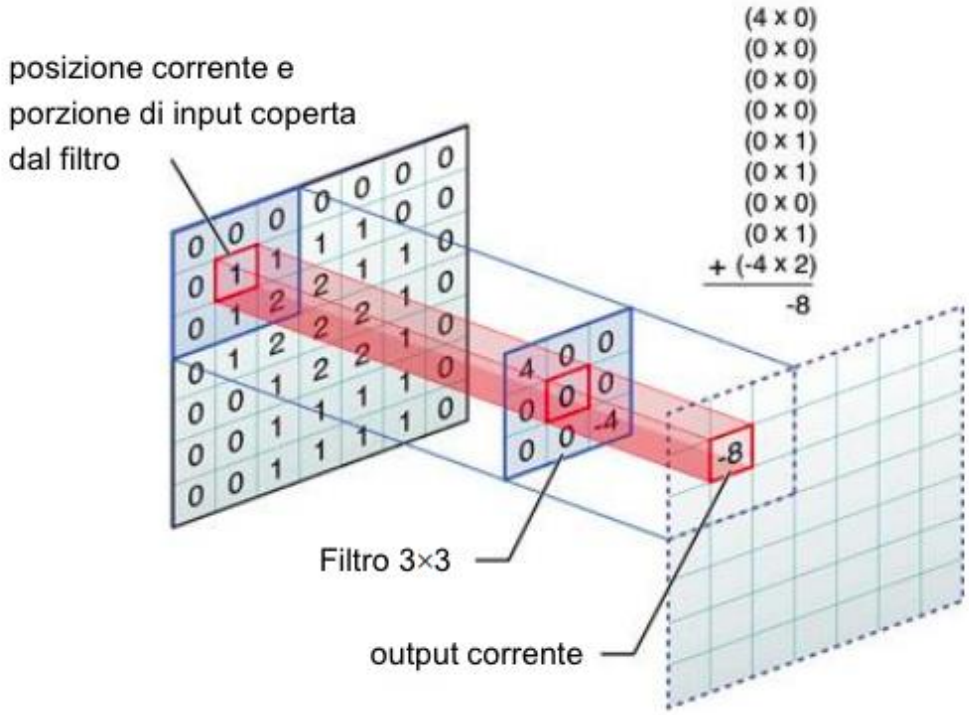

Figura 2.8: *Applicazione filtro 3x3*

In realtà non viene applicato solo un filtro, ma ad ogni livello convoluzionale si applicano più filtri per riuscire a distinguere più caratteristiche. Non sono disponibili solo filtri 2D ma possiamo utilizzarne anche a più dimensioni se è necessario. Ma che cosa significa concretamente applicare un filtro ? Significa effettuare una convoluzione, cioè una operazione tra due matrici che hanno le stesse dimensioni, in particolare è la somma delle moltiplicazioni tra ciascuna coppia di celle aventi gli stessi indici. Questa operazione viene quindi applicata tra la matrice del filtro e la matrice che rappresenta il pezzo di immagine che si considera in quel momento.

2.4.2 Pooling layer

Il pooling layer è un metodo per ridurre la risoluzione e rendere la rappresentazione più piccola e maneggevole. A differenza della convoluzione che agisce su tutto l'insieme delle feature map, questo tipo di layer opera su ciascuna di esse. Quando si effettua il pooling, si utilizza una matrice quadrata che viene fatta scorrere sui dati di input come viene fatto per i filtri. Anche in questo caso si utilizza lo stride per indicare di quanto spostare la matrice.

Esistono due principali tipi di pooling:

- Max pooling, è il più facile ed il più utilizzato. Ritorna il valore massimo presente all'interno della finestra considerata. L'idea alla base è quella di mantenere le informazioni rilevanti e scartare quelle meno significative.
- Average pooling, per diminuire il numero di informazioni scartate rispetto all'utilizzo del max pooling, si esegue la media dei valori presenti nella finestra considerata. In questo modo l'output dipende da tutto l'input e non solo da un valore di esso.

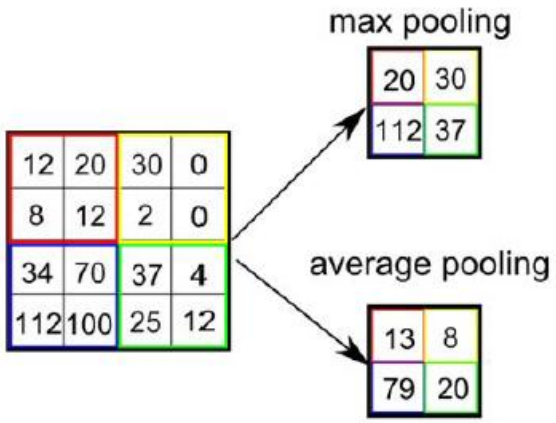

Figura 2.9: *Esempi di pooling*

2.4.3 Fully connected layer

Al termine di una rete convoluzionale, dopo l'ultimo livello di maxpooling viene sempre messo un fully connected layer. Deve il suo nome al fatto che ogni neurone che lo compone, riceve in input tutti i valori derivanti dal precedente livello, riprendendo la struttura del multilayer perceptron. L'obbiettivo di questo livello è di combinare tutti i dati ricevuti in input per assegnare l'etichetta all'immagine.

2.4.4 Batch normalization layer

Le reti neurali convoluzinali sono caratterizzate dal problema della normalizzazione. Quasi tutti gli algoritmi di machine learning lavorano meglio con dati normalizzati. Nonostante si normalizzino i dati in input, questi non lo sono più dopo ogni livello convoluzionale o fully connected. Questo problema rallenta i tempi di addestramento della rete e ne peggiora le prestazioni. La soluzione è quella di normalizzare i dati dopo ogni convolutional/fully connected layer. Ad occuparsi di questa operazione sono i batch normalization layers.

2.4.5 Funzione di attivazione

All'uscita di ogni convolutional layer viene associata una funzione di attivazione. Quella maggiormente utilizzata è la Rectified Linear Unit (ReLU) [10]. Con ReLU si intende una funzione di attivazione non lineare, che porta a zero tutti gli input minori di zero e lascia invariati quelli maggiori di zero. Questa soluzione è molto diffusa perché velocizza il processo di convergenza del modello e non è soggetta a saturazione per valori positivi elevati, come si verifica nel caso della funzione sigmoidea. Ha però un problema che in alcuni casi può risultare determinante, se i valori che riceve sono in parte negativi vengono perse delle informazioni e nel caso in cui siano ricevuti solo valori negativi, il neurone "muore" e non vengono più propagate informazioni. Un'alternativa è l'utilizzo della leaky ReLU [11] che invece di assegnare zero per ogni input negativo, assegna un valore negativo ma piccolo, andando ad evitare il problema della disattivazione dei neuroni.

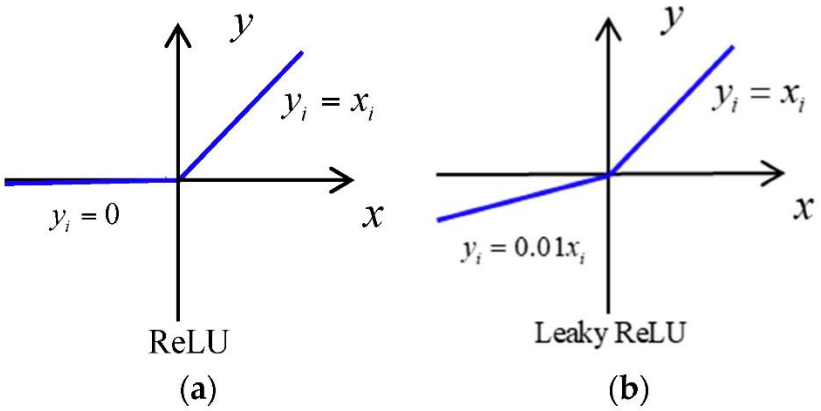

Figura 2.10: *Grafici ReLU e Leaky ReLU*

2.4.6 Training

Per poter funzionare, le reti convoluzionali hanno bisogno di essere addestrate. Per riuscire a effettuare questa operazione è necessario disporre di un insieme di dati, già etichettati correttamente, chiamato training set. Inizialmente abbiamo a disposizione una rete, a cui sono stati assegnati dei pesi in modo casuale, oppure estratti da qualche configurazione nota. Iniziamo la fase di training sottomettendo alla rete il primo dato preso dal training set. La rete elabora l'input e gli assegna un'etichetta di classe. La predizione fatta viene confrontata con quella corretta e se ne calcola la differenza, e di conseguenza vengono modificati i pesi della rete. Queste operazioni sono ripetute un gran numero di volte, su un gran numero di dati, in modo da migliorare sempre più la predizione.

Per calcolare la differenza tra output ottenuto e quello sperato, si utilizzano le funzioni di loss. Esistono varie tipologie di queste funzioni, vediamo alcuni esempi:

- L_2 loss, considera la metà del quadrato della differenza tra l'etichetta vera e quella predetta.

$$l\big(y, f(x)\big) = \frac{1}{2}(y - f(x))^2$$

- L_3 loss, considera il valore assoluto della differenza tra l'etichetta vera e quella predetta.

$$l\big(y, f(x)\big) = |y - f(x)|$$

- Cross entropy loss, è la funzione di loss che utilizzerò per l'addestramento della rete nella mia ricerca.

$$L(x, y, \theta) = -\sum_{c=1}^{M} y_{o,c} \log\left(p_{o,c}\right)$$

Dove:

- M è il numero di classi
- Y vale 0 o 1 a seconda che la classe c sia la classificazione corretta oppure no per l'osservazione o
- p è la probabilità che l'osservazione o sia della classe c

Una volta calcolata la loss è necessario andare a modificare i pesi della rete, per minimizzare la loss. Il processo utilizzato per questa operazione è chiamato backpropagation. L'idea è quella di mandare indietro nella rete dei feedback sui risultati ottenuti e aggiornare di conseguenza i pesi della rete. Viene utilizzata la tecnica del Gradient Descent sulla funzione di loss per individuare il minimo della funzione.

2.4.7 Iper-parametri

Nel processo di addestramento di una rete neurale, l'impostazione degli iper-parametri rappresenta una fase molto delicata, che andrà a influenzare tempi computazionali richiesti e performance del modello. Per selezionare il set migliore di iper-parametri, è necessario testare diversi valori e combinazioni, valutando di volta in volta l'accuratezza raggiunta dall'algoritmo con quello specifico insieme.
I principali iper-parametri sono:

- **learning rate**, indica di quanto cambiare il modello in base all'errore stimato, ogni volta che si si aggiornano i pesi. È difficile scegliere il giusto valore perché se è troppo basso può portare a tempi di addestramento eccessivamente lunghi, mentre se è troppo alto riduce i tempi di addestramento ma può portare a trovare un set si pesi non ottimale.
- **weight decay**, utilizzato per incrementare la capacità di generalizzazione del modello
- **batch size**, è possibile che il training set utilizzato sia troppo grande per essere elaborato tutto in una volta, a causa di limiti imposti da memoria e potenza di calcolo

disponibile. In questi casi il training set è suddiviso in gruppi più piccoli chiamati batch, composti da un numero di campioni detto batch size. Il batch size indica quindi il numero di elementi che vengono sottomessi alla rete prima di aggiornare i pesi. Questo parametro va scelto con cura perché un valore troppo elevato potrebbe portare alla saturazione della memoria, mentre uno troppo piccolo non permette di ottimizzare le performance. Valori tipici del batch size sono 32, 64, 128.

- **numero di epoche**, un'epoca corrisponde ad un passaggio attraverso la rete dell'intero training set. Un valore troppo basso di questo parametro potrebbe fermare la fase di training troppo presto, quando l'accuratezza sta ancora migliorando. Al contrario un valore troppo elevato può comportare uno spreco di tempo, infatti ad un certo punto l'accuratezza si stabilizza intorno ad un certo valore e proseguire con ulteriori iterazioni è inutile.

2.5 Altre tipologie di algoritmi di classificazione

Gli algoritmi di classificazione basati su reti neurali non sono gli unici a poter essere utilizzati, ne esistono infatti altre tipologie come K-Nearest Neighbors (KNN), Support Vector Machine (SVM) e Decision Tree.
Il KNN è un'algoritmo di apprendimento supervisionato, il cui scopo è quello di predire la classe di appartenenza di una nuova istanza, basandosi sulle caratteristiche degli oggetti vicini a quello considerato. Si basa su due parametri, la distanza e il parametro K. La distanza viene utilizzata per determinare quanto si somigliano due campioni e quella maggiormente utilizzata è la distanza euclidea. Il parametro K indica invece il numero di oggetti più vicini da considerare per fare la previsione. L'algoritmo calcola la distanza tra la nuova istanza e tutti gli oggetti già presenti, considera le K distanze più piccole e poi assegna la nuova istanza alla classe che è maggiormente rappresentata all'interno delle K distanze. Nella figura 2.11 abbiamo un esempio di classificazione con KNN. Il campione rappresentato con il pallino verde è quello sotto osservazione. Se si considera K = 3 sarà assegnato alla classe dei triangoli rossi perché due su tre appartengono a questa classe, al contrario se si prende K = 5 il pallino viene assegnato ai quadrati blu.

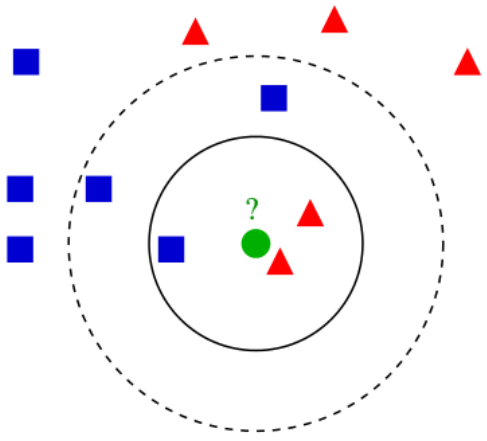

Figura 2.11: *Esempio KNN*

Anche le SVM sono un algoritmo di apprendimento supervisionato e vengono utilizzate in molti ambiti tra cui classificazione del testo e delle immagini. Questo tipo di algoritmo ottiene le prestazioni migliori nel caso di problemi di classificazione binaria. L'idea alla base delle SVM è quella di individuare un iperpiano che divida al meglio un set di dati in N classi. I vettori di supporto sono i punti dato più vicini all'iperpiano come si può osservare nella Figura 2.12 e se rimossi modificano la posizione dell'iperpiano.

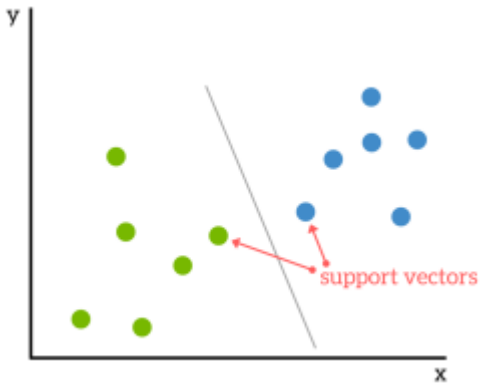

Figura 2.12: *Vettori di supporto*

L'altro elemento fondamentale per le SVM è il margine, che identifica la distanza tra i vettori di supporto di due classi differenti più vicini all'iperpiano.

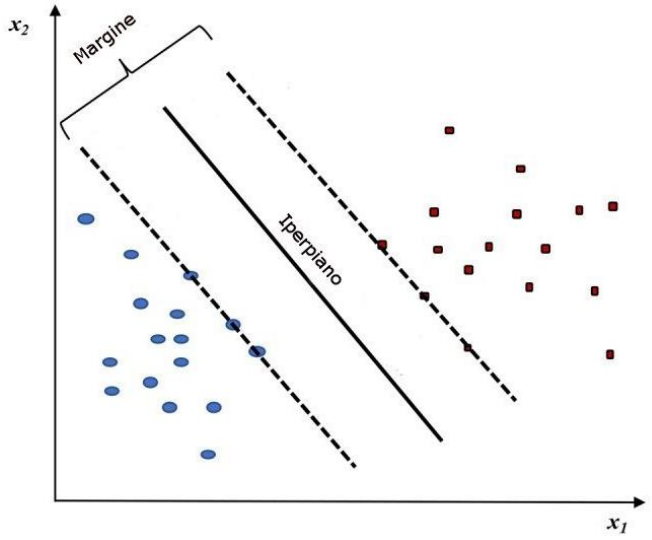

Figura 2.13: *Margine nelle SVM*

Per identificare l'iperpiano che meglio divide i vettori di supporto in classi, una SVM cerca un iperpiano linearmente separabile che separi i dati delle due classi. Se ne esiste più di uno

viene considerato solo quello con il maggior margine rispetto ai vettori di supporto, in modo da migliorare l'accuratezza del modello. Si possono anche utilizzare SVM con un numero di classi superiore a due, ma in questo caso la complessità aumenta ed è necessario utilizzare kernel che permettano divisioni del piano non lineari.
Il decision tree è anch'esso un algoritmo di apprendimento supervisionato e può essere utilizzato sia per problemi di classificazione che per problemi di regressione. Come si può vedere dalla Figura 2.14 questo tipo di algoritmo è descritto da una struttura ad albero i cui nodi foglia rappresentano le classificazioni mentre le ramificazioni rappresentano l'insieme delle proprietà che hanno portato a definire tali classificazioni. Per creare un albero di questo tipo è necessario definire quali caratteristiche scegliere, ossia i dati in input, quali condizioni usare per le divisioni e quando fermarsi nella definizione dell'albero. Per fare ciò sono disponibili diversi algoritmi tra cui l'algoritmo CART [12] e l'algoritmo C4.5 [13]. Una volta definito l'albero, il processo per assegnare un nuovo dato alla classe corrispondente è semplice: si parte sempre dal nodo radice e si procede verso il basso. Ad ogni nodo viene effettuato un test il cui risultato determina la direzione da prendere per proseguire, e si continua fino al raggiungimento di un nodo foglia che indicherà la classe di appartenenza. Ad esempio, consideriamo l'albero decisionale della Figura 2.14 e supponiamo di avere un dato da classificare con le seguenti caratteristiche: rimborso: no, stato civile: single, reddito imponibile: 100k. Si parte dal nodo radice, che fa riferimento all'attributo "refund", siccome nel caso preso in esame il valore di questo attributo è "no", si procede nel ramo di destra. Al nodo successivo, relativo allo stato civile si andrà a sinistra nel ramo che ha come valore "single" ed infine all'ultimo nodo, quello sull'attributo "taxable income" si scende verso destra, arrivando così ad un nodo foglia e di conseguenza alla decisione finale per il dato considerato.

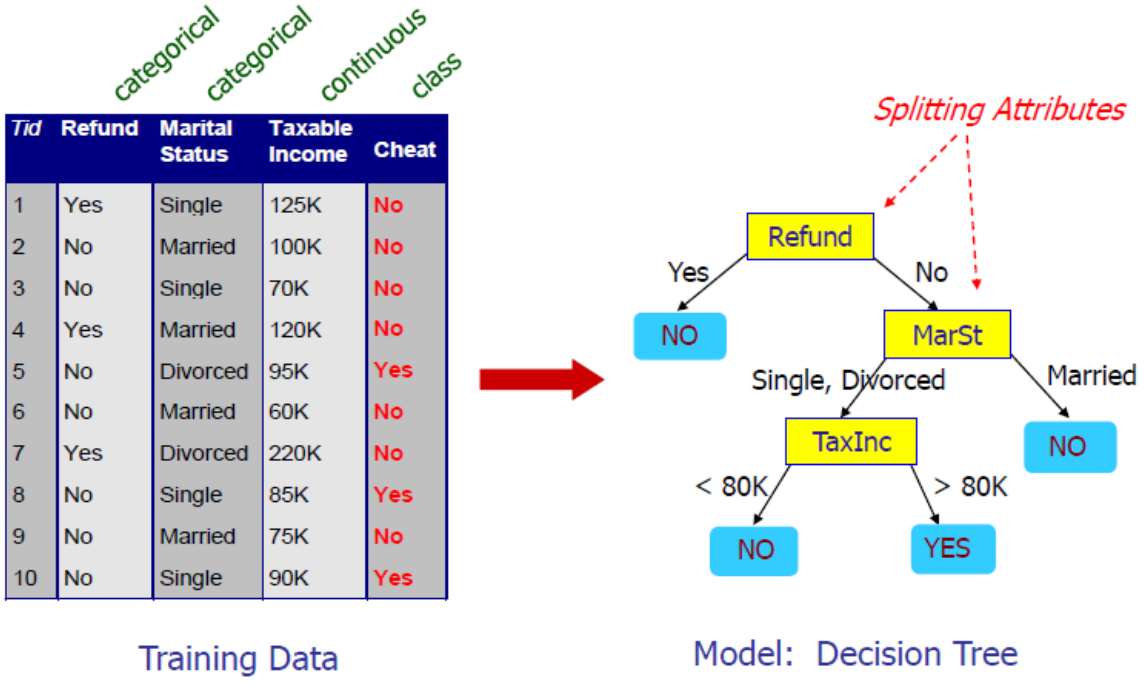

Figura 2.14: *Esempio albero decisionale*

Rispetto alle reti neurali gli alberi decisionali sono più facilmente comprensibili dagli esseri umani, e per questo è più facile verificare come si è giunti ad una decisone. Ad esempio, se consideriamo l'utilizzo di queste due tecnologie per fornire una diagnosi in ambito medico, è opportuno che ci sia sempre una verifica da parte di un medico, che sarà più facile se si utilizza un albero decisionale rispetto ad una rete neurale. Gli alberi decisionali sono invece poco adatti per problemi complessi, perché la complessità spaziale dell'algoritmo potrebbe diventare proibitiva.
Nonostante siano appena stati elencati diversi strumenti con i quali è possibile affrontare l'argomento trattato in questa tesi, si è scelto di utilizzare una rete neurale perché l'azienda che ha richiesto questo tipo di analisi era interessata all'utilizzo di questa tecnologia e inoltre, anche a livello personale ero fortemente interessato ad approfondire questa tipologia di algoritmo che non avevo avuto la possibilità di affrontare in modo così dettagliato durante il percorso universitario.

2.6 Metriche

Per verificare le prestazioni ottenute da un algoritmo di classificazione, possono essere utilizzate diverse tipologie di metriche, tra le più significative troviamo accuratezza, richiamo precisione e punteggio F1. Prima di passare alla descrizione di ciascuna di esse, è però opportuno introdurre uno strumento matematico chiamato matrice di confusione. Esso non è altro che un grafico che ci mostra le prestazioni del nostro modello, come si può vedere dall'esempio nella Figura 2.15. Le righe indicano quanti campioni ci sono per una determinata classe, in questo caso per la classe happy avremo 12+15+6 = 33 campioni, mentre le colonne indicano le predizioni che abbiamo fatto. Da questa rappresentazione possiamo facilmente ricavare per ciascuna classe i valori di true positive (TP), true negative (TN), false positive (FP) e false negative (FN). Considerando come esempio la classe happy, abbiamo un TP quando un campione viene predetto per quella classe ed effettivamente gli appartiene. Un TN invece si verifica quando un campione viene assegnato ad un'altra classe rispetto a happy ed effettivamente apparteneva a quella classe. Si parla di FP quando un campione viene predetto come happy ma in realtà non lo è, ed infine abbiamo un FN quando un campione appartiene alla classe happy ma viene assegnato ad un'altra classe. Questi quattro parametri sono la base per il calcolo delle metriche.

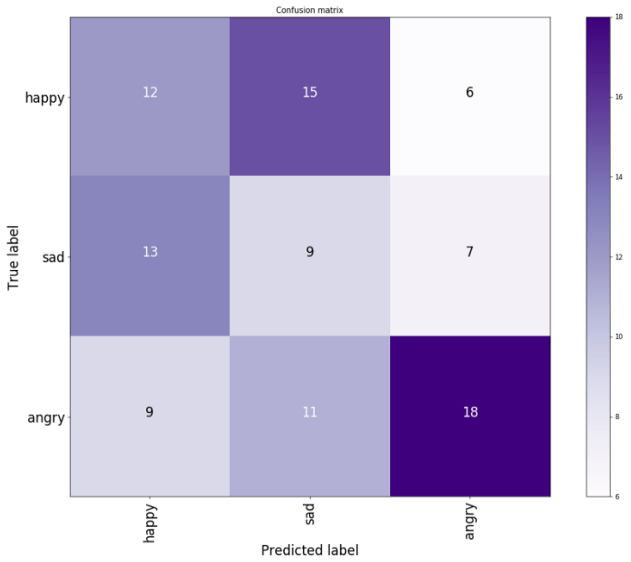

Figura 2.15: *Esempio confusion matrix*

L'accuratezza è una delle misure di performance più intuitiva, infatti è semplicemente il rapporto tra predizioni corrette e totale delle predizioni effettuate:

$$\text{Accuratezza} = \frac{TP+TN}{TP+TN+FP+FN}$$

La precisione è il rapporto tra le predizioni positive corrette e il totale delle predizioni positive e risponde alla domanda: di tutti gli elementi che sono stati dichiarati appartenere ad una determinata classe, quanti effettivamente le appartengono?

$$\text{Precisione} = \frac{TP}{TP+FP}$$

Il richiamo è il rapporto tra le predizioni positive corrette e tutti gli elementi effettivamente appartenenti a quella classe e risponde alla domanda: Di tutti gli elementi appartenenti ad una classe, quanti sono effettivamente stati correttamente etichettati?

$$\text{Richiamo} = \frac{TP}{TP+FN}$$

Il punteggio F1 rappresenta la media pesata di precisione e richiamo, perciò tiene in considerazione sia i falsi positivi che i falsi negativi. Questo tipo di metrica può essere più efficace rispetto all'accuratezza soprattutto nel caso in cui si abbia una distribuzione irregolare delle classi.

$$\text{Punteggio F1} = 2 * \frac{Richiamo*Precisione}{(Richiamo+Precisione)}$$

L’accuratezza solitamente lavora meglio quando il costo di FP e FN è simile, mentre se FP e FN hanno costi molto differenti è meglio considerare richiamo e precisione, quindi il punteggio F1. Per costo si intende quanto la presenza di un FP o FN sia significativa per il nostro modello, ad esempio se si sta utilizzando un modello per identificare le e-mail indesiderate, un FP sarebbe molto sconveniente perché potrebbe catalogare una mail molto importante come spam.

Capitolo 3

Stato dell'arte

In questo capitolo vengono presi in considerazione due articoli che rappresentano l'attuale stato dell'arte per quello che riguarda la human action recognition e che sono, per motivi diversi, strettamente collegati al lavoro svolto in questa tesi. Il primo per la tecnica utilizzata per il riconoscimento delle azioni umane, che prevede l'utilizzo di una rappresentazione dello scheletro umano, il secondo perché, pur utilizzando una tecnica differente, affronta il riconoscimento delle azioni umane per il gioco del tennis.

3.1 Action Recognition Based on 2D Skeletons Extracted From RGB Videos [14]

3.1.1 Introduzione

È stato preso in considerazione questo articolo perché utilizza un metodo molto simile a quello utilizzato per questa tesi. Il riconoscimento delle azioni umane viene eseguito attraverso l'estrazione di uno scheletro in 2D della persona, composto da 18 keypoint. Successivamente, la sequenza temporale delle posizioni dello scheletro viene trasformata in una immagine RGB. Questa viene poi data in input ad una rete neurale per il riconoscimento delle immagini. Si passa così da un problema di classificazione delle azioni, ad un problema di classificazione delle immagini, che ha una complessità minore.
Gli scheletri delle persone individuate vengono estratti tramite OpenPose [15]. Questa rete estrae la posizione di 18 keypoints per ogni corpo identificato e ritorna, per ogni keypoint, le coordinate (X,Y) ed un punteggio C che indica l'affidabilità della predizione.

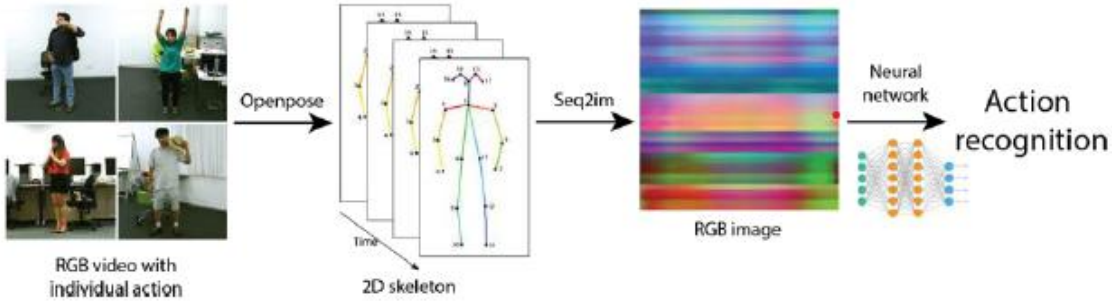

Figura 3.16: *Sequenza utilizzata nella rete [14]*

3.1.2 Estrazione dello scheletro con OpenPose

OpenPose è una rete in grado di individuare la postura di diverse persone all'interno di una immagine. Per riuscire in questa operazione viene utilizzato un approccio bottom-up: prima vengono individuate le parti del corpo e successivamente associate ai corrispondenti individui nella foto. Come si può osservare nella Figura 3.17, l'architettura proposta prevede una CNN con due flussi, quello superiore, contraddistinto dal colore rosa, specializzato nell'identificazione degli arti, e quello inferiore, contraddistinto dal colore azzurro, specializzato nel calcolo del grado di associazione tra parti del corpo ed individui presenti nella foto.

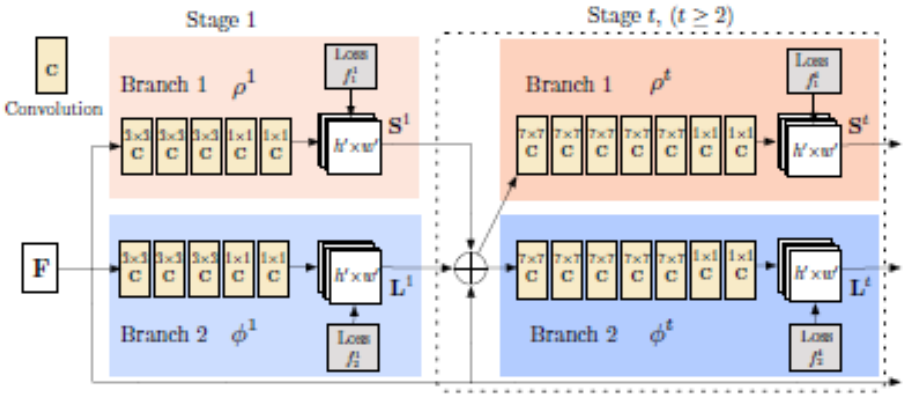

Figura 3.17: *Architettura OpenPose*

3.1.3 Dataset utilizzato

Il dataset preso in considerazione è NTU RGB+D [16] dataset, che contiene 60 differenti classi e 56880 video RGB, ma in questo caso sono state considerate solamente 49 azioni. Per ciascun video sono stati estratti gli scheletri degli attori principali utilizzando OpenPose, e nei casi in cui sono stati identificati gli scheletri di persone in secondo piano che nulla hanno a che fare con l'azione da identificare, i loro dati sono stati rimossi.

3.1.4 Rappresentazione dati

Le reti neurali per il riconoscimento delle immagini hanno avuto una grande evoluzione negli ultimi anni, raggiungendo risultati sorprendenti. I modelli disponibili sono molti e le performance raggiunte sono ottime, ciò ha portato gli autori di questo studio a decidere di rappresentare i movimenti dello scheletro attraverso un'unica immagine RGB. Una volta ottenuti gli scheletri da ciascun frame attraverso OpenPose, e quindi i valori delle coordinate (X,Y) e della confidenza C, questi valori sono salvati all'interno di tre matrici. Per ognuna, una riga corrisponde al variare del dato nel tempo, mentre una colonna al valore corrispondente a ciascun nodo dello scheletro in un singolo frame. Queste tre matrici vengono

poi normalizzate tra 0 e 255 prima di essere unite per formare un'unica immagine RGB come mostrato nella Figura 3.18. La rappresentazione appena descritta non è l'unica possibile, in particolare possono variare il numero e l'ordine in cui vengono considerati i nodi dello scheletro e il valore che va a comporre la terza matrice, che non deve per forza essere C.
Negli esperimenti condotti in questo paper, verranno considerate diverse rappresentazioni e codifiche dei dati, per capire quale sia quella che garantisce le migliori prestazioni.
Per iniziare si sono interrogati su quale fosse, oltre alle coordinate (X,Y), la terza dimensione da tenere in considerazione ed hanno analizzato due differenti opzioni:

1. Per ogni keypoint vengono considerate le coordinate (X,Y) e il parametro C, che indica l'affidabilità della predizione, rispettivamente codificate nei canali R, G e B come mostrato nella Figura 3.18.

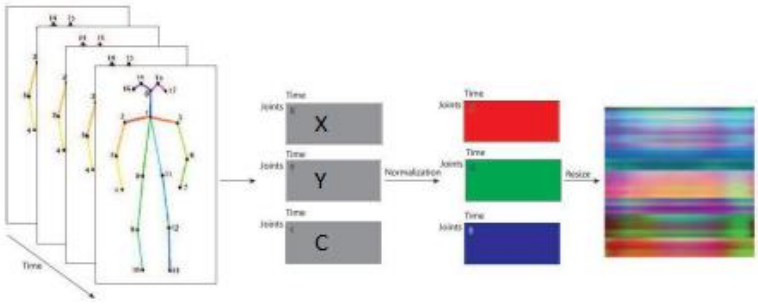

Figura 3.18: *Rappresentazione X,Y,C*

2. Per ogni keipoint vengono considerate le coordinate (X,Y) e il valore medio tra (X,Y) per rappresentare la profondità. Questo deriva dal fatto che le dimensioni del corpo nel video possono essere utilizzate per ricavare informazioni sulla vicinanza dell'attore. Più l'attore si allontana dalla telecamera e minori saranno le sue dimensioni, vice versa se si avvicina apparirà più grande. Quindi alcune informazioni sulla profondità sono contenute all'interno dei valori (X,Y) di ciascun nodo dello scheletro. Per questo motivo si propone di utilizzare la media tra i valori di (X,Y).

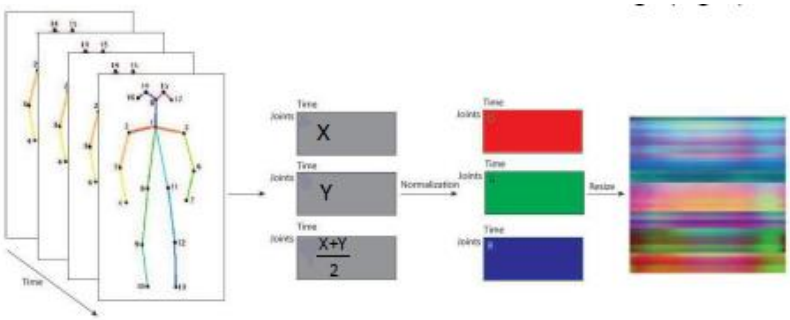

Figura 3.19: *Rappresentazione X,Y,* $\frac{X+Y}{2}$

Come seconda analisi è stata fatta una valutazione delle performance al variare del numero di keypoints utilizzati, passando da 18 a 14. Sono stati rimossi i punti relativi agli occhi e alle orecchie. La scelta è ricaduta su questi punti perché, per le azioni che dovevano essere valutate, non erano coinvolti e per di più nel 50% dei casi OpenPose non era in grado di individuarli. I test con 14 nodi sono stati ripetuti sia utilizzando la confidenza C che il valore medio tra (X,Y) come terza coordinata per il canale B.
Come ultimo caso è stato preso in considerazione un diverso ordine nella codifica dei keypoints, sempre utilizzandone 14. Se nei casi precedenti era stata utilizzata la sequenza: 0,1,2,3,4,5,6,7,8,9,10,11,12,13, in questo caso si passa alla sequenza: 0,4,3,2,1,5,6,7,10,9,8,11,12,13.

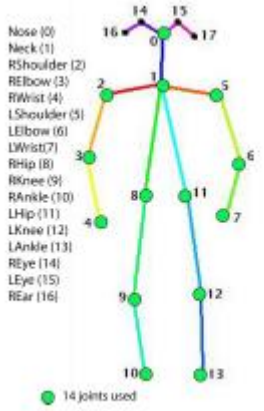

Figura 3.20: *Keypoints scheletro*

3.1.5 Esperimenti e conclusioni

Per valutare le prestazioni delle diverse configurazioni, è stato utilizzato il dataset NTU RGB+D e sono stati effettuati i test con due diversi modelli di reti neurali per la classificazione delle immagini: SqueezeNet [17] e DenseNet [18]. SqueezeNet è molto piccola e ha pochi parametri, ciò la rende molto veloce da addestrare. DenseNet invece è più complessa, ha un numero maggiore di parametri e per questo richiede più tempo per essere addestrata ma garantisce una migliore accuratezza rispetto alla SqeezeNet.
I test effettuati hanno evidenziato che in generale:

- è meglio assegnare al terzo canale RGB, il valore C che indica l'affidabilità della predizione, rispetto alla media tra le coordinate (X,Y).
- la rimozione dei keypoints di orecchie e occhi è significativa e nel caso di SqeezeNet ha migliorato le prestazioni.
- l'ordine in cui vengono considerati i keypoints è importante e la miglior sequenza trovata è: 0,1,2,3,4,5,6,7,8,9,10,11,12,13.

I due modelli utilizzati non concordano però sulla miglior rappresentazione. Come mostrato nelle Tabelle 3.1 e Tabella 3.2, con SqueezeNet la miglior rappresentazione prevede di usare le coordinate (X,Y) e il parametro C, 14 keypoints e la sequenza 0,1,2,3,4,5,6,7,8,9,10,11,12,13. Invece con DenseNet abbiamo sempre le coordinate (X,Y) e il parametro C ma stavolta con 18 keypoints così ordinati 14,15,16,17,0,1,2,3,4,5,6,7,8,9,10,11,12,13. L'unica differenza sta quindi nel numero di keypoints considerati.

Test	RGB conversion	Number of nodes	Node's order
1	(X,Y), C	18	Order 1
2	(X,Y), mean	18	Order 1
3	(X,Y), C	14	Order 2
4	(X,Y), mean	14	Order 2
5	(X,Y), C	14	Order 3
6	(X,Y), mean	14	Order 3
Legend			
Order 1 = (14,15,16,17,0,1,2,3,4,5,6,7,8,9,10,11,12,13)			
Order 2 = (0,1,2,3,4,5,6,7,8,9,10,11,12,13)			
Order 3 = (0,4,3,2,1,6,7,10,9,8,11,12,13)			

Tabella 3.1: *Elenco test*

SqueezeNet		DenseNet	
Test number	Accuratezza	Test number	Accuratezza
1	74.253%	**1**	**82.718%**
2	71.439%	2	80.338%
3	**74.553%**	3	82.711%
4	73.160%	4	80.981%
5	73.512%	5	82.688%
6	72.487%	6	80.735%

Tabella 3.2: *Confronto risultati*

Le configurazioni risultate migliori in termini di accuratezza, cioè la 1 e la 3, sono poi state utilizzate per effettuare dei test con reti più complesse. In particolare, con SqeezeNet, AlexNet [19], Inception [20], DenseNet, Resnet [21] e VGG [22].
Dai risultati ottenuti si può osservare che non è semplice decidere quale sia la miglior rappresentazione, se quella con 18 oppure con 14 keypoints. Per alcuni modelli (SqueezeNet, DenseNet e VGG13), eliminare alcuni keypoints può migliorare l'accuratezza in altri casi no. L'accuratezza migliore utilizzando 14 keypoints è stata ottenuta con ResNet152 ed è pari a 83.347%. Con 18 keypoints è stata invece raggiunta un'accuratezza dell'82.651% con DenseNet. I risultati ottenuti sono visibili nella Tabella 3.3.

Model (retrained deep)	Memory size (MB)	Parameters optimized	Accuratezza for Test 1 (%)	Accuratezza for Test 3 (%)
SqueezeNet	3	747633	75.778	75.803
AlexNet	228.8	57204593	74.545	74.051
Inception v3	98.2	24481346	81.895	81.528
DenseNet169	51.1	12566065	81.940	**82.651**
ResNet34	85.4	21309809	82.591	81.356
ResNet152	233.8	58244209	**83.347**	81.693
VGG13	516.6	129151601	79.096	78.871
VGG19	559.88	139770993	78.497	78.984

Tabella 3.3: *Confronto risultati*

3.2 RGB Video Based Tennis Action Recognition Using a Deep Historical Long Short-Term Memory [23]

3.2.1 Introduzione

È stato deciso di inserire questo articolo per due motivi pricipali: si occupa del riconoscimento delle azioni dei giocatori di tennis, pur se utilizzando un metodo differente rispetto a quello presente nella mia analisi e sfrutta il dataset THETIS [24] per il training e testing della rete. In questo caso viene proposto un metodo che sfrutta una CNN e una Long Short-Term Memory network migliorata per riuscire ad analizzare anche la componente temporale presente quando si lavora nell'ambito dell'action recognition.

3.2.2 Architettura proposta

I due aspetti principali che influenzano le prestazioni nell'ambito action recognition sono la rappresentazione dello spazio e del tempo. Per riuscire a modellarli entrambe nel migliore dei modi viene proposta una Long Short-Term Memory network migliorata, accompagnata da una convolutional neural network per il riconoscimento delle azioni del tennis. Gli elementi chiave introdotti in questo paper sono:

1. L'utilizzo di una recurrent neural network (RNN) composta da unità chiamate Long Short-Term Memory (LSTM) che sono in grado di rappresentare le sequenze video con un alto livello di precisione.
2. La LSTM proposta utilizza un sistema di pesi in grado si assegnare una maggiore importanza ai frames che sono più significativi al fine del riconoscimento dell'azione.
3. Siccome la memoria storica che si va a formare all'interno della rete può essere molto lunga e può contenere errori che si accumulano nel tempo, viene proposta una

tecnologia in grado di azzerare la memoria storica per eliminare gli errori presenti al suo interno.

4. La rete LSTM da sola non è sufficiente per questo viene utilizzata insieme ad una CNN che si occupa di estratte le features relative allo spazio.

La struttura completa è rappresentata nella Figura 3.21. Come si può osservare dall'immagine, il primo step consiste nel dare in input alla convolutional neural network, in questo caso una Inception V3 [26] pre-addestrata su ImageNet [27], ogni frame del video originale. In questo modo vengono identificate le caratteristiche spaziali delle immagini. Al passo successivo entra in azione il modello LSTM, per riuscire a catturare anche le caratteristiche nel dominio del tempo. Questa parte del modello prende in input, l'output precedentemente prodotto dalla CNN. In realtà il modello utilizzato non è un classico LSTM, ma un historical LSTM, costruito apposta per descrivere meglio le informazioni temporali come caratteristiche globali. Ogni modulo historical LSTM utilizza un sistema di pesi per produrre le features, utilizzando la combinazione dello stato al tempo t e di quello al tempo t-1. Questi moduli sono utilizzati per costruire una deep historical LSTM network, composta da 5 livelli, ciascuno dei quali composto da 30 moduli historical LSTM. Infine, l'output prodotto dalla deep historical LSTM network viene analizzato da un livello di tipo soft-max per classificare l'azione contenuta nel video.

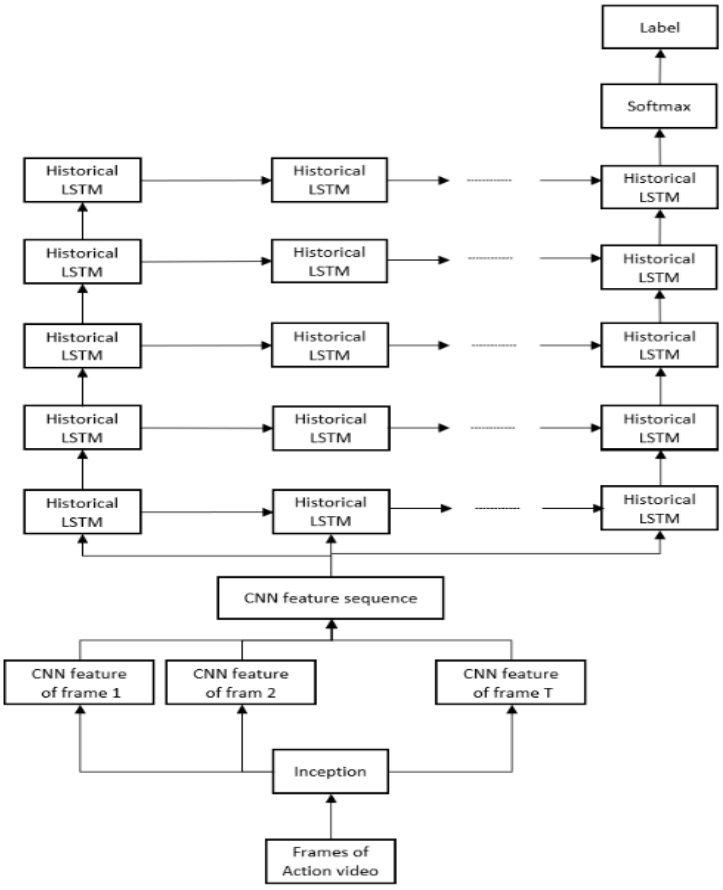

Figura 3.21: *Struttura LSTM network*

3.2.3 Esperimenti e risultati

Per dimostrare l'efficienza del nuovo modulo historical LSTM sono stati condotti esperimenti sia nel dominio dell'action recognition generica, sia specifica per il gioco del tennis. In entrambi i casi è stata utilizzata la tecnica del five-fold cross-validation per dividere il dataset in training e test set. Questa modalità prevede di separare il dataset in cinque gruppi, e a rotazione considerare un gruppo come test set e tutti gli altri come training set, addestrare e testare la rete di volta in volta con i diversi set e al termine considerare la media dei risultati ottenuti. Questo sistema garantisce una maggiore precisione nella valutazione dell'efficienza del modello ma richiede tempi più lunghi di esecuzione, tanto che non è sempre possibile utilizzarlo se si ha un dataset molto ampio.
Il dataset impiegato per l'analisi dell'action recognition generica è il HMDB51 dataset [25] che include 6849 video e rappresenta 51 diverse categorie di azioni. Quello utilizzato per il tennis è THree dimEnsional TennIs Shots (THETIS) dataset che contiene 12 diverse azioni base del tennis, eseguite da 31 dilettanti e 24 professionisti. Ogni giocatore esegue dalle 3 alle 4 volte lo stesso colpo, per un totale di 8734 video in formato AVI, di cui 1980 video RGB. Le azioni si svolgono in due diversi ambienti interni, che contengono diverse scene con persone che si muovono sullo sfondo. Per realizzare gli esperimenti sono stati utilizzati solo i 1980 video RGB.
In entrambe i casi è stato fatto un confronto tra la nuova historical LSTM proposta in questo articolo e la classica LSTM. Per la historical LSTM sono stati utilizzati 4 differenti valori di τ (2,3,4,5), un parametro che serve a regolare il numero di frame da tenere in memoria per ricostruire lo storico dell'azione.
Nella Tabella 3.4 sono riassunti i risultati ottenuti in merito all'action recognition sul dataset HMDB51.

Method	**Accuratezza**
Historical LSTM (τ =2)	0.73
Historical LSTM (τ =3)	0.62
Historical LSTM (τ =4)	0.63
Historical LSTM (τ =5)	0.62
LSTM	0.47

Tabella 3.4: *Risultati accuratezza*

L'accuratezza migliore, pari a 0.73, viene raggiunta quando il parametro τ è posto uguale a 2 mentre con il valore di τ fissato a 3 e a 5 l'accuratezza registrata scende a 0.62 e risale leggermente a 0.63 con τ impostato a 4. Con l'utilizzo della LSTM il risultato crolla ad un valore di 0.47. Risulta evidente che il nuovo modello di historical LSTM supera le performance ottenute con il modello LSTM.
Nella Tabella 3.5 è possibile osservare i risultati ottenuti in termini di accuratezza nel caso specifico del gioco del tennis.

Method	**Accuratezza**
Historical LSTM ($\tau = 2$)	0.70
Historical LSTM ($\tau = 3$)	0.74
Historical LSTM ($\tau = 4$)	0.71
Historical LSTM ($\tau = 5$)	0.63
LSTM	0.56

Tabella 3.5: *Risultati accuratezza*

Si nota come in questo caso le prestazioni migliori siano ottenute con un valore di $\tau = 3$ che raggiunge lo 0.74 di accuratezza. Abbiamo risultati leggermente più bassi per gli altri valori di τ, in particolare con $\tau = 5$ si scende a 0.63. Le evidenze raccolte durante l'analisi dell'action recognition generica, vengono confermate anche nel dominio del tennis, dove la LSTM raggiunge un'accuratezza pari a 0.56 sottolineando nuovamente la superiorità dell'historical LSTM introdotta in questo studio.

Capitolo 4

Tennis Action Recognition

Questo è il capitolo in cui vengono esposti i dettagli del lavoro svolto. Inizialmente con una breve introduzione ed esposizione del metodo e del documento utilizzato come punto di partenza per la mia analisi. Andando a spiegare quali dataset, rappresentazioni dei dati e reti sono state prese in considerazione. Per passare poi agli esperimenti condotti e ai risultati ottenuti.

4.1 Introduzione

Questo studio si colloca all'interno del vasto mondo dell'action recognition tramite l'utilizzo di CNN, ed in particolare prende in considerazione l'applicazione di questa tecnologia per il gioco del tennis. Si vogliono mettere le basi per la realizzazione di un'applicazione che permetta di analizzare i colpi effettuati da un giocatore in tempo reale, in modo tale da essere di supporto per gli allenamenti. La CNN che si è scelto di utilizzare sfrutta una rappresentazione dello scheletro umano, per analizzare e distinguere i diversi colpi effettuati e gli esperimenti che si vogliono effettuare, mirano a trovare la miglior combinazione di nodi dello scheletro da utilizzare. La bontà della rappresentazione utilizzata verrà calcolata tenendo conto delle prestazioni in termini di accuratezza e tempo di esecuzione. Il punto di partenza è uno scheletro composto da 15 keypoint che poi verranno gradualmente ridotti passando a sei, nel caso della rappresentazione delle sole braccia, e successivamente a quattro andando a rappresentare solo parzialmente i punti delle braccia come polsi e gomiti o polsi e spalle. Terminata questa parte dello studio si proverà anche ad inserire una rappresentazione della racchetta, da utilizzare da sola, oppure in aggiunta allo scheletro umano, per vedere quale impatto possa avere sulle prestazioni. La racchetta sarà identificata da quattro punti che corrispondono agli angoli del rettangolo che la contiene e che è ricavato dalla rete che esegue la object detection, oppure dal punto centrale del rettangolo stesso.
Gli strumenti di sviluppo utilizzati per questo lavoro sono il linguaggio di programmazione Python insieme alla libreria open source Pytorch. Questa libreria è specifica per l'implementazione di applicazioni nell'ambito machine learning ed è una delle più complete in questo settore.

4.2 Simple yet efficient real-time pose-based action recognition [28]

4.2.1 Introduzione

Si è scelto di prendere come riferimento questo articolo, perché implementa un sistema di riconoscimento delle azioni semplice ad efficiente che può essere utilizzato in real time, caratteristica fondamentale per il campo di applicazione del mio studio.
Il loro sistema prevede l'utilizzo di una sequenza di reti con differenti compiti, ognuna delle quali prende in input il risultato prodotto dalla rete precedente.
Come si può vedere dalla Figura 4.22 i diversi step sono:

1. Si produce l'immagine da analizzare attraverso una telecamera
2. Per ogni immagine si identificano le bounding box all'interno delle quali sono presenti le persone
3. Per ogni persona identificata viene estratto lo scheletro che la rappresenta
4. Si esegue il traking degli scheletri trovati, per riuscire ad associare ogni scheletro con il suo corrispondente nei frames successivi e precedenti
5. Si effettua l'action recognition

Figura 4.22: *Pipeline per action recognition*

4.2.2 Object detection

Per effettuare la object detection viene utilizzato YoloV3 [29], pre-addestrato sui dataset ImageNet e MSCOCO [30]. Questo sistema rappresenta un buon compromesso tra velocità ed accuratezza. Riceve in input un'immagine RGB, va alla ricerca della posizione degli oggetti all'interno dell'immagine e una volta trovati restituisce le coordinate della boundng box che li contiene insieme all'etichetta identificativa dell'oggetto.

4.2.3 Pose estimation

La pose estimation è quella che si occupa di estrarre lo scheletro della persona, che sarà poi fondamentale per distinguere le varie azioni. Viene utilizzato l'approccio preso dal paper Xiao et al. [31], anche qui con un pre-addestramento, stavolta sui dataset MSCOCO e MPII [32]. In questo caso come input vengono prese le bounding box identificate nello step precedente, mentre l'output è rappresentato dalle coordinate (X,Y) ed un coefficiente di probabilità per ciascun keypoint dello scheletro.

4.2.4 Human traking

Fino a questo momento sono state effettuate operazioni solo sui singoli frame, ma per poter distinguere le diverse azioni, c'è bisogno di analizzare più frames in successione. Per questo è necessario utilizzare il terzo step, lo human tracking. Questo strumento permette di mettere in relazione gli scheletri su diversi frame. Quindi di determinare se uno scheletro presente nel frame al tempo t, deve essere analizzato insieme ad un altro scheletro identificato nel frame al tempo t-1. Viene utilizzata un'implementazione piramidale di Lukas Kanade Feature Tracker [33]. Per riuscire ad effettuare il traking in modo corretto bisogna eseguire due step: verificare che non siano stati estratti due scheletri per la stessa persona nello stesso frame e mettere in relazione in modo corretto gli scheletri su frame consecutivi. Per fare questo viene calcolato un coefficiente di similarità nel seguente modo:
consideriamo *a* e *b* due scheletri e consideriamo Δ_a come la distanza massima tra due keyponts *i*, appartenenti a due scheletri, per poterli considerare parte dello stesso scheletro. Δ_a viene calcolato utilizzando l'altezza h_a e la larghezza w_a della bounding box della persona.

$$\Delta_a = F(\sqrt{w_a^2 + h_a^2}) \tag{1}$$

Il fattore F è un iper-parametro che va settato a 0.025, perché provato sperimentalmente.
Dopo viene calcolata la distanza Euclidea δ_{ab_i} del keypoints *i* tra li scheletri *a* e *b*.

$$\delta_{ab_i} = ||a_i - b_i||_2 \tag{2}$$

Questa distanza è considerata solo se *a* e *b* contengono il keypoints *i* con un valore minimo della probabilità T_j che indica la qualità minima che deve avere un keypoint per essere considerato. Si è verificato con la pratica, che un valore T_j = 0.4 funziona bene. Adesso si calcola la similarità S_{ab_i} per il joint *i*.

$$S_{ab_i} = \begin{cases} 1 - \left(\frac{\delta_{ab_i}}{\Delta_a}\right), & if\ \delta_{ab_i} < \Delta_a \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

La similarità totale tra gli scheletri *a* e *b* si calcola combinando le similarità di tutti i singoli keypoints.

$$S_{ab} = \frac{1}{T}\sum_{i=1}^{I}(S_{ab_i}) \quad (4)$$

Il fattore *I* indica il numero di keypoints usato per il tracking. A questo punto, utilizzando il valore della similarità totale, si vanno ad eliminare gli scheletri che sono troppo simili perché sono quelli che sono stati erroneamente individuati per la stessa persona nello stesso frame. Viene fissata una soglia $T_S = 0.15$. Se la similarità supera questa soglia, lo scheletro con lo score minore è rimosso dalla lista degli scheletri. Dopo aver fatto questo primo controllo bisogna mettere in corrispondenza gli scheletri del frame corrente con quelli dei frames precedenti. Se la similarità tra lo scheletro del frame corrente e uno scheletro dei frames precedenti è maggiore della soglia T_S, gli viene assegnato lo stesso id in modo da associare i due scheletri. Potrebbe capitare che alla fine di questo processo rimangano degli scheletri dei frames precedenti che non hanno ricevuto nessuna corrispondenza con scheletri del frame attuale. In questo caso si considera la bounding box corrispondente a quello scheletro e si prova ad effettuare una pose estimation, relativa a quella bounding box ma sul frame corrente e se viene identificato uno scheletro si effettua l'associazione. Questo viene fatto per compensare possibili falsi negativi prodotti dalla fase di object detection.

4.2.5 Pose-based action recognition

Rimane da considerare l'ultimo passaggio, la pose-based action recognition, che insieme alla fase di pose estimation, saranno quelle su cui si concentrerà la mia analisi. A differenza del paper [23], riportato nel capitolo sullo stato dell'arte, dove veniva utilizzata una LSTM network, qui viene utilizzata una convolutional neural network per il riconoscimento delle immagini.

4.2.5.1 Struttura dell'input

L'idea è quella di codificare i movimenti dello scheletro nel tempo, attraverso un particolare tipo di immagini chiamate Encoded Human Pose Image (EHPI). Come vengono prodotte? Il processo è illustrato nella Figura 4.23. Una volta estratto lo scheletro con i corrispondenti keypoint, l'idea è quella di far corrispondere ai valori *x*, *y* e *z*, i valori di rosso, verde e blu di un'immagine RGB. In questo caso si è deciso di lavorare solo in 2D e quindi la terza coordinata è sempre posta uguale a zero.
La struttura di un EHPI sarà così composta: ogni riga rappresenta un keypoint, ogni pixel di una riga rappresenta la posizione del keypoint ad un certo istante di tempo t. Quindi sposandosi da sinistra verso destra si osserva l'evoluzione della posizione dei keypoint, col passare del tempo.

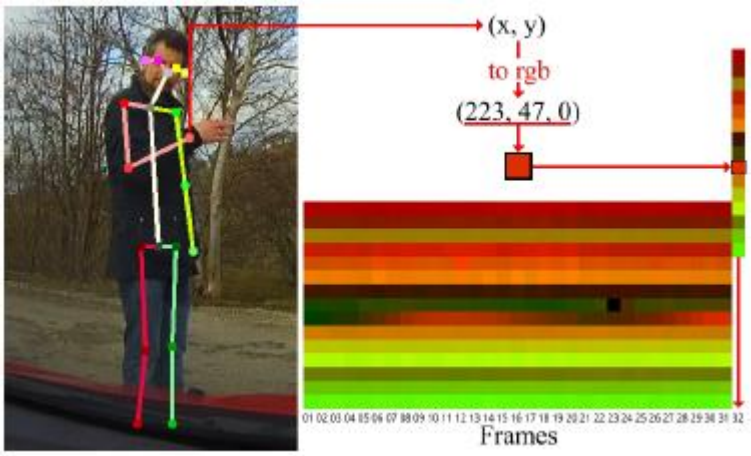

Figura 4.23: *Creazione EHPI*

Prima di essere convertite nei corrispondenti colori, le coordinate x e y vengono normalizzate su valori continui tra 0 e 1. Eventuali coordinate, che risultano essere al di fuori delle dimensioni dell'immagine, vengono poste a zero. Lo stesso vale per i keypoints che non vengono riconosciuti o che hanno una probabilità al di sotto della soglia T_J. Inoltre, un EHPI per essere considerato valido, deve avere almeno due frame nei quali è stato riconosciuto uno scheletro.
Si possono considerare diverse configurazioni di keypoints da utilizzare, per questo studio sono stati considerati i seguenti nel seguente ordine: naso, collo, centro delle anche, spalla sinistra, gomito sinistro, polso sinistro, spalla destra, gomito destro, polso destro, anca sinistra, ginocchio sinistro, caviglia sinistra, anca destra, ginocchio destro e caviglia destra.
Ogni EHPI comprende la rappresentazione di 32 frame.

4.2.5.2 Rete

Per classificare gli EHPI viene utilizzata una rete molto semplice composta da sei livelli convoluzionali, di cui ognuno ha un kernel 3x3 e un padding e uno stride di 1. Un livello fully connected è posizionato alla fine della rete per la classificazione. La struttura completa si può vedere nella Figura 4.24. Ciascun livello convoluzionale è seguito da una batch normalization. La funzione di attivazione utilizzata è una ReLu. Dopo la seconda e la quarta convoluzione è posizionato un livello di max pooling con kernel 2x2 che riduce la risoluzione spaziale di un fattore 2. Dopo l'ultimo livello convoluzionale si applica un livello di global average pooling. Ogni livello convoluzionale è inizializzato con l'inizializzazione di Xavier [34]. Per addestrare e testare la rete utilizzano il dataset JHMDB [37] che comprende 928 video con 21 diverse etichette di classe.

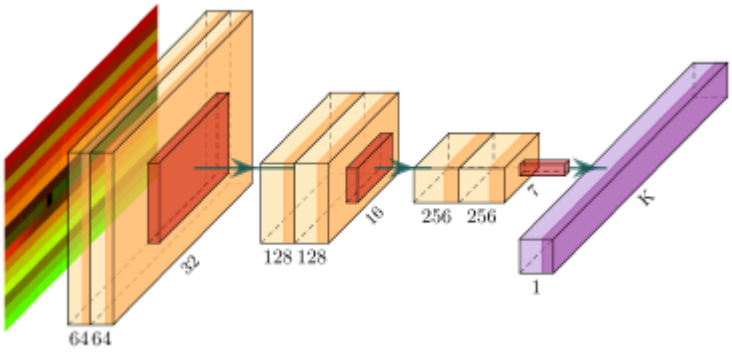

Figura 4.24: *Rete action recognition*

4.2.5.3 Esempi EHPI

Nella Figura 4.25 sono presenti gli esempi di tre EHPI relativi a tre differenti azioni: *idle*, *walk* e *wave*. La riga dell'EHPI che rappresenta il polso destro è messa in evidenza perché è fondamentale per distinguere le tre azioni. Nel caso di *idle* possiamo notare come il colore rimanga pressoché costante, questo perché il polso sinistro rimane quasi fermo e di conseguenza le sue coordinate non cambiano di molto. In *walk* invece, il colore passa progressivamente dal verde all'arancione, conseguenza del fatto che, il valore della x del polso destro aumenta spostandosi da sinistra verso destra e quindi aumenta anche la concentrazione del rosso. Il valore della y del polso destro invece rimane costante. Nell'ultimo esempio preso in considerazione, *wave*, si nota una ripetizione della transizione di colore da verde a rosso, perché il polso destro ripete più volte lo stesso movimento.

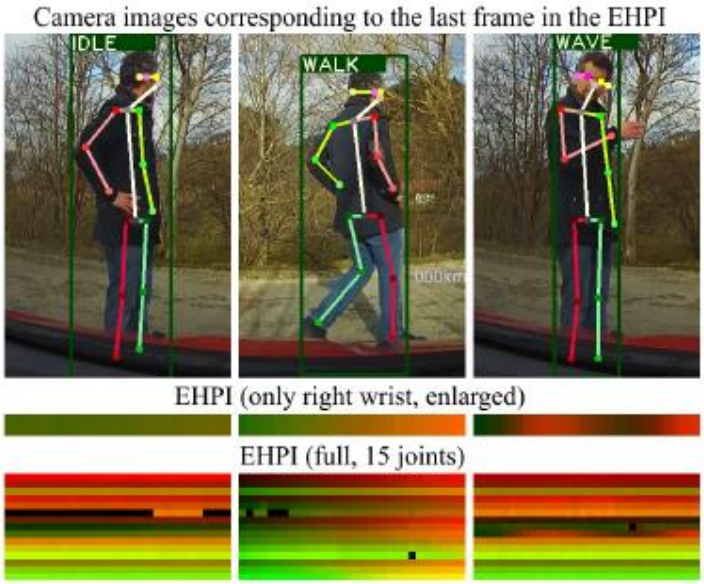

Figura 4.25: *Esempio EHPI*

4.3 Dataset

I dataset che sono stati utilizzati in questo lavoro sono:

- THETIS dataset, che è lo stesso utilizzato nell'articolo [23] presente nel capitolo 3 sullo stato dell'arte.
- Un dataset molto più piccolo, che ho creato io partendo da due video che mi sono stati forniti.

4.3.1 THETIS

Questo dataset è completamente dedicato al tennis, contiene 12 diversi tipi di azioni: backhand with two hands, backhand, backhand slice, backhand volley, forehand flat, forehand open stands, forehand slice, forehand volley, service flat, service kick, service slice e smash. Ogni colpo è eseguito da 31 giocatori dilettanti e 24 esperti, per un numero di volte che varia da 3 a 4. In totale sono presenti 8734 video in formato AVI, di cui 1980 sono RGB video e sono quelli che andrò ad utilizzare. I colpi vengono riprodotti in due differenti ambienti, in uno dei quali sono anche presenti delle persone che si muovono sullo sfondo. Nella Figura 4.26 si possono vedere i frame estratti da due video del dataset con i due differenti ambienti.

Figura 4.26: *Esempi dataset*

Le dodici differenti classi di colpi disponibili sono state raggruppate in quattro classi base: dritto, rovescio, battuta e schiacciata. Questa scelta è stata fatta per effettuare una prima analisi di base, con un modello più semplice, per passare in futuro ad un'analisi più completa. Inoltre, lo scopo principale non è quello di riuscire a classificare in modo appropriato il maggior numero di colpi ma piuttosto di fare un'analisi su numero e tipologia di keypoint da utilizzare, per ottenere il miglior rapporto tra accuratezza e tempo di esecuzione.
Per poter utilizzare il dataset per l'addestramento e il testing della rete è stato diviso in tre partizioni come segue: 60% per il training, 20% per l'evaluation e 20% per il test. Avendo un totale di 1980 video, ne avremo 1188 per il training, e 396 sia per l'evaluation che per il testing. La suddivisione non è però avvenuta in maniera del tutto casuale, sono stati seguiti alcuni criteri per far sì che le valutazioni fossero il più equilibrate possibile. Innanzitutto, si

è tenuto in considerazione il fatto che, professionisti e dilettanti. andassero equamente distribuiti nelle tre partizioni. Questo perché, avere ad esempio tutti i professionisti concentrati nel test, avrebbe probabilmente condizionato le prestazioni della rete che sarebbe stata addestrata solamente con video di dilettanti. L'altro aspetto che si è voluto tenere in considerazione, è che ogni giocatore ripete lo stesso colpo dalle 3 alle 4 volte. Questo significa, che questi 3/4 video saranno molto simili tra loro. Quindi è stato deciso di mantenere sempre all'interno della stessa partizione, tutti i video che rappresentano lo stesso giocatore che esegue lo stesso colpo. Tutto ciò per evitare che video praticamente identici andassero a finire, in parte nel training set e in parte nel test set, e così facendo alterassero in positivo i risultati dei test. Tengo a precisare che non è certo che si sarebbe verificato questo fenomeno, ma per essere sicuri che non influisse sui risultati finali si è deciso di fare questa scelta. Infine, si è fatto in modo che ogni tipo di colpo fosse equamente distribuito nelle tre partizioni, quindi avremo il 60% dei dritti in quella per il test, il 20% in quella per l'evaluation e il 20% in quella per il test e così per tutti gli altri colpi.

4.3.2 Dataset personale

Questo è un dataset molto più piccolo che è stato creato partendo da due video di una partita di tennis. I due video in questione sono stati realizzati tramite una videocamera posta a fondo campo e facendo indossare al giocatore in primo piano un braccialetto per raccogliere informazioni sui movimenti della racchetta in modo da riconoscere il tipo di colpo effettuato. I dati raccolti sono stati appuntati in un file json, tra questi sono presenti l'istante in cui viene colpita la pallina e il tipo di colpo effettuato. Queste due informazioni sono state sfruttate per suddividere il video in clip, ognuna etichettata con la giusta classe e della durata di un secondo, con il colpo della pallina che avviene a metà video. Il risultato sono state 147 clip, di cui 95 dritti e 52 rovesci. Da questi sono stati eliminati ancora 8 dritti e 4 rovesci perché il giocatore risultava fuori dalla ripresa della telecamera. Quindi alla fine della selezione sono rimasti 87 dritti e 48 rovesci. Purtroppo, in questi video che sono stati forniti non sono presenti colpi di tipo battuta o schiacciata. Pur essendo di dimensioni ridotte, questo dataset permette di avere una rappresentazione di un caso reale di partita, e sarà utilizzato solo in fase di test per avere un ulteriore riscontro. Si rivelerà essere molto utile per alcune valutazioni sulle trasformazioni da applicare ai dati in fase di addestramento della rete.

Figura 4.27: *Esempio dataset personale*

4.3.3 Elaborazione

La rete dedicata alla human action recognition, per essere trainata e testata, ha bisogno di ricevere in input un file CSV così composto:

- una riga per ogni clip
- ogni riga contiene la rappresentazione di 32 frame
- ogni frame è composto da N keypoints
- ogni keypoints è identificato dalle coordinate x, y e dallo score

Ogni riga conterrà quindi 32*N*3 elementi. Nella Figura 4.28 è possibile osservare la rappresentazione di un singolo frame composto da N keypoints. Questa struttura sarà replicata sulla stessa riga una volta per ogni frame, nel nostro caso 32.

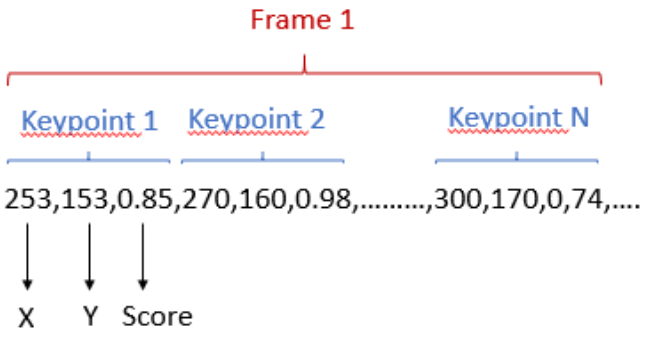

Figura 4.28: *Struttura file CSV*

Per riuscire a creare i files CSV nel modo corretto, tutti i video presenti all'interno dei dataset sono stati inizialmente suddivisi in 32 frames temporalmente equidistanti. Successivamente, da ciascun frame, sono stati estratti tutti gli scheletri delle persone presenti al loro interno ed è stato selezionato solamente lo scheletro del giocatore di tennis. La selezione è stato possibile farla in parte in modo automatico, sfruttando il fatto che spesso il giocatore fosse in primo piano e in una posizione centrale dell'immagine e in parte manualmente quando i vincoli sulla posizione non sono risultati sufficienti per identificare un solo scheletro. Dopo questo passaggio le coordinate dei keypoints sono state salvate sul file seguendo il formato descritto precedentemente. La selezione dello scheletro giusto da utilizzare è stata necessaria perché nella maggior parte delle clip, erano presenti persone che camminavano sullo sfondo. Le operazioni di creazione dei file CSV sono state ripetute per ogni diversa configurazione di keypoints che è stata adottata poiché, cambiando il numero N di keypoints considerati, variano anche i corrispondenti file CSV da sottomettere alla rete. I file CSV cosi costruiti ci permetteranno di ricavare gli EHPI da sottomettere alla rete, che variano a seconda del numero di nodi considerati, come si può notare nella Figura 4.29. In questo esempio abbiamo la rappresentazione di un dritto eseguito da un giocatore destro, in un caso utilizzando la rappresentazione completa dello scheletro con 15 punti, nell'altro sfruttando solo i punti delle braccia. I diversi colori assunti dai pixel indicano il movimento effettuato, nel caso specifico il passaggio dal verde al rosso, che è visibile soprattutto nella fascia centrale, indica un movimento da destra verso sinistra che corrisponde all'esecuzione di un dritto da parte di un destro. Alcuni quadratini risultano essere neri, ciò significa che in quell'istante di tempo, quello specifico keypoinits non è stato identificato dalla rete della pose estimation.

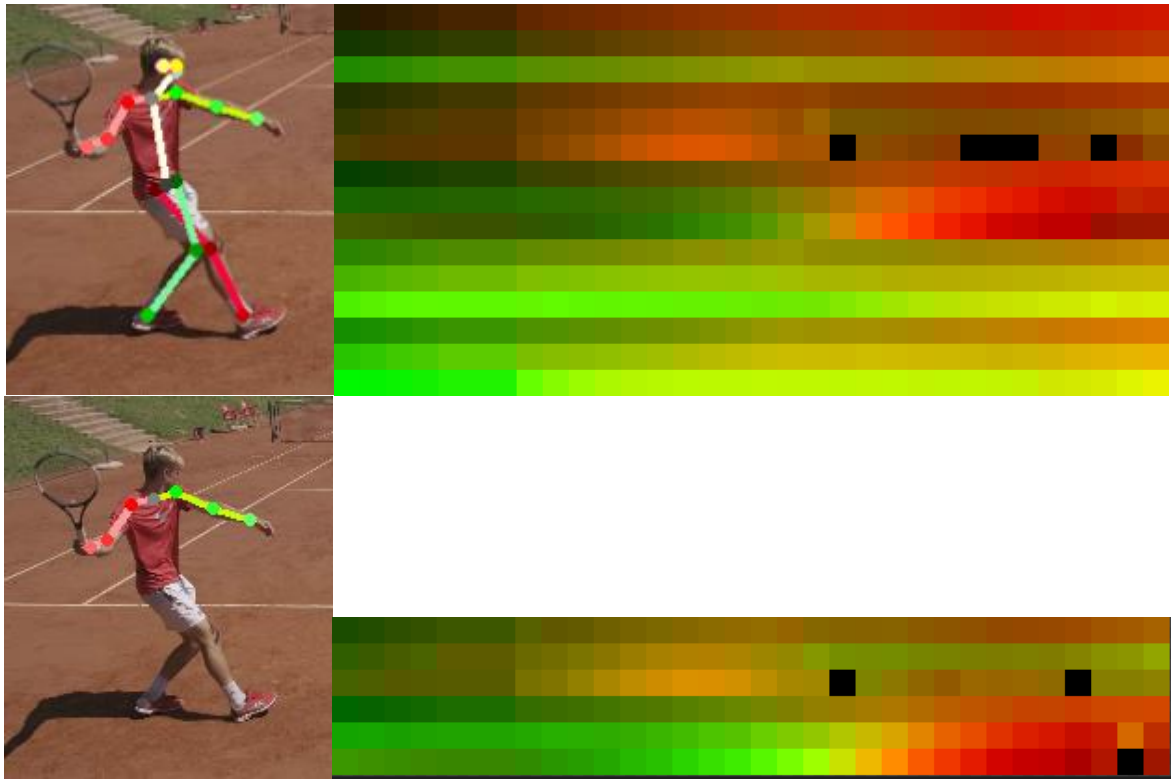

Figura 4.29: *EHPI tennis*

4.4 Valutazione dell'accuratezza

Prima di iniziare ad utilizzare la rete per effettuare la ricerca sulla configurazione migliore dei keypoints, questa è stata addestrata e successivamente testata, utilizzando diverse combinazioni di valori per gli iper-parametri, per decidere quale fosse la migliore. Per questa operazione è stato preso in considerazione il dataset THETIS e la versione di scheletro con 15 keypoints come nel paper [28].
I diversi valori utilizzati sono i seguenti: :

- Learning rate 0.05, 0.01, 0.005
- Weight decay 5e-4, 5e-3, 5e-2
- Batch size 32, 64, 128
- Numero di epoche 140, 200, 300

La combinazione migliore è risultata essere: learning rate = 0.005, weight decay = 5e-2, batch size = 32, numero di epoche = 300, ed è quella che sarà utilizzata da qui in avanti per tutti gli esperimenti. Per effettuare il training si è sempre partiti dal modello pre-addestrato sul dataset JHMDB tratto dal paper [27].

4.4.1 Esperimenti scheletro

In questa sezione vado ad illustrare le scelte fatte e le osservazioni derivate dalle diverse configurazioni di keypoints utilizzate. Come prima analisi sono stati considerati i 15 body joints utilizzati nel paper [28] sfruttando il loro medesimo ordine: naso, collo, centro delle anche, spalla sinistra, gomito sinistro, polso sinistro, spalla destra, gomito destro, polso destro, anca sinistra, ginocchio sinistro, caviglia sinistra, anca destra, ginocchio destro e caviglia destra. Dopodiché si è lavorato con altre combinazioni:

- Polsi, gomiti, spalle
- Polsi, gomiti
- Polsi, spalle
- Gomiti, spalle
- Polsi
- Spalle
- Gomiti

Negli ultimi tre casi, in cui si usano solo due keypoints, non è stato possibile eseguire alcun esperimento, perché avendo così pochi punti, l'EHPI risulta essere troppo piccolo e la rete non è in grado di elaborarlo.
Nel momento in cui si è provato per la prima volta ad eseguire il test con i 15 keypoints, inizialmente sul test set del dataset THETIS e poi sul dataset personale, è subito saltata all'occhio un'apparente anomalia: mentre nel primo caso si otteneva un'accuratezza dell'87.62%, nel secondo caso l'accuratezza crollava a 66.81%. Dopo un'accurata analisi sui due dataset, è stato appurato che il dataset THETIS era composto per oltre l'80% da giocatori mancini, mentre i due giocatori del mio dataset erano entrambi destri. Per verificare se fosse questa la causa del crollo dell'accuratezza è stata introdotta una nuova trasformazione, chiamata swapEHPI, da utilizzare in fase di training della rete. Per compensare l'abbondanza di giocatori mancini rispetto ai destri, è stata applicata una trasformazione che nel 50% dei casi scambia i keypoints della parte destra del corpo con quelli di sinistra. In questo modo si va ad equilibrare il numero di mancini e destri all'interno del training set. La successiva prova eseguita nuovamente con i 15 keypoints sui due test set ha confermato l'importanza di questo aspetto. Infatti, nel caso di THETIS l'accuratezza ha subito un miglioramento di un punto percentuale, passando da 87.62% a 88.64%, ma ancor più significativo è risultato essere l'effetto sull'altro dataset, dove si è riscontrato un miglioramento di ben 18 punti percentuali, da 66.81% a 84.56%.

4.4.2 Risultati scheletro

Nonostante la verifica con 15 keypoint avesse già dimostrato la bontà dell'utilizzo della nuova trasformazione, si è deciso comunque di eseguire i test su tutti gli altri casi, sia con che senza questa trasformazione, per essere certi che la tendenza fosse confermata. I risultati relativi all'accuratezza e alle altre metriche utilizzate sono visibili nella Tabella 4.6 dove sono presenti i valori medi sulle quattro classi e nella Tabelle 4.7 dove sono stati riportati i valori di richiamo, precisione e F1 relativi alle singole classi. Nel caso di utilizzo del dataset THETIS, i valori medi di richiamo, precisione e F1 sono stati ottenuti attraverso il calcolo del parametro per la singola classe ed effettuando poi la media sulle 4 classi. Invece quando

è stato utilizzato il dataset personale, poiché sono presenti solo due tipologie di colpo, dritto e rovescio, le medie sono basate solamente su queste due categorie.

KEYPOINTS	# KEYPOINTS	DATASET	COMP. SX/DX	ACCURATEZZA %	PRECISION %	RECALL %	F1 %
Scheletro Completo	15	THETIS	Si	88.64	83.5	80.75	81.75
			No	87.62	82	82.25	82
		Personale	Si	84.56	90.5	88	88
			No	66.18	76.5	73.5	67.5
Polsi, Gomiti, Spalle	6	THETIS	Si	88.38	85	81.5	82.75
			No	88.64	84	93.5	82.75
		Personale	Si	86.76	88	88.5	88.5
			No	57.35	74.5	67	58.5
Polsi, Gomiti	4	THETIS	Si	**88.64**	84.5	82	83.25
			No	**90.91**	87.5	84.5	85.5
		Personale	Si	**91.18**	93.5	91.5	92.5
			No	38.97	67.5	52	42.5
Polsi, Spalle	4	THETIS	Si	88.38	84	83	83.5
			No	86.62	79.25	76	76.5
		Personale	Si	80.15	86.5	83.5	83.5
			No	33.82	75	47.5	33.5
Gomiti, Spalle	4	THETIS	Si	84.6	80.75	76.5	77.75
			No	86.36	82.25	77.75	79
		Personale	Si	47.06	74.5	58	50.5
			No	36.76	73	50.5	36

Tabella 4.6: *Risultati accuratezza medi scheletro*

KEYPOINTS	COLPO	DATASET	COMP. SX/DX	ACCURATEZZA %	PRECISION %	RECALL %	F1 %
Completo	Dritto	THETIS	Si	93.18	89	93	91
	Rovescio			95.45	93	95	94
	Battuta			86.87	88	87	87
	Schiacciata			48.48	64	48	55
	Dritto		No	92.42	90	92	91
	Rovescio			94.70	93	95	94
	Battuta			80.81	89	81	58
	Schiacciata			60.61	56	61	58
	Dritto	Personale	Si	76.14	100	76	86
	Rovescio			100	81	100	90
	Dritto		No	48.86	98	49	65
	Rovescio			97.92	55	98	70
Braccia	Dritto	THETIS	Si	94.70	87	95	91
	Rovescio			92.42	94	92	93
	Battuta			86.87	88	87	87
	Schiacciata			51.51	71	52	60
	Dritto		No	94.70	89	95	92
	Rovescio			93.94	95	94	94
	Battuta			84.85	88	85	86
	Schiacciata			54.54	64	55	59
	Dritto	Personale	Si	82.95	96	83	89
	Rovescio			93.75	83	94	88
	Dritto		No	34.09	100	34	51
	Rovescio			100	49	100	66
Polsi, Gomiti	Dritto	THETIS	Si	95.45	88	95	92
	Rovescio			92.42	95	92	94
	Battuta			85.86	86	86	86
	Schiacciata			54.54	69	55	61
	Dritto		No	100	89	100	94
	Rovescio			92.42	100	92	96
	Battuta			87.88	88	88	88
	Schiacciata			57.58	73	58	64
	Dritto	Personale	Si	90.91	95	91	93
	Rovescio			91.67	92	92	92
	Dritto		No	7.95	78	8	14
	Rovescio			95.83	57	96	71
	Dritto	THETIS	Si	93.94	87	94	91
	Rovescio			92.42	94	92	93
	Battuta			88.89	90	85	88
	Schiacciata			60.61	65	61	62
	Dritto			93.18	90	93	91

Polsi, Spalle	Rovescio		No	93.94	94	94	94
	Schiacciata			89.90	80	87	83
	Battuta			30.30	53	30	38
	Dritto	Personale	Si	72.73	97	73	83
	Rovescio			93.75	76	94	84
	Dritto		No	1.14	100	1	2
	Rovescio			93.75	50	94	65
Gomiti, Spalle	Dritto	THETIS	Si	93.94	82	34	87
	Rovescio			87.12	92	87	89
	Schiacciata			82.83	85	83	84
	Battuta			42.42	64	42	51
	Dritto		No	94.70	83	95	89
	Rovescio			88.64	94	89	91
	Schiacciata			87.88	84	88	86
	Battuta			39.40	68	39	50
	Dritto	Personale	Si	21.59	100	22	36
	Rovescio			93.75	49	94	65
	Dritto		No	4.54	100	5	9
	Rovescio			95.83	46	96	63

Tabella 4.7: *Risultati accuratezza singoli colpi scheletro*

Dai risultati ottenuti si può confermare che la trasformazione swapEHPI è fondamentale per equilibrare il dataset. Infatti, se viene utilizzata con il dataset THETIS si ha, nel caso peggiore un calo del 2%, in un caso nessuna variazione e in alcuni un piccolo aumento dell'accuratezza, mentre si ha sempre un significativo miglioramento, compreso tra il 53% e l'11%, sul dataset personale.
Spostando l'attenzione su quello che è l'obbiettivo principale di questa tesi, cioè l'analisi delle prestazioni al variare del numero di keypoints considerati è meglio effettuare osservazioni separate sui due differenti dataset. Nel caso di THETIS si nota subito che la combinazione che garantisce le prestazioni minori sia la coppia gomiti-spalle, che prevede l'impiego di quattro body joints e paga uno scarto di quattro punti percentuali sull'accuratezza, rispetto alle configurazioni migliori, sia che si utilizzi oppure no la rete addestrata con swapEHPI. Subito dopo troviamo la coppia polsi-spalle che ha un comportamento pari alle migliori configurazioni se si sfrutta la compensazione ma perde di accuratezza nel caso contrario. Rimangono tre combinazioni: scheletro completo, braccia e polsi-gomiti, che hanno tutte e tre prestazioni molto simili, nonostante il numero di keypoints vari di molto, passando dai 15 dello scheletro completo, ai 6 delle braccia e i soli 4 di polsi-gomiti. Un po'a sorpresa, i risultati migliori appartengono alla configurazione con il minor numero di keypoints che raggiunge il 90.91% di accuratezza se non si utilizza la compensazione e l'88,64% nell'altro caso. Per quanto riguarda il dataset personale le prestazioni rispecchiano i risultati ottenuti con THETIS, anche se le differenze tra le varie configurazioni sono più marcate. Nel caso del mancato utilizzo della compensazione si notano valori generalmente molto bassi, che oscillano tra il 33.82% e il 66.81%, ma questo è

dovuto al fatto che la rete è stata addestrata con maggioranza di campioni con giocatori mancini mentre i giocatori del dataset personale sono tutti destri. Se si sfrutta la swapEHPI le prestazioni migliorano di molto e i body joints con i quali vengono raggiunti i risultati migliori si confermano essere polsi-gomiti con 91,18%. Riassumendo i risultati ottenuti per l'accuratezza, si può dire che nonostante preveda l'utilizzo di soli quattro keypoints, la combinazione migliore in assoluto sia quella con polsi e gomiti, seguita da quella con le braccia e dallo scheletro completo. Quindi, considerando i risultati ottenuti, per il caso specifico del tennis, considerare un numero maggiore di punti non è sinonimo di miglioramenti. Osservando i valori medi di richiamo, precisione e punteggio F1, sia per il dataset THETIS che per quello personale vengono confermati i risultati ottenuti considerando l'accuratezza. Le tre migliori opzioni sono nell'ordine: polsi-gomiti, braccia e scheletro completo, mentre la peggiore è gomiti-spalle. I dati relativi alle singole classi mostrano invece la presenza di una tipologia di colpo con prestazioni molto al di sotto delle altre tre, si tratta della schiacciata, che ha valori di precisione, richiamo ed F1 inferiori del 20-25% rispetto alle altre tre classi. Il motivo è probabilmente da imputare al fatto che per questa classe sono presenti molti meno campioni nel dataset di addestramento rispetto alle altre, si hanno 396 elementi per dritto e rovescio, 297 per il servizio e soli 99 per la battuta.
Il fatto che, pur diminuendo il numero di punti considerati, si riescano ad ottenere prestazioni migliori può essere spiegato andando ad osservare la Figura 4.30 e la Figura 4.31. Nella prima abbiamo la rappresentazione di un EHPI che identifica un dritto eseguito da un mancino mentre nella seconda un rovescio eseguito da un mancino. Abbiamo quindi due colpi che richiedono movimenti opposti. Nonostante ciò, è facile notare come le prime tre linee, che si riferiscono a naso, collo e centro delle anche, e le ultime sei che si riferiscono alle giunzioni delle gambe, siano molto simili. Invece le altre righe, che rappresentano le braccia, sono molto differenti. Nel primo caso c'è una transizione da un colore tendente all'arancione ad un verde intenso, nel secondo si passa invece dal verde al rosso. Tutto ciò dimostra che alcuni punti non portano informazioni utili al riconoscimento dei colpi e considerando i risultati ottenuti per l'accuratezza, possiamo dire che causino del rumore.

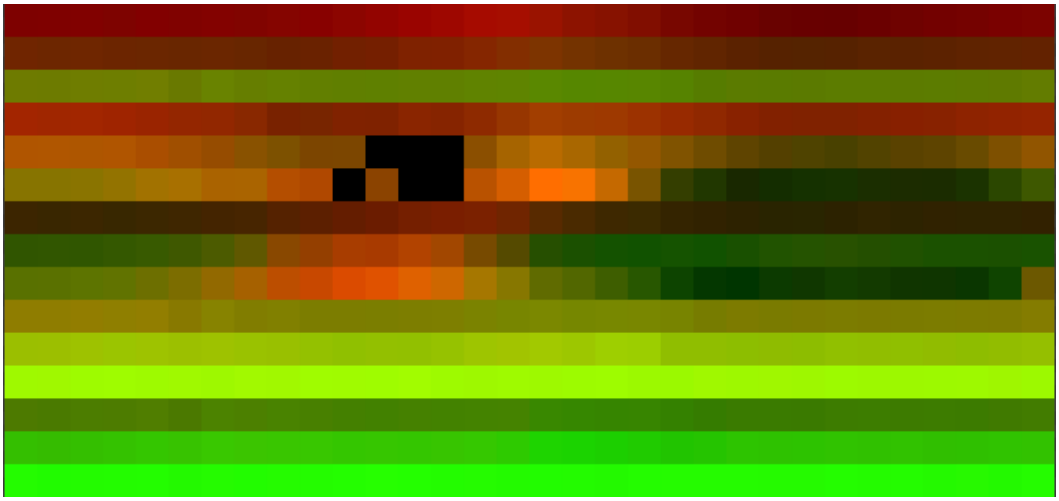

Figura 4.30: *EHPI dritto mancino*

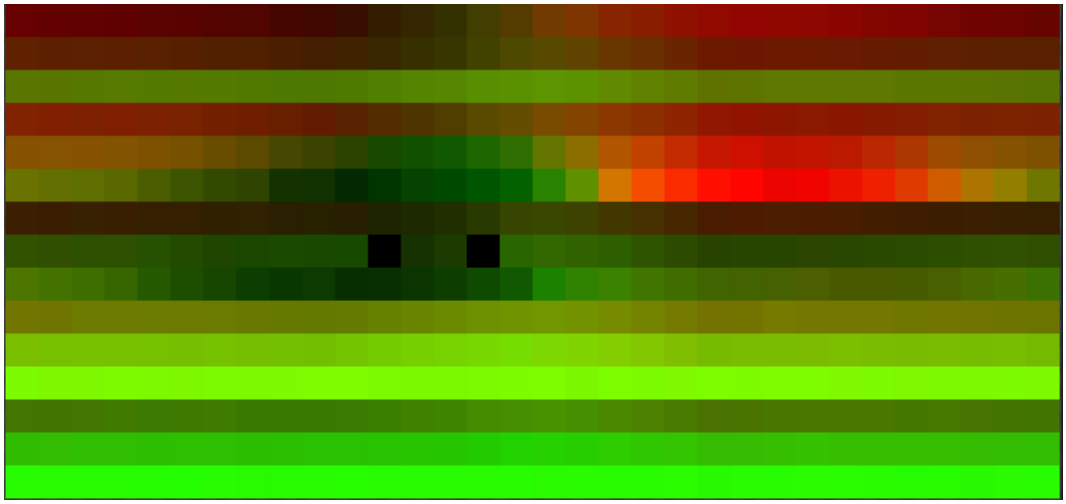

Figura 4.31: *EHPI rovescio mancino*

A questo punto bisognerà valutare quali saranno gli eventuali vantaggi in termini di efficienza temporale, di una configurazione rispetto all'altra, per scegliere quella con il miglior rapporto qualità-tempo.

4.4.3 Esperimenti scheletro e racchetta

Dopo aver provato differenti combinazioni di keypoints dello scheletro, si è deciso di svolgere un'ulteriore analisi andando a considerare anche la racchetta. È necessario fare alcune premesse. L'idea è quella di estrarre una sorta di scheletro della racchetta, con relativi keypoints, ed utilizzarli, come nei casi precedenti, in combinazione alle parti dello scheletro umano. C'è però un problema, la rete che si sta utilizzando ed altre che sono state prese in considerazione, non estraggono questo tipo di informazioni. Una possibilità sarebbe stata quella di ottenere la bounding box e la maschera binaria che rappresenta il contorno e poi ricavare i punti di intersezione tra le due. Questo però richiederebbe un lavoro dispendioso che rallenterebbe l'esecuzione. Il caso ideale sarebbe invece quello di avere una rete che estragga in un colpo solo, lo scheletro umano e i punti della racchetta come fossero un tutt'uno. Per il momento non è stato possibile realizzarlo ma rientrerà sicuramente tra i possibili approfondimenti futuri. Si è optato per utilizzare due diversi elementi per rappresentare la racchetta: la bounding box, rappresentata dalle coordinate dei suoi quattro angoli, oppure il punto centrale della bounding box, pur se consapevole di alcune limitazioni che andrò a spiegare. Il problema di utilizzare questo sistema è che non riesce a rappresentare l'orientamento della racchetta. Come si può vedere nella Figura 4.32, sia che la racchetta punti verso destra, sia che punti verso sinistra, i vertici della bounding box sono sempre etichettati nello stesso modo, rendendo impossibile distinguerne la direzione. In questo modo ad esempio, un dritto eseguito da un mancino potrebbe risultare molto simile a un rovescio eseguito da un destro. Anche per questo motivo andrò a considerare l'utilizzo della bounding box, sia da sola, che insieme ad alcuni punti dello scheletro.

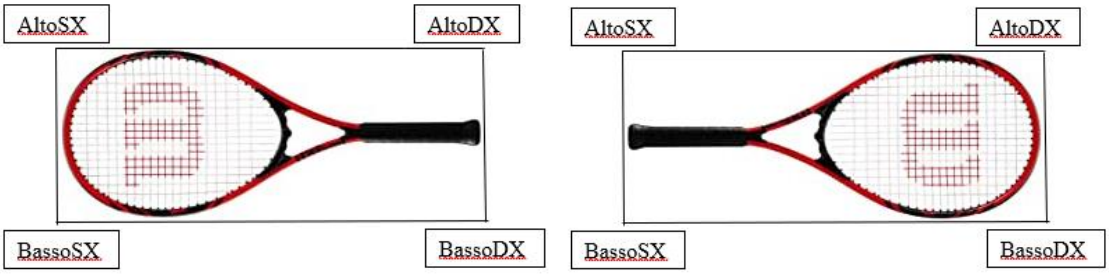

Figura 4.32: *Problema bounding box*

Riguardo alla trasformazione descritta nei paragrafi precedenti, utilizzata per compensare lo squilibrio tra giocatori mancini e destri presente nel dataset THETIS, si è deciso di riproporla anche in questo caso. Si sono però rese necessarie delle modifiche perché, mentre per trasformare uno scheletro da destro a mancino, è sufficiente scambiare la coordinate dei keipoints, ad esempio polso destro con polso sinistro, nel caso della racchetta bisogna spostarla da una parte all'altra del giocatore perché non esistono due racchette come nel caso dei keypoints da poter scambiare. Per riuscire a fare questo tipo di trasformazione, è stata calcolata la posizione della bounding box rispetto al punto centrale delle anche e poi di conseguenza è stata modificata la posizione, spostandola da destra a sinistra o viceversa, della distanza appropriata. Nella Figura 4.33 si vede lo spostamento effettuato. Questo ha imposto di effettuare un ulteriore controllo sulla nuova posizione della bounding box, che potrebbe finire tutta o in parte al di fuori della dimensione dell'immagine. In questo caso i punti che fuoriescono vengono posti uguali a zero. In questo modo si introduce un peggioramento dei dati che non avveniva con la trasformazione swapEHPI.

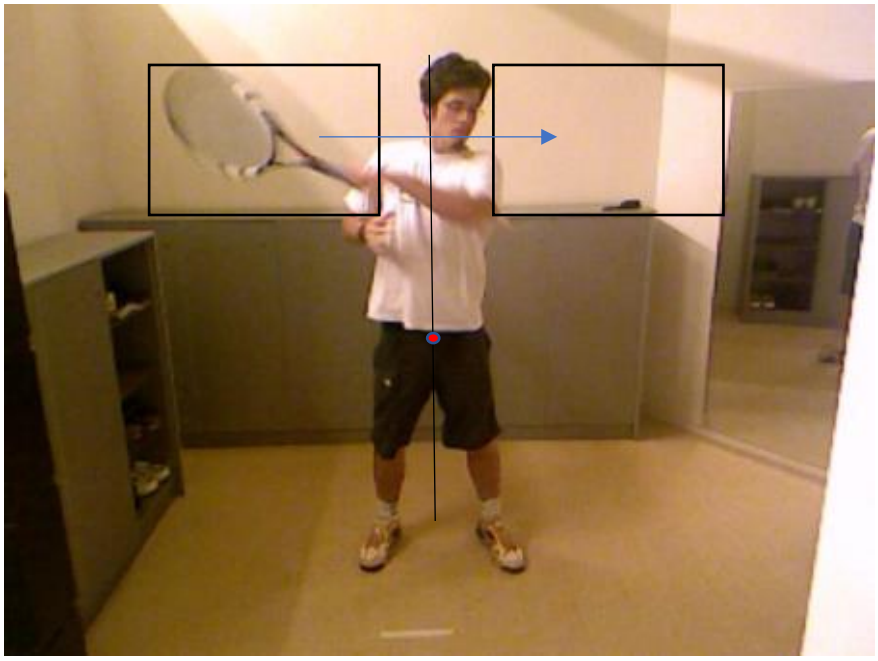

Figura 4.33: *Compensazione bounding box*

L'ultima differenza che abbiamo rispetto all'utilizzo del solo scheletro, è che in questo caso è stato necessario cambiare la rete che effettua la object detection. Infatti, YoloV3 non era sufficientemente accurata nell'identificazione delle racchette. Il numero di racchette riconosciute sui 32 frames relativi ad una clip erano troppo poche, per questo motivo si è deciso di adottare Detectron2 [36] pur se leggermente più lenta.

4.4.4 Risultati scheletro racchetta

I risultati relativi ai valori medi di accuratezza, precisione, richiamo e F1 sono visibili nella Tabella 4.8, mentre quelli relativi alle singole classi sono elencati nella Tabella 4.9.

KEYPOINTS	# KEYPOINTS	DATASET	COMP. SX/DX	ACCURATEZZA %	PRECISION %	RECALL %	F1 %
Boundung Box	4	THETIS	Si	58.33	44.5	48.25	46
			No	72.22	61.75	60.25	59
		Personale	Si	47.06	59	48.5	51
			No	68.38	83.5	69.5	75
BB, Polsi	6	THETIS	Si	84.34	79.5	80	79.5
			No	87.63	80	76.25	77
		Personale	Si	63.23	63	51.5	49
			No	46.32	79.5	57	55
BB, Gomiti	6	THETIS	Si	82.83	78.75	74.25	75.5
			No	84.85	83	75	76
		Personale	Si	61.03	64.5	64.5	61.5
			No	39.71	68	46.5	45
BB, Polsi, Gomiti	8	THETIS	Si	86.36	80.75	82.25	81.25
			No	89.90	84	80.75	82
		Personale	Si	86.03	94.5	85.5	89.5
			No	38.24	65	51	40.5
CentroBB, Polsi, Gomiti	5	THETIS	Si	85.86	80.25	81	82.75
			No	91.16	86.25	84.25	84.75
		Personale	Si	86.76	93.5	86.5	90
			No	55.88	73	64.5	59

Tabella 4.8: *Risultati accuratezza medi scheletro-racchetta*

KEYPOINTS	COLPO	DATASET	COMP. SX/DX	ACCURATEZZA %	PRECISION %	RECALL %	F1 %
Bounding Box	Dritto	THETIS	Si	68.94	55	69	61
	Rovescio			52.27	55	52	53
	Battuta			71.72	68	72	70
	Schiacciata			0	0	0	0
	Dritto		No	87.88	69	88	77
	Rovescio			69.94	84	69	76
	Battuta			77.78	69	78	73
	Schiacciata			6.06	25	6	10
	Dritto	Personale	Si	40.91	78	41	54
	Rovescio			58.33	41	58	48
	Dritto		No	65.91	91	66	76
	Rovescio			70.83	76	73	74
BB, Polsi	Dritto	THETIS	Si	96.21	82	96	89
	Rovescio			78.89	95	79	86
	Battuta			83.84	87	84	86
	Schiacciata			60.61	54	61	57
	Dritto		No	95.45	91	95	93
	Rovescio			94.70	95	95	95
	Battuta			87.88	78	88	83
	Schiacciata			27.27	56	27	37
	Dritto	Personale	Si	90.91	66	91	77
	Rovescio			12.5	60	12	21
	Dritto		No	21.59	95	22	35
	Rovescio			91.67	64	92	75
BB, Gomiti	Dritto	THETIS	Si	96.21	79	96	87
	Rovescio			78.79	94	79	86
	Battuta			85.86	82	86	84
	Schiacciata			36.36	60	36	45
	Dritto		No	94.70	82	95	88
	Rovescio			84.09	93	84	88
	Battuta			90.91	88	91	85
	Schiacciata			30.30	77	30	43
	Dritto	Personale	Si	52.27	82	52	64
	Rovescio			77.08	47	77	59
	Dritto		No	23.86	95	24	38
	Rovescio			68.75	41	69	52
	Dritto	THETIS	Si	94.70	89	95	92
	Rovescio			89.39	94	89	92
	Battuta			77.78	89	78	83
	Schiacciata			66.67	51	67	58

BB, Polsi, Gomiti	Dritto		No	95.45	95	95	95
	Rovescio			96.21	95	96	96
	Schiacciata			89.90	82	90	86
	Battuta			42.42	64	42	51
	Dritto	Personale	Si	87.5	91	88	89
	Rovescio			83.33	98	83	90
	Dritto		No	7.95	78	8	14
	Rovescio			93.75	52	94	67
BBCentro, Polsi, Gomiti	Dritto	THETIS	Si	95.45	85	95	90
	Rovescio			89.39	96	89	93
	Schiacciata			75.76	90	76	82
	Battuta			63.64	50	64	56
	Dritto		No	97.73	93	98	95
	Rovescio			96.21	98	96	97
	Schiacciata			87.88	87	88	87
	Battuta			54.54	67	55	60
	Dritto	Personale	Si	77.78	92	88	90
	Rovescio			85.42	95	85	90
	Dritto		No	35.22	94	35	51
	Rovescio			93.75	52	94	67

Tabella 4.9: *Risultati accuratezza singoli colpi scheletro-racchetta*

Dai risultati ottenuti si può notare come l'utilizzo delle sole coordinate della bounding box della racchetta non diano risultati sufficientemente accurati, abbiamo un massimo di 72.22% per il dataset THETIS senza compensazione e raggiungiamo al massimo il 68.38% utilizzando il dataset personale senza compensazione. Anche i dati su richiamo, precisione ed F1 sono bassi. Si nota però un'anomalia rispetto a tutti gli altri casi, compresi quelli che utilizzano solo parti dello scheletro, nel caso del dataset personale, si raggiungono prestazioni migliori quando non si utilizza la compensazione. Probabilmente sia le prestazioni ridotte che quest'anomalia sono dovute ai problemi evidenziati prima, relativi all'impossibilità di distinguere l'orientamento della racchetta. Invece, negli altri casi in cui andiamo a considerare insieme alla bounding box, anche alcune parti dello scheletro, arriviamo in tutti i casi in cui si utilizza il THETIS, sopra l'80%. In particolare, le configurazioni migliori risultano essere quelle con bounding box, polsi e gomiti con valori massimi su THETIS di 89.90% e con il dataset personale di 86.03, oppure centro della bounding box, polsi e gomiti con 91.16% su THETIS e 86.76% sul dataset personale. In particolare, ritengo quest'ultimo risultato molto significativo, perché la rappresentazione con il centro della bounding box sembrerebbe quella più semplice e consigliata per una rete che riesca ad estrarre questo punto aggiuntivo insieme allo scheletro, come se la racchetta fosse un prolungamento del corpo umano. Prendendo in considerazione i valori medi delle altre tipologie di metriche, nelle occasioni in cui viene utilizzata la bounding box non si segnala nessun comportamento particolare, seguono tutte un andamento paragonabile a quello dell'accuratezza. La

combinazione che sfrutta centro della bounding box, polsi e gomiti, risulta anche in questo caso sorprendente raggiungendo il valore in assoluto più alto per l'F1, 90%. È importante notare che l'aggiunta della bounding box, alla coppia polsi-gomiti, non porti miglioramenti nelle metriche prese in considerazione nonostante l'utilizzo di un numero maggiore di punti. Invece va fatto un discorso a parte per l'utilizzo del centro della bounding box insieme a polsi e gomiti. Perché possiamo notare come quando si testa su THETIS e non si è usata la compensazione si raggiunge il più alto valore di accuratezza in assoluto, 91.16%. Con la rete addestrata con la compensazione per mancini e destri l'accuratezza cala rispetto al solo utilizzo di polsi e gomiti, ma questo potrebbe essere dovuto al fatto che la trasformazione per la compensazione, nel caso delle racchette può far perdere dei dati, come spiegato nel capitolo 4.4.4. Sarebbe interessante, in uno studio futuro disporre di un dataset equilibrato tra destri e mancini e poter testare questo tipo di configurazione. Come è già stato notato nell'analisi dei risultati relativi all'utilizzo del solo scheletro umano, anche in questo caso i valori di precisione, richiamo e F1 delle singole classi mostrano che, il tipo di colpo "battuta" ottiene prestazioni di molto inferiori alle altre classi.

4.4.5 Esempi significativi

In questa sezione vengono considerate le matrici di confusione di alcune configurazioni, per evidenziare quali tipi di movimento vengono meglio riconosciuti e quali meno.

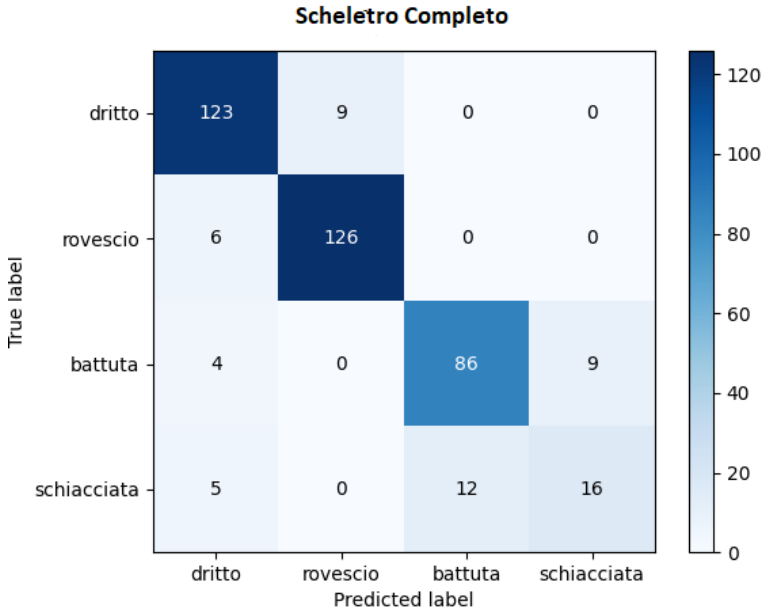

Figura 4.34: *Confusion martrix scheletro completo*

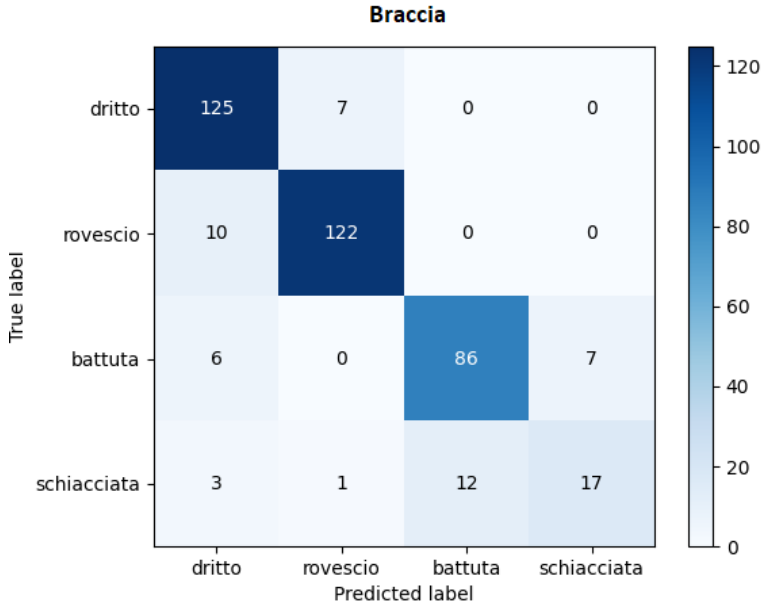

Figura 4.35: *Confusion matrix braccia*

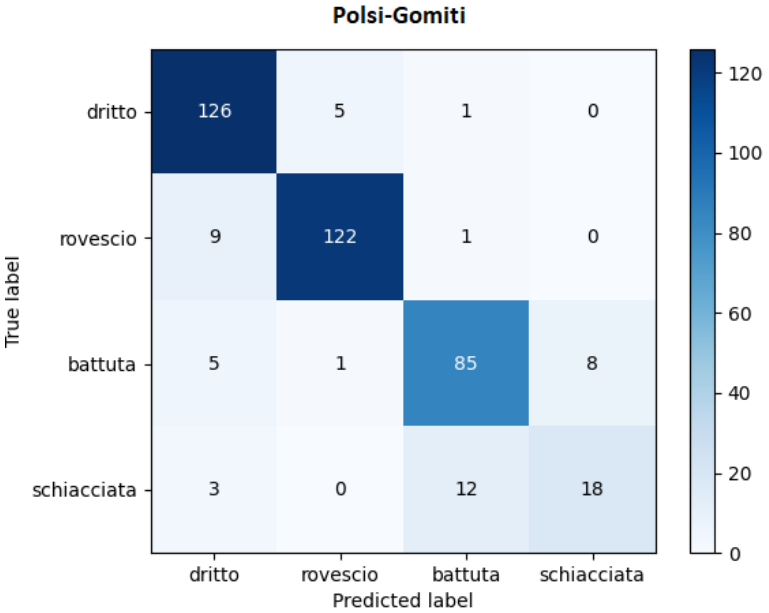

Figura 4.36: *Confusion matrix polsi-gomiti*

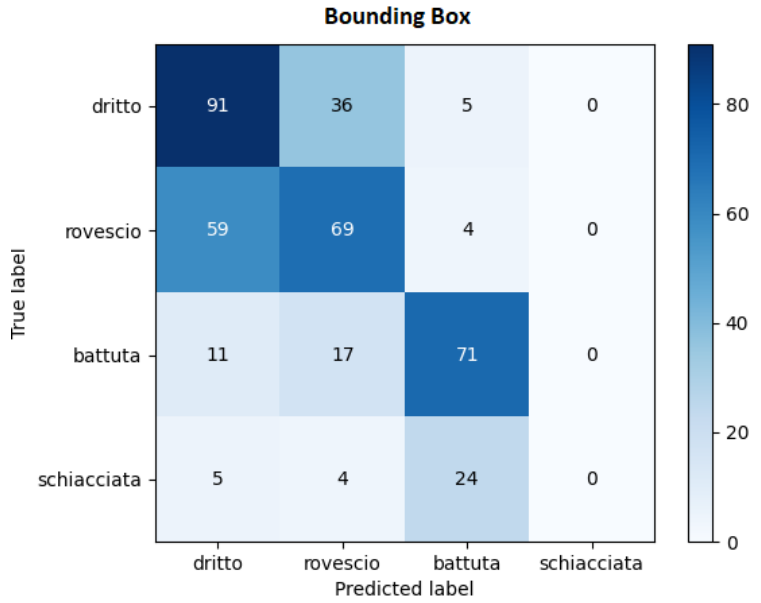

Figura 4.37: *Confusion matrix bounding*

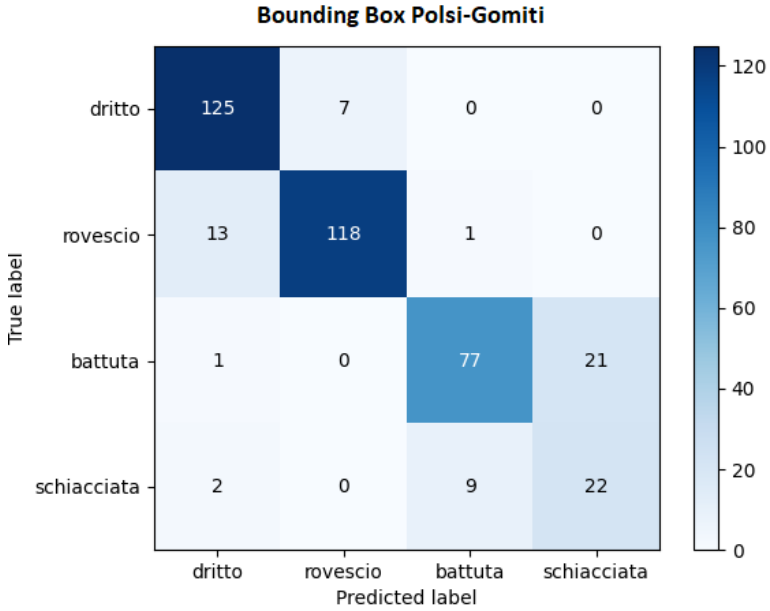

Figura 4.38: *Confusion matrix bounding box polsi-gomiti*

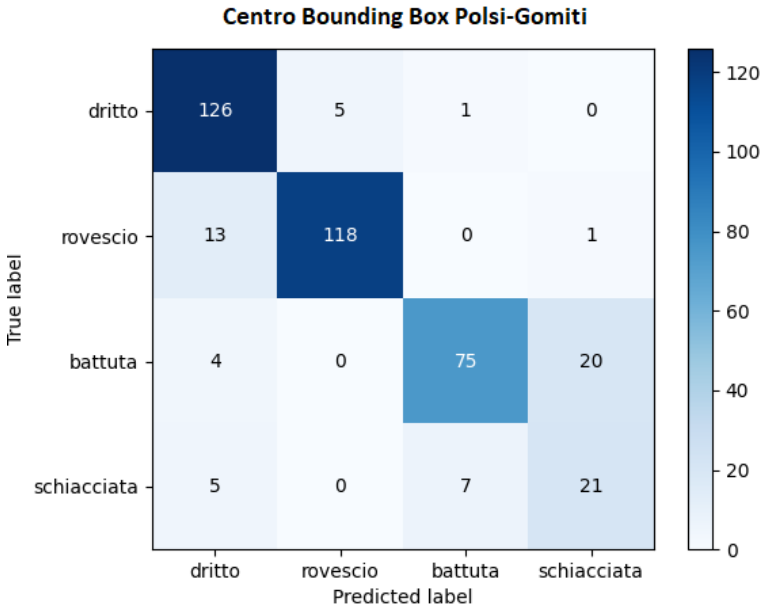

Figura 4.39: *Confusion matrix centro bounding box polsi-gomiti*

Le configurazioni analizzate sono: scheletro completo, solo braccia, polsi e gomiti, bounding box della racchetta, bounding box racchetta, polsi e gomiti e centro della bounding box, polsi e gomiti. Tutti i casi si riferiscono all'utilizzo del dataset THETIS e delle reti addestrate utilizzando la compensazione tra mancini e destri.
Analizzando il quadro generale è evidente che dritto e rovescio siano le classi che meglio vengono identificate dal modello, seguite dalla battuta ed infine dalla schiacciata, che mostra risultati particolarmente negativi. Questo andamento rispecchia la composizione del training set utilizzato per l'addestramento, che contiene 396 campioni per il dritto, 396 per il rovescio, 297 per la battuta e solo 99 per la schiacciata. Le difficoltà nel distinguere alcune classi è probabile che derivino da questo sbilanciamento nel dataset.
Effettuando invece un paragone tra le diverse configurazioni notiamo che, come già evidenziato dai valori delle metriche ottenuti, per il caso di utilizzo della sola bounding box i risultati sono pessimi ed in particolare per la schiacciata le predizioni sono totalmente errate. Ai fini dello studio effettuato è importante notare come passando dalla considerazione dell'intero scheletro composto da 15 punti, alle sole braccia con 6 o addirittura ai soli polsi e gomiti con 4, le statistiche riguardanti i vari colpi rimangono molto simili. Quando si introduce l'utilizzo della bounding box della racchetta o del suo punto centrale, in combinazione con polsi e gomiti, viene meglio identificato il colpo della schiacciata rispetto alle altre configurazioni.

4.5 Valutazione dell'efficienza temporale

Durante lo studio sull'accuratezza l'attenzione è stata principalmente focalizzata sulla sola rete per l'action recognition, la quale riceveva in input un file csv precedentemente preparato in base alla configurazione desiderata. Per questo motivo la rete della pose estimation poteva essere usata per estrarre l'intero scheletro per poi annotare sul file csv solo i keypoints necessari. Passando alla valutazione delle prestazioni temporali è invece necessario considerare la pipeline nella sua interezza ed in particolare sfruttare una rete della pose estimation che estragga solo i keypoints relativi alla configurazione presa in considerazione in quel momento. Si è quindi reso necessario eseguire più addestramenti della rete per specializzarla nell'estrazione di un numero ristretto di keypoints e poter così verificare se esista un risparmio in termini di tempo.
Per effettuare questa analisi si sono presi in considerazione dieci clip tra quelle presenti nel dataset personale e le si sono analizzate in sequenza. Per tener traccia del tempo impiegato, è stata sfruttata la funzione time.perf_counter() che permette di ottenere una misurazione del tempo molto accurata, grazie all'utilizzo del clock con la maggior risoluzione disponibile nel sistema. La misurazione è espressa in secondi, in particolare è stata considerata fino alla seconda cifra decimale. Per ciascuna clip, l'intervallo di tempo preso in considerazione ha inizio con l'operazione di estrazione dello scheletro umano dai 32 frames e termina una volta effettuata la predizione del tipo di colpo attraverso l'analisi dell'EHPI. Tutti gli intervalli sono poi sommati per ottener il tempo totale necessario per analizzare le 10 clips e calcolare il tempo medio per l'elaborazione di una singola clip. La misurazione è stata ripetuta cinque volte per ciascuna configurazione, ed è stato considerato il tempo medio.
Le combinazioni di keypoints di corpo umano e racchetta considerati, sono le stesse analizzate nel caso della valutazione dell'accuratezza, la rete utilizzata per l'action recognition è stata addestrata su THETIS utilizzando la compensazione tra mancini e destri.

4.5.1 Risultati scheletro

KEYPOINTS	# KEYPOINTS	TEMPO TOTALE (s)	TEMPO MEDIO (s)
Scheletro completo	15	5.33	0.53
Polsi, Gomiti, Spalle	6	4.41	0.44
Polsi, Gomiti	4	4.47	0.47
Polsi, Spalle	4	4.45	0.45
Gomiti, Spalle	4	4.49	0.49

Tabella 4.10: *Risultati tempo scheletro*

Dall'analisi della Tabella 4.10 emerge come sia presente un miglioramento nei tempi di esecuzione tra la configurazione con 15 body joints e tutte le altre. Utilizzando l'intero scheletro sono necessari 5.33 secondi per l'analisi di tutte e dieci le clips, con un tempo medio di 0.53 secondi per clip, mentre con le altre configurazioni abbiamo tutti tempi molto simili, che vanno da un minimo di 4.41 secondi, ad un massimo di 4.49 secondi. La conseguenza di

questi risultati è un incremento nelle prestazioni temporali compreso tra il 15.75% e il 17.16% se viene utilizzato un numero ridotto di punti.

4.5.2 Risultati scheletro racchetta

KEYPOINTS	# KEYPOINTS	TEMPO TOTALE (S)	TEMPO MEDIO (S)
Bounding Box	4	0.79	0.079
BB, Polsi	6	4.83	0.48
BB, Gomiti	6	4.81	0.48
BB, Polsi, Gomiti	8	4.75	0.47
CentroBB, Polsi, Gomiti	5	4.70	0.47

Tabella 4.11: *Risultati tempo scheletro-racchetta*

Osservando la Tabella 4.11 salta immediatamente all'occhio il valore sorprendentemente basso nel caso dell'utilizzo della sola bounding box, solamente 0.79 secondi complessivi con un tempo medio di 0.079 secondi. Tutto ciò si spiega col fatto che con questa soluzione si sfrutta solamente la bounding box della racchetta, che viene estratta dalla rete di object detection, e quindi non viene mai sfruttata la rete che fa pose estimation ed estrae lo scheletro umano. In tutti gli altri casi i dati sono invece in linea con i risultati ottenuti per le configurazioni con meno punti del corpo umano analizzate al punto precedente.

Capitolo 5

Conclusioni

In questo che è il capitolo conclusivo della tesi, vengono in un primo momento messi a confronto i risultati ottenuti per l'accuratezza e quelli ottenuti per l'efficienza temporale, in modo da stabilire quali siano le migliori combinazioni di body joints da utilizzare per avere un'applicazione veloce ma allo stesso tempo accurata e rispondere così a quello che è l'obbiettivo principale dello studio. Successivamente si fanno alcune considerazioni su quelli che potrebbero essere studi ed esperimenti futuri.

5.1 Analisi dei risultati

L'obbiettivo di questa tesi è stato quello di verificare se e come, diminuendo il numero di body joints presi in considerazione, variassero le prestazioni in termini di accuratezza e tempi di esecuzione, di una pipeline di reti per il riconoscimento delle azioni del tennis. Il punto di partenza era rappresentato dall'utilizzo di 15 keypoints che rappresentano le giunzioni dell'intero scheletro, e sono state poi considerate configurazioni con un minor numero di body joints, come quelli delle sole braccia. In aggiunta si è provato anche ad utilizzare una rappresentazione della racchetta, da sola o in combinazione con alcuni punti del corpo. Facendo un'analisi conclusiva e complessiva dei risultati ottenuti possiamo dire che abbiamo una configurazione che si discosta totalmente dalle altre sia per quanto riguarda l'accuratezza sia per l'efficienza temporale, ed è quella che utilizza la sola bounding box della racchetta. Presenta infatti ottime prestazioni temporali, ma raggiunge livelli di accuratezza molto inferiori rispetto a tutti gli altri casi presi in considerazione. Per la sua grande velocità di elaborazione potrebbe essere interessante approfondirne lo studio, cercando di migliorarne la qualità delle predizioni, magari riuscendo a trovare un sistema che sia in grado di risolvere il problema dell'orientamento della bounding box spiegato nel capitolo 4.4.4. Messo da parte questo caso particolare, in tutte le altre circostanze sono stati riscontrati valori abbastanza simili tra loro, anche se con alcune differenze significative. In particolare, si è visto che l'utilizzo dell'intero scheletro, con i suoi 15 keypoints, non è la rappresentazione migliore da utilizzare, ma diminuendo i punti presi in considerazione e concentrandosi sulle parti del corpo più rappresentative per il riconoscimento dei colpi del tennis si ottengono prestazioni migliori sia riguardo all'accuratezza che alle prestazioni temporali. I body joints che garantiscono le migliori performance sono la coppia polsi-gomiti, che permette di raggiungere livelli di accuratezza maggiori, con punte del 91%, ed allo stesso tempo di avere un risparmio, in termini di velocità di predizione, del 17% rispetto all'utilizzo dei 5 keypoints.

5.2 Approfondimenti futuri

In questa tesi si è visto quali possono essere le prestazioni raggiungibili diminuendo il numero di punti dello scheletro presi in considerazione, ma secondo i miei studi ci sono alcuni accorgimenti e miglioramenti che potrebbe essere utile adottare in futuro. In primis il dataset THETIS può essere migliorato e ampliato. È già stato sottolineato come sia di fondamentale importanza poter usufruire di un dataset ampio, equilibrato e vario per ottenere analisi quanto più precise e affidabili. In questo caso specifico il dataset andrebbe ampliato inserendo campioni per la classe "battuta" ma soprattutto per quella "schiacciata" che al momento risulta essere deficitaria, con appena 165 elementi su un totale di 1980 suddivisi in quattro differenti classi e ciò porta valori di precisione, richiamo ed F1 molto bassi per questa classe, come visto nel capitolo 4.4. In aggiunta si potrebbero inserire anche clip estratte da situazioni reali di gioco e non solamente riprodotte ad hoc da giocatori al di fuori del contesto della partita vera e propria, sempre nell'ottica di arricchire il dataset e permettere al modello di apprendere attraverso dati più simili al caso di applicazione. Il secondo aspetto riguarda la rappresentazione della racchetta e il modo in cui viene estratta. L'obbiettivo è quello di addestrare la rete per la pose estimation in modo tale che sia in grado di estrarre il o i punti che identificano la racchetta come se fossero una estensione dello scheletro umano. Per riuscire ad ottenere questo obbiettivo è necessario disporre di un dataset composto da persone che impugnano una racchetta con le relative annotazioni sulla posizione dei punti da estrarre e successivamente utilizzarlo per addestrare la rete della pose estimation. Grazie a questi miglioramenti riguardanti il dataset e la rete della pose estimation sarà possibile portare avanti lo studio, effettuando un'analisi più accurata per quanto riguarda l'utilizzo della rappresentazione della racchetta e migliorare l'efficienza delle predizioni. In questo modo si disporrà di una solida base per realizzare un'applicazione che identifichi le azioni del tennis in tempo reale.

Bibliografia

[1] K. Simonyan, A. Zisserman (2014). *Two-Stream Convolutional Networks fia Action Recognition in Videos.*
[2] C. Feichtenhofer, A. Pinz, A. Zisserman (2016). *Convolutional Two-Stream Network Fusion for Video Action Recognition.*
[3] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri (2015). *Learning Spatiotemporal Features with 3D Convolutional Networks.*
[4] J. Carreira, A. Zisserman (2018). *Quo Vadis, Action Recognition? A New model and the Kinetics Dataset.*
[5] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, T. Darrell (2016). *Long-term Recurrent Convolutional Network for Visual Recognition and Description.*
[6] F. Rosenblatt (1958). *The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain.*
[7] T. Kayzoglu (2001). *An Investigation of the Design and Use of Feed-Forward Artificial Neural Networks.*
[8] D.E. Rumelhart (1986). *Sequential Thought Processes in PDP Models.*
[9] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel (1989). *Backpropagation Applied to Handwritten Zip Code Recognition.*
[10] V. Nair, G.E. Hinton (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines.*
[11] B. Xu, N. Wang, T. Chen, M. Li (2015). *Empirical Evaluation of Rectified Activations in Convolution Network*
[12] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone (1984). *Classification and Regression Trees.*
[13] J.R. Quinlan (1993). *Programs for Machine Learning.*
[14] S. Aubry, S. Laraba, J. Tilmanne, T. Dutoit (2018). *Action Recognition Based on 2D Skeletons Extracted from RGB Videos.*
[15] Z. Cao, T. Simon, S. -E. Wei, Y. Sheikh (2017). *Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields.*
[16] A. Shahroudy, J. Liu, T.-T. Ng, G. Wang (2016). *NTU RGB+D: A Large A Scale Dataset for 3D Human Activity Analysis.*
[17] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J.Dally, K. Keutzer (2016). *Squeezenet: Alexnet-level Accuratezza with 50x Fewer Parameters and 0.5 mb Model Size.*
[18] G. Huang, Z. Liu, K. Q. Weinberger, L. van der Maaten (2017). *Densely Connected Convolutional Networks.*
[19] A. Krizhevsky, I. Sutskever, G. E. Hinton (2012). *Imagenet Classification with Deep Convolutional Neual Networks.*

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich (2015). *Going Deeper with Convolutions*.
[21] K. He, X. Zhang, S. Ren, J. Sun (2016). *Deep Residual Learning for Image Recognition.*
[22] K. Simonyan, A. Zisserman (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition.*
[23] J. Cai, X. Tang, H. Xia, M. Chi (2018). *RGB Video Based Tennis Action Recognition Using a Deep Historical Long Short-Term Memory.*
[24] S. Gourgari, G. Goudelis, K. Karpouzis, S. Kollias (2013). *THETIS: Three Dimensional Tennis Shots. A Human Action Dataset.*
[25] H. Kuehne, H. Jhuang, E. Garrote (2011). *A Large Video Database for Human Motion Recognition.*
[26] C. Szegedy, V. Vanhoucke, S. Ioffe (2016). *Rethinking the Inception Architecture for Computer Vision.*
[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei (2009). *ImageNet:A Large-Scale Hierarchical Image Database.*
[28] D. Ludl, T. Gulde, C. Curio (2019). *Simple Yet Efficient Real-Time Pose-Based Action Recognition.*
[29] J. Redmon, A. Farhadi (2018). *YOLOv3: An Incremental Improvement.*
[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, C. L. Zitnick (2014). *Microsoft COCO: Common Objects in Context.*
[31] B. Xiao, H. Wu, Y. Wei (2018). *Simple Baselines for Human Pose Estimation and Tracking.*
[32] M. Andriluca, L. Pishchulin, P. Gehler, B. Schiele (2014). *2D Human Pose Estimation: New Benchmark and State of the Art Analysis.*
[33] J.-Y. Bouguet (2000). *Pyramidal Implementation of Lucas Kanade Feature Tracker.*
[34] X. Glorot, Y. Bengio (2010). *Understanding the Difficulty of Training Deep Feedforward Neural Networks.*
[35] Y. Wu, A. Kirillow, F. Massa W.-Y. Lo, R. Girshick (2019). *Detectron2.*
[36] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, M. J. Black (2013). *Towards Understanding Action Recognition.*